Power Control and Clustering in Ad Hoc Networks

Vikas Kawadia and P. R. Kumar
Department of Electrical and Computer Engineering, and Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign, 1308 West Main St. Urbana, IL-61801.
E-mail:{kawadia,prkumar}@uiuc.edu

***Abstract*— In this paper, we consider the problem of power control when nodes are non-homogeneously dispersed in space. In such situations, one seeks to employ per packet power control depending on the source and destination of the packet. This gives rise to a joint problem which involves not only power control but also clustering. We provide three solutions for joint clustering and power control.**

The first protocol, CLUSTERPOW, aims to increase the network capacity by increasing spatial reuse. We provide a simple and modular architecture to implement CLUSTERPOW at the network layer.

The second, Tunnelled CLUSTERPOW, allows a finer optimization by using encapsulation, but we do not know of an efficient way to implement it.

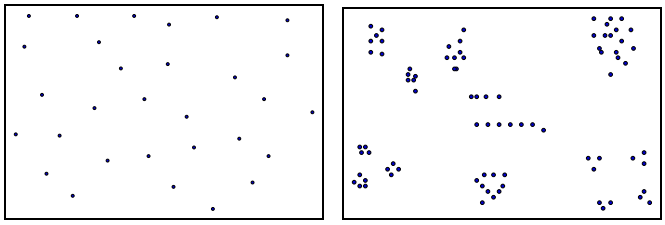
The last, MINPOW, whose basic idea is not new, provides an optimal routing solution with respect to the total power consumed in communication. Our contribution includes a clean implementation of MINPOW at the network layer without any physical layer support.

We establish that all three protocols ensure that packets ultimately reach their intended destinations. We provide a software architectural framework for our implementation as a network layer protocol. The architecture works with any routing protocol, and can also be used to implement other power control schemes. Details of the implementation in Linux are provided.

I. Introduction

The power control problem is to choose the transmit power level for every packet in a wireless ad hoc network. The per-packet choice is to be guided by several considerations. The choice of transmit power, and thus the range, affects the traffic-carrying capacity of the network, and it was shown in [1] that after taking into consideration the additional relaying burden of using small hops versus the interference caused by long hops, it is optimal to reduce the transmit power level. The choice of power level also affects battery life. In [2], it was shown that for the commonly used propagation path loss attenuation models, low power levels are commensurate with power optimal routing. This was done by showing that the latter necessarily results in planar graphs of power optimal routes, with only nearby nodes exchanging packets. Moreover, power control affects routing since the ranges of the transmitters depend on the transmit power levels. A further factor to be considered is that power control affects packet end-to-end latency. With small power levels, a packet will take a large number of hops which may linearly increase latency due to the packetization delay at each hop.

This material is based upon work partially supported by DARPA under Contract No. N00014-01-1-0576, AFOSR under Contract No. F49620-02-1-0217, DARPA/AFOSR unfer Contract No. F49620-02-1-0325, NSF under Contract No. NSF ANI 02-21357, USARO under Contract Nos. DAAD19-01010-465 and DAAD19-00-1-0466. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the above agencies.



(a) Homogeneous spatial dispersion of nodes

(b) Nodes non-homogeneously dispersed

Fig. 1. Homogeneous vs clustered networks

Given this complexity of considerations, how does one i) conceptualize the power control problem, ii) determine how to trade off the multiple objectives of capacity, battery life and latency, and iii) develop a protocol which is modular and elegant enough to work with the OSI architecture?

A first cut solution was presented in [2]. A network layer protocol, called COMPOW, was developed which ensured that the transmit power used by all the nodes would converge to a common power level: the lowest power level at which the network is connected. A software architecture was also developed with the requisite properties of modularity and layering. An implementation in the Linux kernel was also provided.

When nodes are homogeneously dispersed in space, as in Fig. 1(a), the choice of a common transmit power level has several appealing properties as noted above. However, when nodes are non-homogeneously dispersed as in Fig. 1(b), then the lowest common power level for network connectivity is hostage to the outlying nodes which are far from others. For example, in Fig. 2, all nodes except node F are mutually reachable at 1 mW, i.e., they form a 1 mW cluster, but F is reachable only by using a power level of 100 mW. The COMPOW algorithm, designed to converge to the lowest power level such that the network is connected, will thus converge to 100 mW, even though 1 mW is enough for most communications.

Such non-homogeneous scenarios are ripe for *clustering*. One wishes to group nodes into clusters, with several clusters at power level k forming a cluster at power level $k+1$. Such

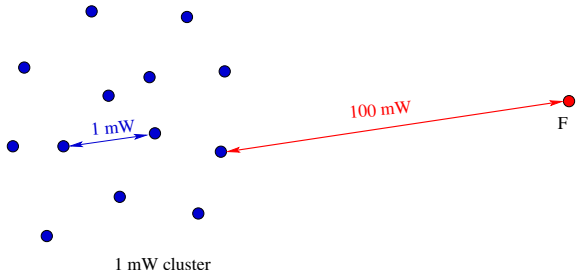


Fig. 2. A common power level is not appropriate for non-homogeneous networks.

clustering of nodes cannot simply be based on geographical co-ordinates since obstacles and shadowing may prevent two nodes from forming a wireless link, even if they are in close proximity.

Power control should also be done in conjunction with routing, since it needs to keep connectivity in mind, which is known only through the existence of routes. Conversely, routing depends on power control since the power level dictates what links are available for routing. All these interdependences need to be resolved in a manner compatible with the layered and modular architecture for networking systems.

In this work we consider the power control problem and the clustering problem in non-homogeneous networks, that is, where nodes can exist in clusters. The goal is to choose the transmit power level, so that most of the intra-cluster communication is at lower transmit power levels, and a higher transmit power level is used only when going to a different cluster. We provide dynamic and implicit clustering of nodes based on transmit power level, rather than on addresses or arbitrary geographical regions. There are no leader or gateway nodes. The clustered structure of the network is automatically manifested in the way routing is done. We propose two solutions: the CLUSTERPOW power control protocol and the Tunnelled CLUSTERPOW power control protocol, which aim to increase network capacity by increasing spatial reuse. The CLUSTERPOW protocol has been implemented in the Linux kernel.

We also present the MINPOW routing and power control protocol, which is a distance vector routing protocol with power consumption as the link cost. We consider the problem of effectively estimating the cost, and finally provide a simple and efficient implementation of MINPOW in the Linux kernel without any physical layer support.

A. A Brief Survey of the Literature

Most work on power control problem can be classified into one of three categories. The first class comprises of strategies to find an optimal transmit power to control the connectivity properties of the network, or a part of it. As noted earlier, in [2] power control is conceptualized as a network layer problem , and the COMPOW protocol is presented. In [3] it is proposed that each node adjust its transmit power so that its degree (number of one-hop neighbors) is bounded. In [4], it is proposed to use transmit power control to optimize the average end-to-end network throughput by controlling the degree of a node. A distributed topology control algorithm using direction information is proposed in [5]. The second class of approaches could be called power "aware" routing. Most schemes use some shortest path algorithm with a power based metric, rather than a hop count based metric. Some suggestions for the metric in [6] include energy consumed per-packet, time to network partition, variance in battery life of nodes, and the energy cost per-packet, while other schemes in this class are proposed in [7], [8], and [9]. The third class of approaches aim at modifying the MAC layer. In [10] it is suggested to modify IEEE 802.11's handshaking procedure to allow nodes to transmit at a low power level, while [11] proposes enabling nodes to power themselves off when not actively transmitting or receiving. Some other schemes focus on energy conservation by putting nodes to sleep, using either location information [12], or local topology information obtained using broadcast messages [13].

In a general sense, the clustering problem is one of classifying nodes hierarchically into equivalence classes, according to certain attributes. These attributes could be node addresses [14], geographical regions or zones [15], or a small neighborhood (typically 1 or 2 hop) of certain nodes elected as cluster-heads or leaders [16]. The leader election, or the cluster set up phase, uses heuristics like node addresses, node degrees, transmission power, mobility, or more sophisticated node weights combining the above attributes, as in WCA [17], and in DCA [18]. Cluster-heads can be used for routing, for resource allocation among nodes in its cluster [19], and for network management. Cluster-heads can be used as base stations as in cellular networks in [20]. Most schemes for ad hoc networks maintain clusters in dynamic network conditions in addition to forming them. Gateway nodes are also elected in some cases to ensure connectivity among clusters. Clustering can also be done implicitly without electing cluster-heads and gateways, as in ZRP [21], and in GPS based hierarchical link state routing [15]. Some algorithmic aspects of clustering are analyzed in [22] and [23].

One goal of clustering could be to reduce route discovery overhead (by address space aggregation or by localizing control messages) to optimize resources like battery power and network capacity, or to simplify addressing and management. IP subnetting is a good example of clustering for routing efficiency, as well as ease of management. Reduction in routing overhead is achieved by backbone formation in spine based routing [24], and in VDBP [25], where a fraction of the nodes, called the backbone nodes, assume responsibility for route discovery. However, address space aggregation, where a node's address is determined by the cluster it belongs to, seems feasible only in quasi-static or infrastructure type ad hoc networks as in Landmark [26], or in networks with a natural logical hierarchy.

II. The CLUSTERPOW Power Control Protocol

The CLUSTERPOW power control protocol has been designed for power control, clustering and routing in non-

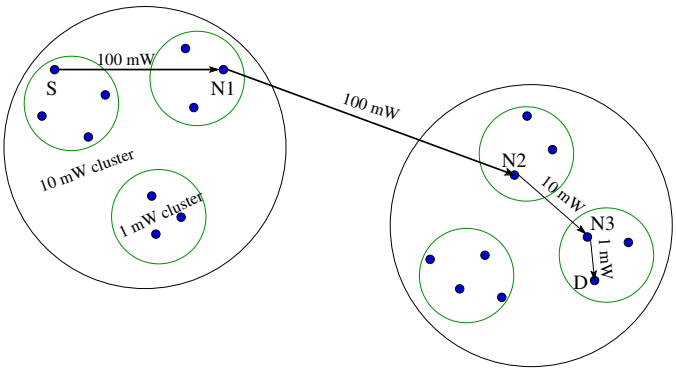


Fig. 3. Routing by CLUSTERPOW in a typical non-homogeneous network.

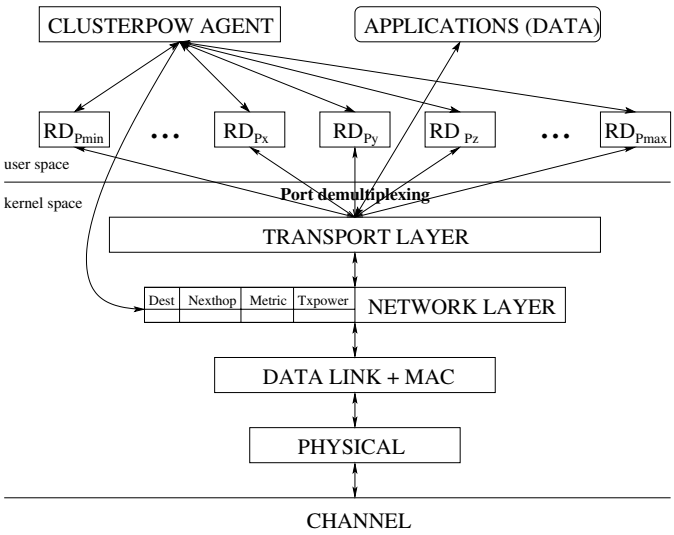


Fig. 4. Architectural design of CLUSTERPOW.

homogeneous networks. A route in CLUSTERPOW generally consists of hops of different transmit power such that the clustered structure of the network is respected. The algorithm consists of simply using the lowest transmit power level p, such that the destination is reachable (in multiple hops) by using power levels no larger than p. This algorithm is executed at the source, and at every intermediate node along the route from the source to the destination for every packet.

The route resulting from running this algorithm in a typical clustered network is illustrated in Fig. 3. The network has three levels of clustering corresponding to power levels of 1 mW, 10 mW and 100 mW, the whole network being the 100 mW cluster. To get from the source node S to the destination D, a power level of 100 mW is used at each hop until the packet gets to the 10 mW cluster to which the destination belongs. Then 10 mW is used at each hop until the 1 mW cluster to which the destination belongs is reached, and finally a sequence of 1 mW hops gets the packet to the destination. It should be noted that transmit power control in this fashion leads to automatic clustering in the network.

A. CLUSTERPOW Architecture

We now describe the architectural design to implement the algorithm in a simple way, and to integrate it into the IP stack as a network layer protocol. The architecture of CLUSTERPOW involves running multiple routing daemons, one corresponding to each power level P_i in a finite and discrete set of feasible power levels. These routing daemons build their own separate routing tables RT_{P_i} by communicating with their peer routing daemons of the same power level at other nodes, using hello packets transmitted at power level P_i. This idea of parallel modularity at the network layer is illustrated in Fig. 4. The next hop in CLUSTERPOW is determined by consulting the lowest power routing table in which the destination is reachable. That is, for every destination D, the entry (row) in the kernel routing table is copied from the lowest power routing table in which D is reachable, i.e., has a finite metric. The kernel routing table has an additional field, transmit power (txpower) for every entry, which indicates the power level to be used when routing packets to the next hop for that destination.

Consider the network in Fig. 3. The user space routing tables at each power level and the kernel IP routing table at each of the nodes are shown in Fig. 5. At node S, the destination D appears (i.e., has a finite metric) only in the 100 mW routing table, with N1 as the next hop. Thus this entry is copied into the kernel IP routing table and used for routing. The situation is similar for N1, since the destination appears only in the 100 mW routing table, with N2 as the next hop. At N2, however, the lowest power level at which D is reachable is 10 mW. So this is used for routing and the packet is sent to N3, which has D in its 1 mW routing table. Hence the final hop of the packet is at 1 mW. This architecture provides a simple way to implement the CLUSTERPOW algorithm.

B. Properties of CLUSTERPOW

The CLUSTERPOW power control protocol has the following properties:

1) *CLUSTERPOW provides implicit, adaptive, and distributed clustering based on transmit power.* Clustering is implicit because there are no cluster-head or gateway nodes. It is dynamic and distributed, because it is integrated with a routing protocol which has these properties. The clusters are determined by reachability at a given power level, and the hierarchy of clustering could be as deep as the number of power levels.
2) *The routes discovered consist of a non-increasing sequence of transmit power levels.* When a particular power level p is used, the destination is present in the routing table corresponding to p, and there exists a path of power level at most p from the current node to the destination. Thus, further "downstream", a higher transmit power will not be used by this algorithm.
3) COMPOW *is a special case of* CLUSTERPOW. If the network is homogeneous, CLUSTERPOW will use a common power level throughout the network.
4) *CLUSTERPOW can be used with any routing protocol, reactive or proactive.* In the case of a proactive routing

1 mW Routing Table

Dest	NextHop	Metric
D		Inf

10 mW Routing Table

Dest	NextHop	Metric
D		Inf

100 mW Routing Table

Dest	NextHop	Metric
D	N1	3

Node S — Kernel IP Routing Table

Dest	NextHop	Metric	TxPower
D	N1	3	100 mW

1 mW Routing Table

Dest	NextHop	Metric
D		Inf

10 mW Routing Table

Dest	NextHop	Metric
D		Inf

100 mW Routing Table

Dest	NextHop	Metric
D	N2	3

Node N1 — Kernel IP Routing Table

Dest	NextHop	Metric	TxPower
D	N2	3	100 mW

1 mW Routing Table

Dest	NextHop	Metric
D		Inf

10 mW Routing Table

Dest	NextHop	Metric
D	N3	2

100 mW Routing Table

Dest	NextHop	Metric
D	D	1

Node N2 — Kernel IP Routing Table

Dest	NextHop	Metric	TxPower
D	N3	2	10 mW

1 mW Routing Table

Dest	NextHop	Metric
D	D	1

10 mW Routing Table

Dest	NextHop	Metric
D	D	1

100 mW Routing Table

Dest	NextHop	Metric
D	D	1

Node N3 — Kernel IP Routing Table

Dest	NextHop	Metric	TxPower
D	D	1	1 mW

Fig. 5. Routing tables for all power levels, and the kernel IP routing table, at all the nodes in the network of Fig. 3.

protocol (e.g., DSDV [27]), all the routing tables at different power levels are maintained through hello packets and the kernel routing table is composed using them. For a reactive or on-demand routing protocol like AODV [28], route discovery requests can be sent out at all the power levels available. The lowest power level which results in a successful route discovery can be used for routing the packet.

5) *CLUSTERPOW is loop free.* The kernel routing table in CLUSTERPOW is a composite of the individual routing tables at different power levels. It is possible that this interaction between routing protocols could lead to packets getting into infinite loops. However this is not the case, as we prove in the theorem below.

Theorem 1: The CLUSTERPOW power control protocol provides loop free routes.

Proof:

The proof is by contradiction. Suppose there is a loop as shown in Fig. 6, i.e., a packet on its way from node S to node D follows the path S-X-Y-X…, That is, it comes to back to node X after traversing it once. We show that this violates

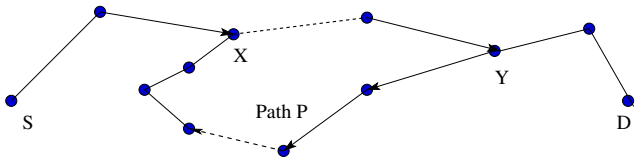


Fig. 6. Suppose there is a loop on the path P from S to D. Dashed lines indicate paths consisting of many hops.

one of the following facts or properties, and hence provides a contradiction.

Property i) The underlying routing protocols at every fixed power level are loop free.

Property ii) CLUSTERPOW chooses routes such that subsequent hops use a sequence of non-increasing power levels.

Property iii) Routes do not change if the network conditions do not change. Note that the specification of the routes at any node includes both the next hop as well as the power used to reach the next hop.

There are two cases to consider.

Case i) The path P has all hops of the same power level. This implies that the underlying routing protocol has loops which is a contradiction.

Case ii) If the hops on path P are not of the same power level then they have to be of decreasing power levels. This is ensured by the design of the CLUSTERPOW algorithm (see Section II-B). But if the packet follows the path P as shown and comes back to X then, by Property iii), it has to follow the same path from X to Y which it followed previously. This involves a higher power level hop and violates Property ii), i.e., the hops in CLUSTERPOW use a sequence of non-increasing power levels. ■

C. CLUSTERPOW Implementation and Software Architecture

We now describe the software architecture (see Fig. 7) and the implementation details of CLUSTERPOW in the Linux kernel. The first task is to run multiple routing daemons at different power levels. In Linux, route discovery and maintenance is done by user space programs called routing daemons, and the actual packet forwarding is done by consulting the kernel IP routing table, which is populated by the routing daemons (see [29]). Thus, running multiple routing daemons simply involves starting many of these routing daemons, one for each power level, on pre-assigned ports. They use UDP packets for communication, thus transport layer port demultiplexing ensures that a routing daemon at a particular power level communicates only with its peers at other nodes. From the routing tables at all the power levels, the composition of the kernel routing table is done by the CLUSTERPOW agent running in user-space. The routing daemons themselves do not modify the kernel routing table directly.

The next task in the implementation is to make the Linux kernel aware of the concept of transmit power of a packet. The most natural way to do this is to add a field (txpower) to the data structure associated with a packet, called skb, which is of type struct skbuff. skb contains various protocol headers

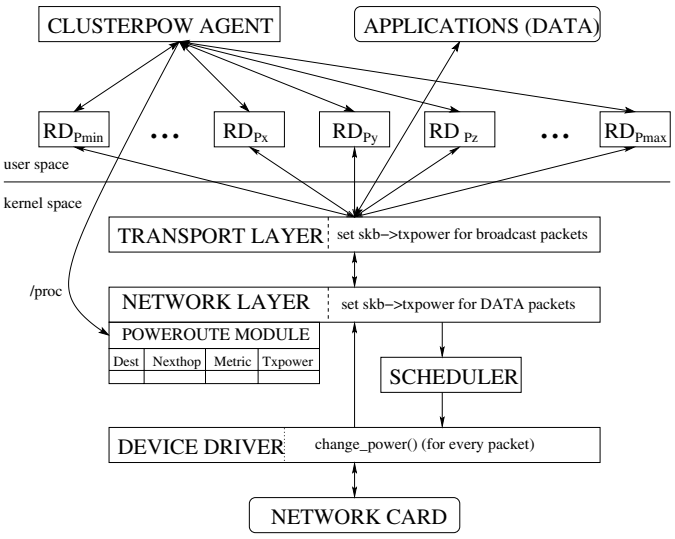


Fig. 7. The software architecture of CLUSTERPOW.

and other layer independent parameters, apart from the data payload. As we will see, skb→txpower is set by the network layer and used by the device driver to set the power on the card before transmitting the packet on the air.

Now we consider the issue of extending the kernel routing table by adding an extra field skb→txpower, which specifies the power level to be used when forwarding packets for that destination. A possible approach is to modify the core Linux IP forwarding code to add this field to the kernel routing table and to modify the forwarding functionality accordingly. Not only is this task tedious, but it also has the disadvantage of requiring extensive changes in the kernel core, hence making it unacceptable for possible inclusion in the standard distribution. This modification would also necessitate a change in the kernel API and would break programs like "route" which rely on this API.

The approach we take is to implement the addendum to the kernel routing table (i.e., the txpower field) in a kernel module called *poweroute*. This modules uses Netfilter, a generic packet filtering and mangling framework in the Linux 2.4 kernel, to intercept packets after they have consulted the kernel routing table and sets the skb→txpower field in accordance with the additional table, which is administered by user-space programs using the /proc interface.

The transmit power of broadcast packets (e.g., hello packets from the routing daemon) cannot be decided by the routing table, since different broadcast packets may need to be sent at different power levels. Hence the transmit power for such packets has to be specified by the application sending these packets. For hello packets this application will be the routing daemons running at different power levels. We have provided such a mechanism by modifying the sendto() system call, so that the transmit power can be specified by using certain values for the flags argument of this call.

The network device driver was modified so that it can read the transmit power from the skb and set it on the card. The driver also maintains a new variable called DefaultTxPower, which is used to transmit packets for which no power level has been specified from above. This is used by the COMPOW agent to specify the default node power level, whereas for CLUSTERPOW it is set to the max power level. We have used the Cisco Aironet 350 cards in our implementation, which are the only commercially off-the shelf available cards supporting multiple transmit power levels.

Finally, the scheduler (see Sec. II-A) to reduce power switch-over latencies has been implemented in the generic device queues below the IP layer.

The CLUSTERPOW protocol has been implemented in the 2.4.18 Linux kernel and its correct functioning has been tested on our ad hoc networking testbed. The implementation and architecture we have provided can also be used to implement other power control schemes. Source-code is available on line at http://www.uiuc.edu/∼ kawadia/txpower.html.

D. Some Comments on Hardware

The overhead of CLUSTERPOW is smaller when there are only a small number of discrete transmit power levels. This is true of the current off-the-shelf wireless network interface cards capable of transmit power control. For example, the Cisco Aironet 350 series cards (IEEE 802.11b compliant) allow the transmit power level to be set to one of 1, 5, 20, 30, 50 and 100 mW, while the Cisco Aironet 1200 series cards (IEEE 802.11a compliant) allow the power level to be set to one of 5, 10 and 20 mW. These cards operate in different frequency bands, and are the only currently available off-the-shelf cards which allow the transmit power level to be changed. In the event of more vendors providing different cards having different transmit ranges and power levels, there needs to be a calibration equivalence of power levels between vendors, to enable the use of diverse hardware in a network. Standardization is required for interoperability.

We assume in our design that the hardware is capable of per-packet power control. The above Cisco cards appear to comply with this assumption only partially, as there is an inexplicably large power change latency. The latency when measured in the driver was found to be 6ms, but even after the power level has been changed on the card, it takes some time to resume transmission at full throttle. When we estimated the latency of a power change at the network layer by monitoring *ping* traffic on the network, it was close to 100ms. However, power in cellular CDMA networks is adjusted 800 times a second, and current electronics is capable of much more frequent power changes. But the firmware in the Cisco cards, unfortunately, appears to be so written that it requires a reset for every power level change. To reduce this wasteful switch-over latency, we use a *scheduling policy* (see [30]) of serving all the queued packets of current power level before changing the power level.

III. Recursive Lookup Schemes

In this section we explore improvements to the CLUSTERPOW protocol using schemes involving recursive lookup of

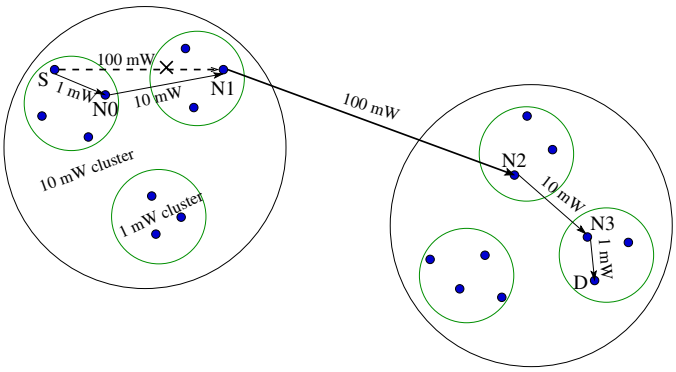


Fig. 8. Modifying the CLUSTERPOW protocol, so that the 100 mW hop from S to N1 can be replaced by two hops of 1 mW and 10 mW each.

routing tables. This leads to the development of the Tunnelled CLUSTERPOW protocol.

A. Recursive Lookup of Routing Tables

It was noted in Section I and in [1], [2], that numerous low power hops are preferable to fewer high power hops for optimizing network capacity. In light of this, it is advantageous to replace the first 100 mW hop in Fig. 3 by two shorter hops of 1 mW and 10 mW respectively, as shown in Fig. 8. It seems possible to achieve this by a more sophisticated composition of the routing tables at various power levels to form the kernel routing table.

The scheme we consider is to recursively lookup the next hop in lower power level routing tables, until we get to the lowest power level routing table at which the next hop is reachable. Thus in Fig. 8, the next hop N1 at node S is looked up in lower power routing tables to find that it is reachable at 10 mW through N0, which in turn is reachable at 1 mW. So ultimately the packet is given to N0 at 1 mW. This same algorithm is carried out at N0 when the packet gets there, and at each subsequent node on the path. Thus we seem to have achieved a finer optimization by recursive lookup of individual routing tables at different power levels to compose the kernel routing table. However, the recursive lookup scheme presented above can lead to packets getting into infinite loops.

B. Counterexample

The system in Fig. 9 provides a counterexample. Node S needs to send a packet to node D. It figures out that the next hop is the node N10 in the 10 mW routing table. Recursive lookup for N10 reveals that it is reachable at 1 mW, and the next hop is N1. Thus S forwards the packet to N1 at 1 mW. After the packet reaches N1, it runs the same algorithm. It finds that the lowest power level at which D is reachable is 10 mW and the next hop is S. S itself is reachable at 1 mW, so the packet is handed over back to node S, and we have an infinite loop.

Note that this loop is not due to the counting to infinity problem of distance vector protocols, but is a consequence of the recursive lookup algorithm.

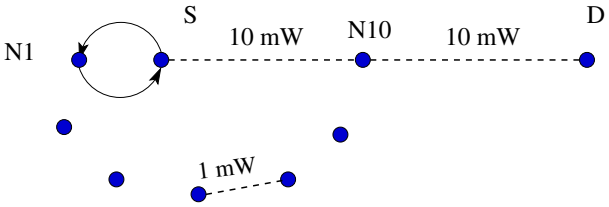


Fig. 9. The recursive lookup scheme is not free of infinite loops.

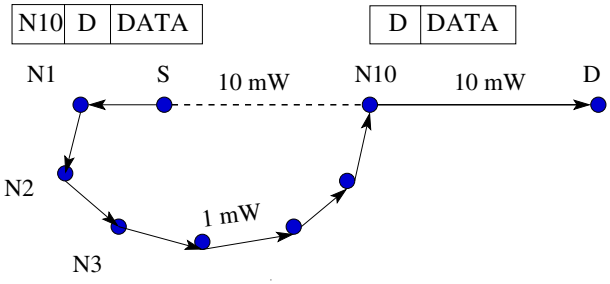


Fig. 10. Tunnelled CLUSTERPOW protocol resolves the infinite routing loop of the network in Fig. 9. The headers added to the packet, as it travels along the route, are also shown.

C. The Tunnelled CLUSTERPOW Protocol

The recursive lookup scheme described above can be modified so that it is indeed free of infinite loops. This is done by tunnelling the packet to its next hop using lower power levels, instead of sending the packet directly. One mechanism to achieve this is by using IP in IP encapsulation. Thus, while doing a recursive lookup for the next hop, we also recursively encapsulate the packet with the address of the node for which the recursive lookup is being done. The decapsulation is also done recursively when the packet reaches the corresponding next hop. We call this the Tunnelled CLUSTERPOW protocol.

As shown in Fig. 10, Tunnelled CLUSTERPOW does resolve the loop in our example of Fig. 9. Now when node S forwards the packet to N1, it encapsulates the packet with the address of N10. Thus N1 does a routing lookup, not for the destination D, but for node N10. It finds that N10 is reachable at 1 mW through the path N2, N3 ..., and it forwards the packet to N2 at 1 mW. When the packet gets to N10 it decapsulates the packet, and then sends it to D at 10 mW. Thus, the packet does reach its destination in this example. We now prove that the Tunnelled CLUSTERPOW power control protocol always ensures that packets reach their destinations.

Theorem 2: The Tunnelled CLUSTERPOW power control protocol ensures that packet reach their destinations.

Proof: The proof is by induction on the number of transmit power levels. As in the proof for CLUSTERPOW, we assume that the underlying routing protocols are loop free at each fixed power level.

Suppose there are t transmit power levels indexed from 1 through t, ordered such that power level t is the lowest. We provide a proof by induction on the number of transmit power levels t.

The base case for $t = 1$ is obvious, since that reduces

to a single routing daemon for a fixed power level, and the underlying routing protocol is assumed loop free.

Assume that the Tunnelled CLUSTERPOW protocol provides routes free of infinite loops when t power levels are in use. This is the induction hypothesis. Now we add the $t + 1^{th}$ power level, which is lower than any power level already in use. Here we note that Tunnelled CLUSTERPOW is a refinement to CLUSTERPOW, as seen in Figure 8. If a packet from source S to destination D visits the sequence of nodes $\{a_i\}$ in CLUSTERPOW, and the sequence of nodes $\{b_i\}$ in Tunnelled CLUSTERPOW, then $\{a_i\}$ is a subsequence of $\{b_i\}$. This is ensured by the encapsulation or the tunnelling mechanism. Thus, if a packet in Tunnelled CLUSTERPOW can get from a node a_j to node a_{j+1}, for any j, then it will indeed get to the destination by Theorem 1, which states that CLUSTERPOW is loop free.

Therefore, consider the sub-problem of getting from node a_j to node a_{j+1}, for any j. Suppose, CLUSTERPOW was using a power level p in getting from node a_j to node a_{j+1}. Tunnelled CLUSTERPOW will introduce more hops between a_j and a_{j+1}, only if they use a power level strictly less than p. This sub-problem thus reduces to running the Tunnelled CLUSTERPOW protocol with t power levels, which is free of infinite loops by the induction hypothesis. ■

D. Architecture and Implementation Issues

The software architecture for Tunnelled CLUSTERPOW is similar to that for CLUSTERPOW. However, the implementation itself is more complicated because of the recursive encapsulation and decapsulation involved. We need a dynamic per-packet tunneling mechanism, which is not available in the Linux kernel and is quite complicated to implement. Forwarding overhead also goes up due to the increase in the IP header, and the increased processing required for the encapsulation and decapsulation. Because of these issues, we have not done an implementation of this protocol. Nevertheless, it provides an interesting concrete example of the sort of schemes that are possible with a sophisticated composition of various individual routing tables built at different power levels.

IV. The MINPOW Routing and Power Control Protocol

The schemes presented so far focus on maximizing network capacity. We would also like to minimize the energy consumption, but given the current hardware, the two goals are not achievable simultaneously. This is because, the power consumption in processing while transmitting and receiving is typically higher than the radiative power required to actually transmit the packet (see Section VII-D). Thus, we provide another protocol, called MINPOW, which minimizes the total power consumption (for communication) on a route. It is essentially the distributed Bellman-Ford algorithm with sequence numbers, and with total power consumption as the cost instead of the hop count metric normally used. Any shortest path algorithm can be used. The basic idea behind MINPOW is not new, and has been suggested before in different forms in [6], [7], [8], [9]. Various metrics like signal strength, transmit power cost of the link, a node's remaining battery life, or variance in battery life among all nodes, have been proposed. These approaches generally require substantial physical layer support, and the lack of standardization for cross-layer interaction appears to have prevented an implementation of such schemes in a real testbed.

We provide a solution which implements MINPOW completely at the network layer using only hello packets, without requiring any support from the physical layer for estimating per link power cost. Our method works for both proactive as well as reactive routing protocols. Our contribution thus involves a generic method to estimate the link cost, and an architecturally clean implementation of the MINPOW protocol.

It should be noted that MINPOW optimizes the energy consumed for communication, i.e., for the transmission and reception of packets, and the associated processing. It does not account for the energy consumed when the nodes are idle, i.e., not communicating. If the idle power consumption is high, then an effective power saving strategy would be to put nodes to sleep. Sleeping is however a difficult problem; we elaborate on it further in Section VII-D. Some strategies for sleeping have been suggested in SPAN [13], and in GAF [12]. Here, we focus on optimizing the power consumption for communication, and our scheme can be used in conjunction with a strategy for sleeping.

The link cost, i.e., the power consumption for communication, has three components (as elaborated in [31]): $P_{Rx_{elec}}$ is the power consumed in the receiver electronics, $P_{Tx_{elec}}$ is the power consumed by the transmitter electronics, and $P_{Tx_{Rad}}(p)$ is the power consumed by the power amplifier to transmit a packet at the power level p, where p is the actual power that is put on the air. $P_{Tx_{elec}}$ and $P_{Rx_{elec}}$ are known locally to the transmitter and receiver respectively, according to their hardware specifications. This enables the protocol to function with heterogeneous hardware as well. The third component $P_{Tx_{Rad}}(p)$ can be calculated if the smallest transmit power p required to traverse the link can be estimated.

One possible scheme to estimate the smallest transmit power required to traverse the link is as follows: the transmit power can be calculated by measuring the distance between the two nodes on the link and using a decay model for the path loss. One of the common models assumes that path loss in the medium follows an inverse α-th law with $\alpha \geq 2$, i.e., the received power at a distance ρ from a transmitter using a power level P_{trans} is $\frac{cP_{\text{trans}}}{\rho^{\alpha}}$, where c is a constant. Suppose that in order to receive a packet the received power level must be at least γ, i.e., $\frac{cP_{\text{trans}}}{\rho^{\alpha}} \geq \gamma$. Then the needed transmitter power level is at least $\frac{\gamma\rho^{\alpha}}{c}$. Thus, P_{trans} can be estimated, given the received power and the distance from the transmitter.

However, there are a few problems with estimating the link cost in this manner. The first difficulty is that distance information is not always available. This would require nodes to be equipped with location equipment like GPS, and advertise their location to other nodes in the network. Secondly, even

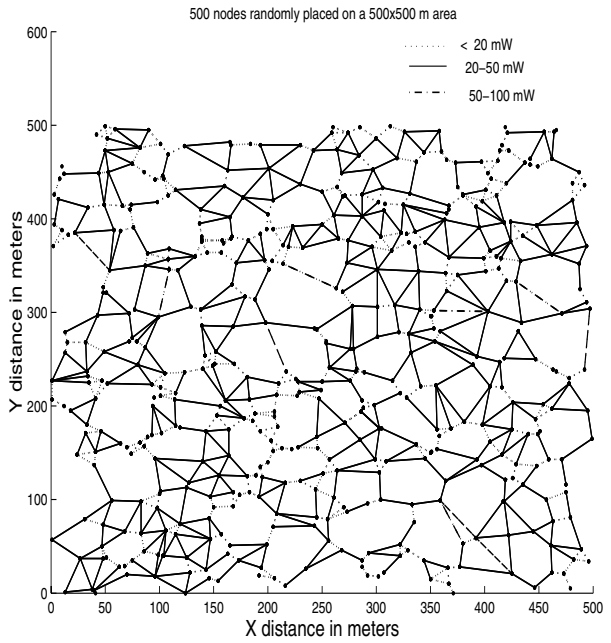


Fig. 11. The graph of optimal power routes with the required transmit power for the link as the link cost.

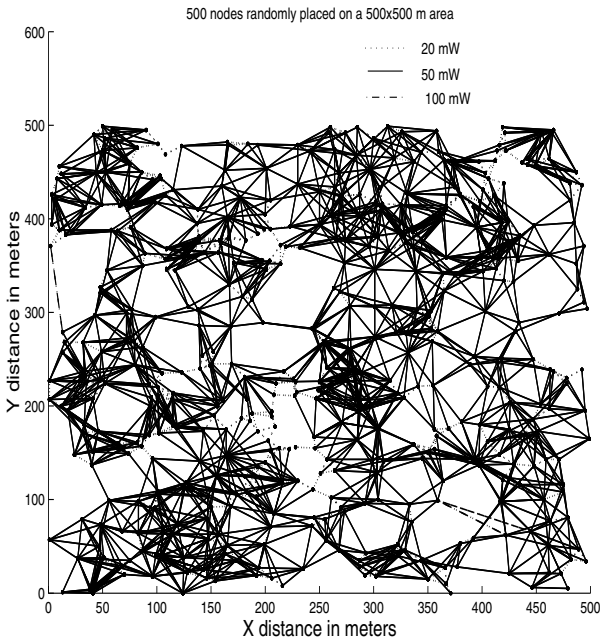


Fig. 12. The graph of optimal power routes is no longer planar if the link costs are discrete.

if distance information is available, it should be noted that relying on the geographical co-ordinates can lead to errors in calculating the transmit power cost for the link, because they do not take into account obstacles in the environment and shadowing in the channel. Third, the scheme requires accurate measurement of the per-packet received signal strength. This support is however not available on all hardware, and, even when it is, reliable measurements are difficult because of channel fluctuations. Another problem in using this methodology for estimating cost has to do with the accuracy of the path loss model. The parameter α is strongly environment dependent, and can vary significantly (see [32]). Thus, the above approach to estimate p is unsuitable.

We should also consider the fact that typically there are only a few discrete power level setting available, e.g., the Cisco Aironet 350 cards have only six distinct transmit power levels. When the transmit power level can be set to any value from a continuum of power levels and is chosen as the link cost, the graph formed of all edges lying along some power optimal route is planar for $\alpha \geq 2$, where α is the path loss exponent. This result has been proved in [2]. Such a graph for $\alpha = 2$ is illustrated in Fig. 11, which was generated by a simulation involving 500 nodes placed randomly on a 500x500m area. When the same simulation was repeated with the constraint that the power level can be chosen only from the discrete set of three power levels, then the graph in Fig. 12 was obtained. This graph has properties different from the one in Fig. 11; for example it is not a planar graph. Thus, we should use a discretized link cost, rounded above to the nearest transmission power level that the hardware is capable of, rather than using the exact value of the transmit power required for successful transmission.

Taking all these issues into consideration, we now describe our scheme.

A. MINPOW Implementation

We have modified the DSDV implementation in [33] to implement MINPOW. To estimate the link cost, every node pro-actively sends hello packets at each of the transmit power levels available, all of them containing the same sequence number. Only the hello packets at the maximum power level contain the routing updates. The rest are only "beacons" which contain the address of the originator, the total power consumed, $P_{Tx_{total}}$, in transmitting that packet, the transmit power level p used for transmitting the packet, and the sequence number of the corresponding maximum power level hello packet. Note that $P_{Tx_{total}} = P_{Tx_{elec}} + P_{Tx_{Rad}}(p)$ where p is the transmit power level of the current beacon packet. The neighbors receiving these beacons set the link cost to be the minimum $P_{Tx_{total}}$ value among the beacons that they have successfully received, plus the energy they spent in receiving:

$$linkcost = \min_{beacons} (P_{Tx_{total}}) + P_{Rx_{elec}} \quad (1)$$

This link cost is then used in the distance vector algorithm for computing the routes. The corresponding transmit power p is used for sending packets to the next hop. Note that these beacons are sent pro-actively at regular intervals, thus the link cost is continuously updated to adapt to mobility and changes in the network topology. The software architecture for this MINPOW implementation is illustrated in Fig. 13.

The method suggested above works for both proactive as well as reactive routing protocols. Most reactive routing

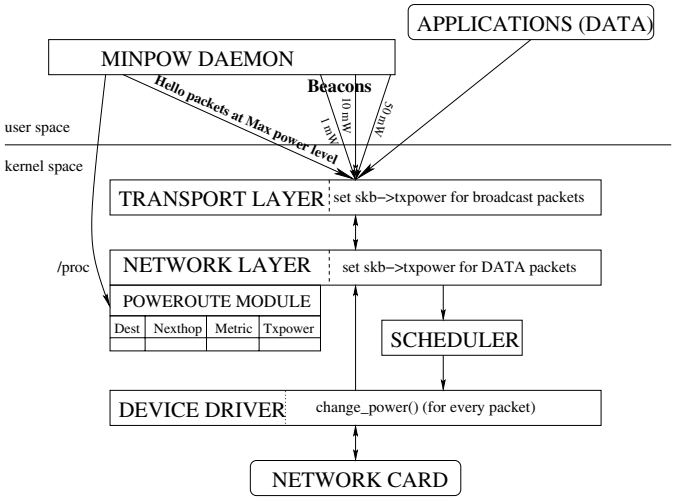
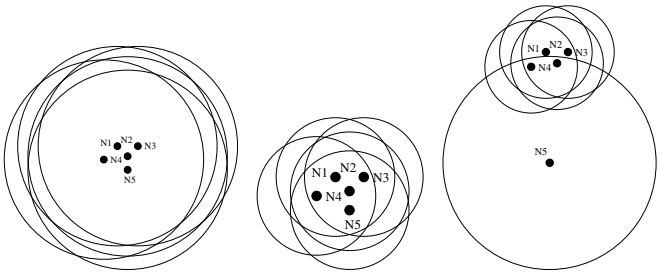


Fig. 13. The software architecture of MINPOW.



(a) Co-located nodes at 100 mW. (b) 1 mW is enough. (c) A clustered network.

Fig. 14. Some topologies for experimentation.

protocols, e.g., AODV [28], use beacons for sensing link status, i.e., to check if a neighbor has moved away. These beacons can be sent at all available power levels in turn, and can be used to estimate the link cost as described above. The route requests themselves are sent at maximum power, but the nodes use the link cost as calculated above.

Our implementation does not need any measurement support from the physical layer. However, we do need the extra txpower field in the kernel routing table. Also needed is per-packet power change support from the network driver. The architecture used for CLUSTERPOW already provides these facilities, making MINPOW readily implementable.

B. Properties of MINPOW

To summarize, the MINPOW protocol has the following properties:

1) It provides a globally optimal solution with respect to total power consumed in communication. This follows from the optimality of the distributed Bellman-Ford algorithm. However this may not be the optimal solution for network capacity. In general, the two objectives cannot be simultaneously satisfied.
2) MINPOW provides loop free routes. This is true because the distributed Bellman-Ford algorithm with sequence numbers is loop free provided the link cost is non-negative (see [27], which is true in our case.
3) No measurement support is needed from the physical layer. Neither is information needed regarding node locations. The cost estimation for the power level for a link is done through hello packets only at the network layer. We only need a one-time characterization of the power consumed by the electronics during reception and transmission.
4) The suggested architecture works for both proactive (table-driven), as well as reactive (on-demand) routing protocols.
5) The protocol, as designed, particularly the link cost estimation technique, works with diverse hardware as well.

V. Experimentation

The correctness of our MINPOW and CLUSTERPOW implementations was tested in some scenarios on our ad hoc networking testbed. In one of the tests, we started with 5 nodes co-located on a desk, using 100 mW by default, as shown in Figure 14(a). When CLUSTERPOW was allowed to run, the kernel routing tables at all the 5 nodes were built. The txpower field for all the entries was 1 mW and as expected, all the nodes were using 1 mW, as shown in Figure 14(b). The same result was obtained for MINPOW as well. Next, one of the nodes, N5, was moved away from the others so that it could be reached only at 100 mW, as shown in Figure 14(c). The routing table entries for this outlying node N5, at nodes N1-N4, were then automatically modified by the protocol to use a power level of 100 mW, while N5 had 100 mW in its routing table for all the other nodes. The nodes of the 1 mW cluster used 1 mW for intra cluster communication. MINPOW resulted in the same result for this particular scenario.

We now elaborate on some problems that we faced during our efforts at more extensive experimentation.

1) Even though the Cisco Aironet 350 cards that we are using support multiple power levels, they do not appear to be designed for per-packet power switching. As we noted in Section II-D, the firmware automatically forces a reset when the power level is changed. Apart from the latency, frequent power changes caused these cards to crash often during our experimentation. Thus, any experimentation with a significant amount of traffic was rendered impossible.
2) Formation of effective multi-hop topologies proved to be difficult, due to a subtlety in the carrier sensing strategy used in the IEEE 802.11 MAC protocol. The interference range in these cards is approximately twice that of the communication range. That means that if any transmission within a radius r can be received successfully, then the carrier can be sensed for any ongoing transmission in

a radius $2r$. This issue is considered in [34] to suggest a MAC protocol using power control. Since an IEEE 802.11 transmitter does not transmit when it senses the carrier, the expected capacity improvements of using low power levels cannot be ascertained in a small testbed consisting of a few tens of nodes, with a networks radius of 3-4 hops, as the carrier sensing mechanism silences most of the nodes in the network.

VI. Concluding Remarks

We have presented solutions to the problems of power control and clustering in non-homogeneous networks. Our approach provides an implicit and dynamic clustering of the network using transmit power. Unlike many other approaches, there are no cluster-head or gateway nodes. The clustered structure of the network is automatically manifested in the way routing is done.

The protocol details of CLUSTERPOW, Tunnelled CLUSTERPOW, and MINPOW are presented along with the software architecture and the implementation details in the Linux kernel. CLUSTERPOW strives to increase network capacity, whereas MINPOW provides a globally optimal routing solution with respect to total power consumed in communication. MINPOW has been implemented at the network layer using hello packets only, without any support from the physical layer. The architecture works for any routing protocol.

VII. Directions for Future Work

A. The MAC Problem

It should be noted that the four-phase handshake of the IEEE 802.11 MAC protocol [35] works smoothly only when a common power level is used throughout the network. This is because a CTS packet sent at a lower power level may not silence some nodes, even though they are capable of interfering with the ongoing transmission when using a higher power level. Thus, any power control or clustering scheme using multiple power levels at the same time has to pay some throughput penalty due to the MAC interference caused when IEEE 802.11 MAC is used. It is possible to design MAC schemes using an extra signaling or reservation channel which can alleviate this problem, see e.g., Yeh and Zhou [36].

However, IEEE 802.11 is the most common MAC protocol, especially on current off-the-shelf equipment. So a high power level should be used sparingly, and most of the intra-cluster communication should use a low common power level. Long distance communication is expensive, either because it silences too many nodes, or because it disrupts ongoing traffic. The solutions we have suggested comply with the above guidelines. In fact, the COMPOW protocol, proposed in [2], is probably the only power control protocol that does not hamper IEEE 802.11. However, it works well only for networks with homogeneous spatial distribution of nodes, and requires proactive routing protocols.

B. QoS Issues: The Latency Problem

We have shown that reducing the power level is optimal with regard to network capacity. However, increasing the number of hops increases the average end-to-end latency linearly with the number of hops, at least when the system is not heavily loaded. Under heavy load, increased MAC contention may induce a latency overhead which may offset the latency gains of using a high power level.

The CLUSTERPOW architecture provides an elegant architecture for implementing Quality of Service a la Diffserv, where latency can be traded for capacity or energy. Multiple routes using different power levels are always available in this architecture, and a QoS policy can be implemented to utilize all this information. Integrating QoS facilities with the CLUSTERPOW architecture is part of the work we plan to do in the future.

C. Load Adaptive Power Control

Using a high power level causes interference in a larger region, but only if nodes in that region have data to send at that time and are thus contending for the channel. A high power level can thus be used to reduce end-to-end latency or possibly save battery power without hurting capacity, if nodes in the neighborhood do not have much data to send. The power control scheme should thus adapt to the network load, i.e., the amount of traffic that nodes have to send. However, if the traffic is asynchronous and bursty, then load prediction may be difficult. Note that a local measurement of load at the MAC layer is not useful because the relaying burden at a node could suddenly increase due to burstiness of a node far away, which uses this node for forwarding. We intend to investigate metrics that can be propagated along with the routing control messages, to help estimate network load.

D. Sleeping

For current off-the-shelf hardware, the power consumption in the transceiver electronics for transmitting, receiving or even remaining idle, but awake, is almost an order of magnitude higher than the power consumed when sleeping, i.e, turning the radio off (source: Cisco data sheet [37], and measurements in [38]). Thus, given current hardware, the only way to save energy may be to put nodes to sleep. However the decision to sleep cannot be relegated entirely to the MAC layer, because the routing layer may critically depend on the availability of the node for forwarding packets. Thus the decision to sleep impacts the network layer. The neighbors of a node need to be informed, so that they can use alternate routes while the node is asleep, and also buffer packets destined for the sleeping node. The CLUSTERPOW architecture already provides alternate routes of higher power levels which can be used when a node is sleeping. However, putting nodes to sleep randomly may be extremely detrimental to network capacity. We plan to investigate distributed strategies for sleeping, keeping these trade-offs in mind.

ACKNOWLEDGMENTS

CLUSTERPOW implementation reuses components from the COMPOW implementation which was jointly done with Swetha Narayanaswamy; we thank her for her help. We also thank Girish Baliga for pointing out an incoherence in the proof for Theorem 2, prompting us to present a clearer proof. Finally, thanks to Binita Gupta for her implementation of the DSDV protocol.

REFERENCES

[1] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. IT-46, pp. 388–404, 2000.

[2] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar, "Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the COMPOW protocol," in *European Wireless Conference*, 2002.

[3] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *Proceedings of INFOCOM*, 2000, pp. 404–413.

[4] T. A. ElBatt, S. V. Krishnamurthy, D. Connors, and S. Dao, "Power management for throughput enhancement in wireless ad-hoc networks," in *IEEE International Conference on Communications*, 2000, pp. 1506–1513.

[5] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, "Distributed topology control for power efficient operation in multihop wireless ad hoc networks," in *Proceedings of INFOCOM*, 2001, pp. 1388–1397.

[6] S. Singh, M. Woo, and C. S. Raghavendra, "Power aware routing in mobile ad hoc networks," in *Proceedings of ACM MOBICOM*, 1998, pp. 181–190.

[7] M. W. Subbarao, "Dynamic power-conscious routing for manets: An initial approach," in *IEEE Vehicular Technology Conference*, 1999, pp. 1232–1237.

[8] Q. Li, J. Aslam, and D. Rus, "Online power-aware routing in wireless ad-hoc networks," in *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking*, July 2001, pp. 97–107.

[9] R. Dube, C. D. Rais, K.-Y. Wang, and S. K. Tripathi, "Signal stability based adaptive routing (SSA) for ad-hoc mobile networks," in *IEEE Personal Communications*, 1997.

[10] J. P. Monks, V. Bhargavan, and W.-M. Hwu, "A power controlled multiple access protocol for wireless packet networks," in *Proceedings of INFOCOM*, 2001, pp. 219–228.

[11] S. Singh and C. S. Raghavendra, "Power efficient MAC protocol for multihop radio networks," in *The Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 1998, pp. 153–157.

[12] Y. Xu, J. S. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of ACM MOBICOM*, 2001, pp. 70–84.

[13] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *Proceedings og ACM MOBICOM*, 2001, pp. 85–96.

[14] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 1265–1275, September 1997.

[15] M. Joa-Ng and I.-T. Lu, "A GPS-based peer-to-peer hierarchical link state routing for mobile ad hoc networks," in *51st IEEE Vehicular Technology Conference*, 2000.

[16] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan, "A cluster-based approach for routing in dynamic networks," in *SIGCOMM Computer Communications Review (CCR)*, 1997.

[17] M. Chatterjee, S. K. Das, and D. Turgut, "WCA: A weighted clustering algorithm for mobile ad hoc networks," *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, vol. 5, no. 2, pp. 193–204, April 2002.

[18] S. Basagni, "Distributed clustering for ad hoc networks," in *International Symposium on Parallel Architectures, Algorithms, and Networks*, 1999.

[19] J. T. Tsai and M. Gerla, "Multicluster, mobile, multimedia radio network," *ACM/Kluwer Journal of Wireless Networks*, vol. 1, no. 3, pp. 255–65, 1995.

[20] T. J. Kwon and M. Gerla, "Clustering with power control," in *IEEE MILCOM*, 99.

[21] Z. J. Haas, "The routing algorithm for the reconfigurable wireless networks," in *Proceedings of IEEE International Conference on Universal Personal Communications (ICUPC'97)*, San Diego, California, Oct. 1997.

[22] A. D. Amis, R. Prakash, D. Huynh, and T. Vuong, "Max-min D-cluster formation in wireless ad hoc networks," in *IEEE INFOCOM*, 2000, pp. 32–41.

[23] R. Krishnan, R. Ramanathan, and M. Steenstrup, "Optimization algorithms for large self-structuring networks," in *INFOCOM: The Conference on Computer Communications, joint conference of the IEEE Computer and Communications Societies*, 1999.

[24] R. Sivakumar, B. Das, and V. Bharghavan, "An improved spine-based infrastructure for routing in ad hoc networks," in *IEEE Symposium on Computers and Communications*, 1998.

[25] U. C. Kozat, G. Kondylis, B. Ryu, and M. K. Marina, "Virtual dynamic backbone for mobile ad hoc networks," in *IEEE International Conference on Communications*, 2001.

[26] P. F. Tsuchiya, "The landmark hierarchy: a new hierarchy for routing in very large networks," in *Symposium proceedings on Communications architectures and protocols*. ACM Press, 1988, pp. 35–42.

[27] C. E. Perkins and P. R. Bhagwat, "Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers," in *Proceedings of ACM SIGCOMM*, 1994, pp. 234–244.

[28] C. E. Perkins, E. M. Royer, and S. Das, "Ad hoc on demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999, pp. 90–100.

[29] V. Kawadia, Y. Zhang, and B. Gupta, "System services for implementing ad hoc routing protocols," in *International Workshop on Ad Hoc Networking*, 2002.

[30] J. R. Perkins and P. R. Kumar, "Stable distributed real-time scheduling of flexible manufacturing/assembly/disassembly systems," *IEEE Transactions on Automatic Control*, vol. 10, pp. 139–148, 1989.

[31] R. Min and A. Chandrakasan, "Energy-efficient communication for ad-hoc wireless sensor networks," in *35th Asilomar Conference on Signals, Systems, and Computers*, vol. 1, 2001, pp. 139–143.

[32] M. Franceschetti, J. Bruck, and L. Schulman, "Microcellular systems, random walks, and wave propagation," in *Proceedings of the IEEE International Symposium on Antennas and Propagation*, June 2002.

[33] B. Gupta, "Design, implementation and testing of routing protocols for mobile ad-hoc networks," Master's thesis, University of Illinois at Urbana-Champaign, 2002.

[34] E.-S. Jung and N. H. Vaidya, "A power control MAC protocol for ad-hoc networks," in *ACM MOBICOM*, 2002.

[35] IEEE 802 LAN/MAN Standards Committee, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," IEEE Standard 802.11, 1999 edition, 1999.

[36] C.-H. Yeh and H. Zhou, "A new class of collision-free mac protocols for ad hoc wireless networks," in *Proceedings of the International Conference Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet*, Jan 2002.

[37] Data sheet: Cisco aironet 350 series client adapters. [Online]. Available: http://www.cisco.com/warp/public/cc/pd/witc/ao350ap/prodlit/a350c"˙ds.h%tm

[38] E. Shih, P. Bahl, and M. Sinclair, "Wake on wireless: An event driven energy saving strategy for battery operated devices," in *Proceedings of ACM MOBICOM*, 2002.