Clustering by Power Control in Ad Hoc Networks

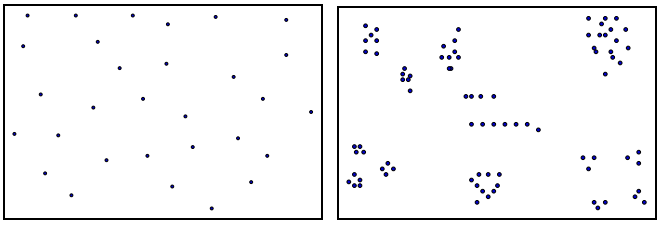
Vikas Kawadia and P. R. Kumar
Department of Electrical and Computer Engineering, and Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign, 1308 West Main St. Urbana, IL-61801.
E-mail:{kawadia,prkumar}@uiuc.edu

***Abstract*—**

In earlier work [1], it was shown that when nodes are uniformly distributed, then, asymptotically as the number of nodes is increased, a common transmit power level is almost optimal with respect to the traffic carrying capacity of the network. In [2] this observation was exploited, and a network layer protocol for power control was developed which ensured convergence to the lowest common power level which ensured connectivity. In this paper, we consider the problem of power control for situations which fall short of the asymptotic regime where nodes may be 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. We address the problem of clustering that results and provide three solutions for joint routing and clustering by power control for ad hoc networks. The first protocol, Clusterpow provides a mechanism for implementing QoS a la Diffserv where power is traded for latency. The second, Tunnelled Clusterpow, allows a finer optimization by using encapsulation. The last, MINPOW, whose basic idea is not new, provides an optimal routing solution with respect to transmit power, but does not readily allow tuning of packet latencies. Our contribution includes a clean implementation of MINPOW at the network layer without any physical layer support. We establish that all three protocols are loop free, while also illustrating how a slightly different approach could lead to packets getting into infinite loops. We provide the software architectural framework of our implementation as a network layer protocol. The architecture works with any routing protocol. Details of the implementation in Linux are also 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 effect the traffic carrying capacity of the network. In [1] it is shown that, generally, 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, if one wants to increase the traffic carrying capacity of the entire network. Additionally, the choice of power level directly affects battery life. Moreover, there is an indirect affect since routing is also affected by the ranges of the transmitters, which depend on the transmit power levels. 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. As noted, power control directly impacts routing and thus cannot be considered in isolation. 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 linearly increases latencies due to the packetization delay at each node.



(a) Homogeneous spatial dispersion of nodes

(b) Nodes non-homogeneously dispersed

Fig. 1. Homogeneous vs clustered networks

Given the 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 was developed which ensured that the

This material is based upon work partially supported by the US-ARO under Contracts DAAD19-00-1-0466 and DAAD 19-01010-465, DARPA under Contracts F33615-01-C-1905 and N00014-01-1-0576, ONR unser Contract N00014-99-1-0696, and AFOSR under Contract Af-DC-5-36128.

transmit power used by all the packets, at 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. Also provided was an implementation in the Linux kernel.

When nodes are homogeneously dispersed in space, as in Fig. 1(a), which is the case asymptotically when a large number of nodes are uniformly distributed, then the choice of a common transmit power level has several appealing features and properties. However, when nodes are non-homogeneously dispersed as in Fig. 1(b), then choosing the lowest common power level for all nodes that results in network connectivity will imply that the common power level is dictated by outlying nodes, those which are far from others, as the node F in Fig. 2. All nodes, except F, are mutually reachable at 1 mW. We say that these nodes form a 1 mW cluster and F is outside this cluster. F can be reached from some of the nodes of the 1 mW cluster but only by using a power level of 100 mW. The COMPOW algorithm, which converges to the lowest power level such that the network is connected, will in this case converge to 100 mW. Thus every node in the network will be forced to use 100 mW even though 1 mW is enough for most communications. Thus, the outlying nodes force all the other nodes to use a higher power level since the protocol works to ensure a common power level at all the nodes.

However, such non-homogeneous scenarios are ripe for clustering. One wishes to group nodes into clusters, with possibly multiple levels of clustering hierarchy, i.e., several clusters at level k form a cluster at level k+1, and use multiple levels of power in a manner commensurate with the multiple layers of clustering. The clustering of nodes cannot be based just on the geographical co-ordinates of a node since the presence of obstacles and shadowing in a wireless channel may prevent two nodes from forming a link at a certain power level, even if they are in close proximity. Power control should also be done in conjunction with routing since they affect each other. Power control cannot be done without keeping connectivity in mind, which is known only through the existence of routes. On the other hand, routing depends on power control since the power level dictates what links are available for routing.

A further factor is that we wish to provide QoS a la Diffserv for the latency of packets, which is not possible in COMPOW. Transmit power affects latency since choosing low power levels forces a packet to travel over many hops of small distance. With packetization delay taken into account, the end-to-end latency grows at least linearly in the number of hops. This QoS allowing a trade-off between latency and power levels also needs to be taken into account architecturally, and should be resolved in a manner compatible with the layering and modular architecture. Before presenting our solutions considering all the above factors, we digress for a brief survey of the literature.

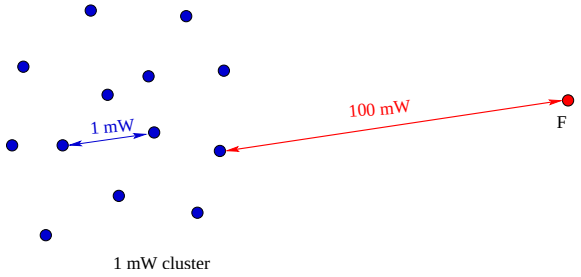


Fig. 2. A common power level is not appropriate for non-homogeneous networks.

A. Related Work

Most of the work on the power control problem, i.e., the problem of choosing the transmit power for every packet in an ad hoc network, 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. Power control is conceptualized as a network layer problem in [2], and the COMPOW protocol is proposed. It is proposed in [3], that each node adjust its transmit power so that its degree (number of one-hop neighbors) is bounded. A distributed topology control algorithm using direction information is proposed in [4]. [5] proposes using transmit power control to optimize the average end-to-end network throughput by controlling its degree. 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. Some other schemes in this class are proposed in [7], [8], and [9]. The third class of approaches aim at modifying the MAC layer. [10] suggests modifying IEEE 802.11's handshaking procedure to allow nodes to transmit at a low power level. [11] proposes enabling nodes to power themselves off when not actively transmitting or receiving.

The clustering problem pertains to classifying nodes hierarchically into equivalence classes according to certain attributes. These attributes could be node addresses [12], geographical regions or zones [13], or a small neighborhood(typically 1 or 2 hop) of certain nodes elected as cluster-heads or leaders, as in [14]. 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 [15], and in DCA [16]. Cluster-heads can be used for routing, for resource allocation among nodes in its cluster [17], and for network management. Cluster-heads are used as base stations to emulate power control as in cellular networks in [18]. Most of the schemes for ad hoc networks take care of cluster maintenance, in addition to cluster formation, to take care of the dynamic network conditions. 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 [19], and in GPS based hierarchical link state routing [13]. Some of the algorithmic aspects of clustering are analyzed in [20] and [21].

The 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 for ease of addressing and management. IP subnetting is a good example of clustering for routing efficiency, as well as ease of management. Routing control message reduction is achieved by backbone formation in Spine based routing [22], and in VDBP [23], 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 [24], or in networks with a natural logical hierarchy (e.g a battlefield). For route aggregation in generic ad hoc networks with mobility, the overhead of an addressing service necessary to inform all other nodes of the dynamically changing addresses of all the nodes in the network, seems to be exorbitant. However, HSR [25] does provide such a location management service.

B. This Work

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. The Clusterpow protocol also provides an architecture for implementing DiffServ type QoS where the power of a packet can be traded off for latency. 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 transmit power 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.

We have assumed a flat addressing space for nodes. A hierarchical addressing scheme, where node addresses are dynamically constructed based on the cluster in which it is present, needs an efficient addressing or location management service which can advertise all addresses to everybody. We are investigating ways to achieve this efficiently in our architecture.

It should be noted that the 4 way handshake of the IEEE 802.11 MAC protocol [26], works smoothly only when a common power level is used throughout the network. This is because a CTS sent at a lower power level may not silence some nodes, which are capable of interfering with the ongoing transmission by 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 802.11 MAC is used. This may not be the case for other MAC protocols which use other means like control channels as in DBTMA [27], or pseudo-random seeds as in SEEDEX [28], for reserving the channel. However, 802.11 is the most common MAC, especially on off-the-shelf equipment, and we should take some care to lessen the problem. 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. As will be seen in the sequel, the solutions we suggest comply with the above guidelines. In fact, the COMPOW protocol proposed in earlier work [2], is probably the only power control protocol that does not hamper 802.11, but it works well for homogeneous networks only, and requires proactive routing protocols.

II. The ClusterPow Power Control Protocol

The Clusterpow power control protocol has been designed for power control and routing in non-homogeneous networks. A single route in Clusterpow can consist of

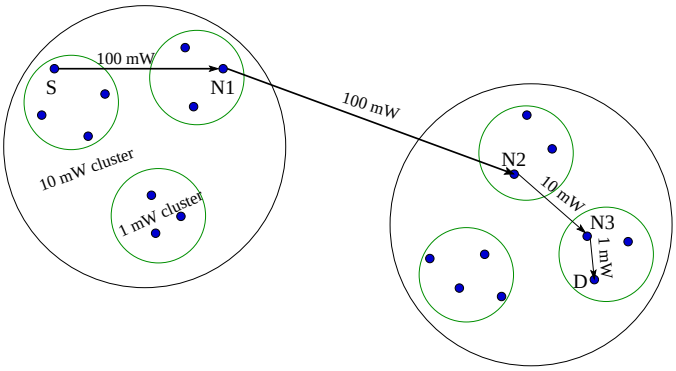


Fig. 3. Routing by Clusterpow in a typical non-homogeneous network.

hops of different transmit power such that the clustered structure of the network is respected. The algorithm is simply to use the lowest transmit power level p, such that the destination is reachable by using a power level 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 by 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 until the packet gets to the 10 mW cluster to which the destination belongs. Then 10 mW is used until the 1 mW cluster to which the destination belongs is reached, and finally a 1 mW hop gets the packet to the destination. Thus, transmit power control leads to automatic clustering in the network.

A. *Clusterpow architecture*

We now describe the architectural design to implement the above algorithm in a simple way, and to integrate it into the network stack as a network layer protocol. The architecture of Clusterpow involves running multiple routing daemons, one corresponding to each power level P_i that is available. These routing daemons build their own separate routing tables RT_{P_i} by communicating with their peers 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. 5. 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 in 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 a transmit power field for every entry which indicates the power to be used when routing packets for that

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

destination.

The user space routing tables at each power level and the kernel IP routing table at each of the nodes corresponding to the network in Fig. 3 is shown in Fig. 4, and the routing procedure is described in detail below. 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 in 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. So the final hop of the packet is at 1 mW. Thus, this architecture provides a simple way to implement the Clusterpow algorithm.

The architectural design of Clusterpow makes certain assumptions. We assume a small number of discrete transmit power levels in our design. This is currently true of the only off-the-shelf wireless network interface cards capable of transmit power control: the Cisco Aironet 350

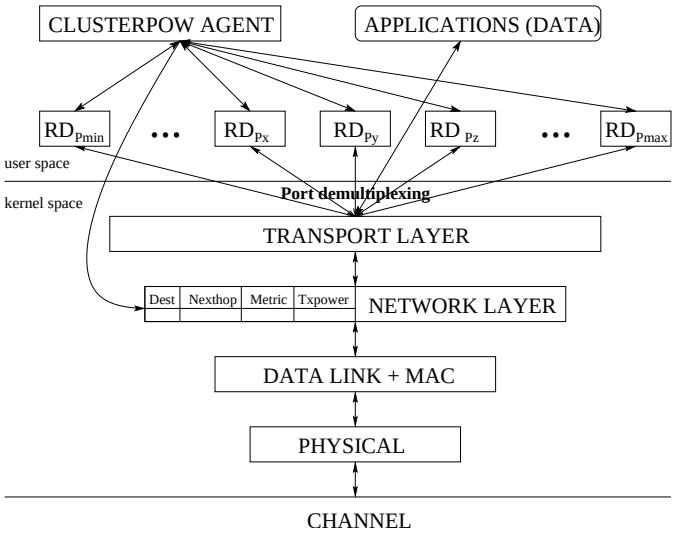


Fig. 5. Architectural design for Clusterpow.

cards, which allow the transmit power level to be set to one of 1, 5, 20, 30, 50 and 100 mW. 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 also assume that the hardware allows per packet power control. Cisco cards 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 control is quite common in CDMA networks and is done 800 times a second. Thus, the current electronics is certainly capable of fast power changes, but the firmware in the Cisco cards, unfortunately, is written so that it requires a reset for every power level change. To reduce this unnecessarily wasteful switch-over latency we use a *scheduling policy*. The policy is to serve all the packets of current power level that are queued before changing the power level. In other words, bunch queued packets together according to power levels before serving them. It can be shown that this reduces the number of power level changes required [29].

B. Clusterpow Properties

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.* This is because 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 least p from the current node to the destination. Thus further 'downstream' there will never be a need of using a higher transmit power.
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.* In the case of a proactive routing protocol (e.g. DSDV [30]), 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 [31], route discovery requests are sent out at all the power levels available. The lowest power level which results in a successful route discovery is used for routing the packet. Some savings are possible for on-demand routing protocols by caching the routes in a user space route cache, and generating route requests for a destination only when it does not have an entry in the cache.
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.

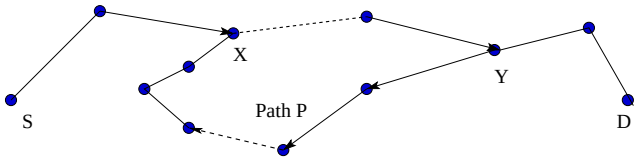


Fig. 6. Suppose, there is a loop on the path P from S to D. Dashed lines indicate possibly multiple hops on the path.

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

Proof: By contradiction:

Suppose there is a loop as shown in Fig. 6, 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 one of the following facts or properties, and hence provides a contradiction.

Property i) The underlying routing protocol is 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, now includes both the next hop as well as the power used to reach the next hop.

Now there are two cases to consider to seek a contradiction.

Case i) The path P has all the hops of same power level. This implies that the underlying routing protocol has loops. This violates Property i) and provides 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. 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. Thus we obtain a contradiction and this completes the proof.

■

C. Clusterpow and QoS

Clusterpow provides a natural architecture for implementing Quality of Service. Multiple routing daemons running in parallel provide lot of information about different routing options which can be exploited in several ways. Alternate routing paths using different power levels can be used to provide QoS. Clusterpow can thus support a DiffServ type of QoS solution. Packets having lower latency requirements can use higher power levels to reduce hop count.

Thus, in the scenario of Fig. 3, higher priority data packets from node S can get to the destination D in only 3 hops of 100 mW each. Normal packets will have to follow the route determined by the protocol which consist of 4 hops. Thus Clusterpow provides an elegant mechanism for cross layer optimization and QoS without disturbing the architectural modularity of the network stack.

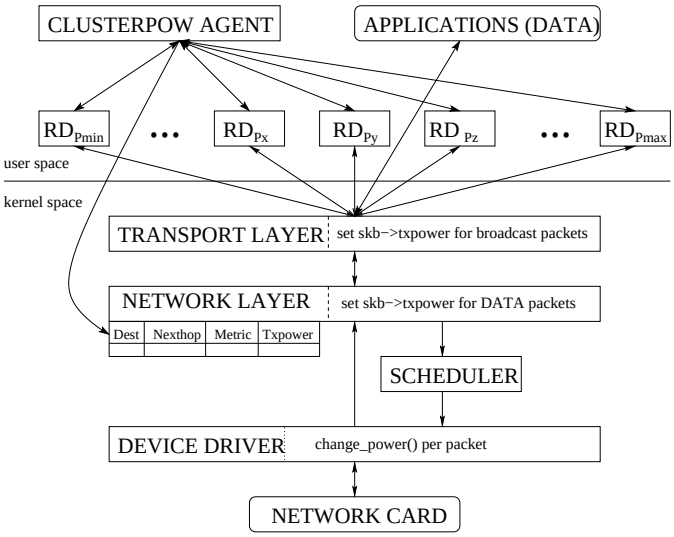


Fig. 7. The Clusterpow Software Architecture.

D. 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. [32] provides more details. 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 to its peers at other nodes. The particular routing protocol we use is DSDV [30], which was also implemented.

Once we have the routing tables at all the power levels, the kernel routing table is constructed by an intelligent composition of these routing tables. The entry for a destination that is copied to the kernel routing table comes from the lowest power routing table in which it appears, i.e., has a finite metric. This composition is done by an agent running in user space, called the Clusterpow agent. It gathers data from all the routing daemons through pipes, and then updates the kernel routing table, the routing daemons themselves do not modify the kernel routing table directly.

In Clusterpow, the transmit power used for a packet depends on the destination of the packet and thus can be decided only when the packet passes through the IP routing module in the network stack. This necessitates the

addition of an extra field txpower, in the kernel routing table. The IP forwarding code of the Linux IP stack was modified so that it puts the transmit power also in the packet, along with other routing information such as the next hop. The IP forwarding code copies the txpower from the kernel routing table to skb→txpower, where skb is the packet data structure in the Linux kernel. This is done for all application or DATA packets. txpower field had to be added to skb because the decision about the transmit power of a packet is made in the network layer, but the transmit power is set only in the network device driver. skb data structure is the most natural mechanism to carry information between the network layer and the device driver. Note that the packets going out on the air do not contain this field.

Broadcast packets (e.g hello packets from DSDV) however, do not consult the routing table. 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 provided such a mechanism by modifying the sendto() system call, so that the transmit power can be specified by adding extra flags. Using extra flags to specify the transmit power preserves the syntax of the system call and does not break existing applications.

The network device driver was modified so that it can read the transmit power from the skb and set it on the card. 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.

Another modification was needed in the routing table administration mechanism so that the transmit power can be specified when adding a routing table entry to the kernel. We modified the SIOCADDRT ioctl so that the transmit power can be specified using extra flags. This mechanism is used by the Clusterpow agent.

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 protocol has been implemented in the 2.4.18 Linux kernel and its correct functioning has been tested on our ad hoc networking testbed.

III. Recursive Lookup Schemes

In this section we explore improvements over the Clusterpow protocol using schemes involving recursive lookup of routing tables, leading to the development of the Tunnelled Clusterpow protocol.

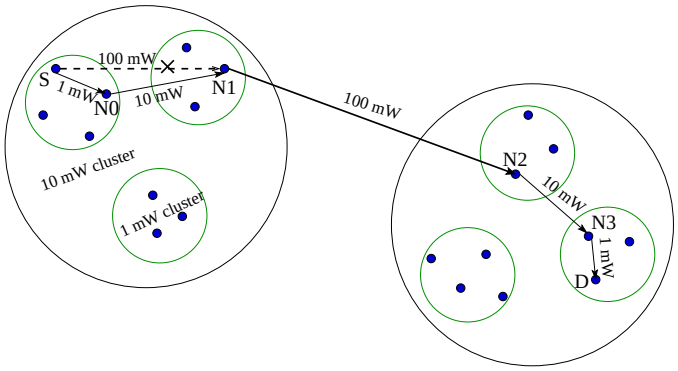


Fig. 8. Improving the Clusterpow protocol. The 100 mW hop from S to N1 can be replaced by two hops of 1 mW and 10 mW each.

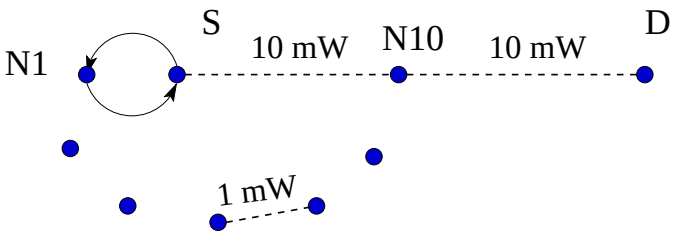


Fig. 9. The recursive look up scheme is not loop free.

A. Recursive look up of routing tables

We have noted before in Sec. 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 the lowest power level is reached 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 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 is not loop free. The counterexample in Fig. 9 demonstrates a routing loop, if the recursive look up scheme is used. S needs to send a packet to 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

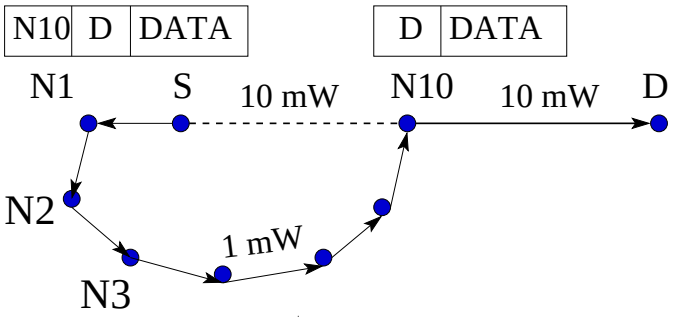


Fig. 10. Tunnelled Clusterpow protocol resolves the routing loop of the network in Fig. 9. The headers added to the packet, as it travels along the route, are also shown.

to N1 at 1 mW. N1 runs the algorithm again and 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 a consequence of the recursive lookup algorithm.

B. The Tunnelled Clusterpow Protocol

The recursive lookup scheme described above can be made loop free if we tunnel the packet to 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 there is no routing loop in this example. We now provide a proof that Tunnelled Clusterpow is indeed loop free.

Theorem 2: The Tunnelled Clusterpow power control protocol provides loop free routes.

Proof: By Induction on the number of transmit power levels.

Assumption: The underlying routing protocol used is loop free.

Let there be t transmit power levels indexed from 1 through t 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, and the underlying routing protocol is loop free.

Now assume that the protocol provides loop free routes when t power levels are in use. This is the induction hypothesis. Now we add the $t + 1$ th power level. Any hops previously using a lower indexed (i.e., higher valued) transmit power level, may now be replaced by multiple hops using the $t+1$ th power level only. Note that any hop previously using a power level m, where $1 \leq m \leq t$, can not be subdivided now to use a hop of power level n, where $m \leq n \leq t$, since if such a subdivision were possible, it would have occurred when power level n was added. Thus, a possible subdivision of any hop, upon the addition of the $t + 1$ th power level, may consist of hops using the $t + 1$ th power level only. This cannot introduce any loops, since the $t + 1$ th power level routing table is loop free. This completes the proof.

■

C. Architecture and Implementation issues

The software architecture of Tunnelled Clusterpow is similar to that of Clusterpow. However the implementation itself is more complicated than Clusterpow because of the recursive encapsulation and decapsulation involved. The tunnelling facility already existing in the Linux operating system cannot be used, since it provides static tunnelling. That is, the endpoints of the tunnel need to be known beforehand, and the tunnel has to be configured before it can be used. What we need is dynamic per packet tunneling with possibly multiple levels of encapsulation. For this dynamic multi-level tunneling, a flag is also needed in the IP header itself to indicate if the packet is encapsulated. Thus, there is a substantial modification of the kernel routing code involved in Tunnelled Clusterpow implementation. The level of encapsulation can be as much as the number of power levels and each encapsulation adds an IP header of 20 bytes. In addition to the routing message overhead, the encapsulation-decapsulation and multiple routing table lookups for every packet is probably lot of processing for a router. Because of these issues, the implementation of this protocol has not been completed yet. Nevertheless it provides an interesting demonstration of schemes that are possible with a sophisticated composition of various individual routing tables built at different power levels.

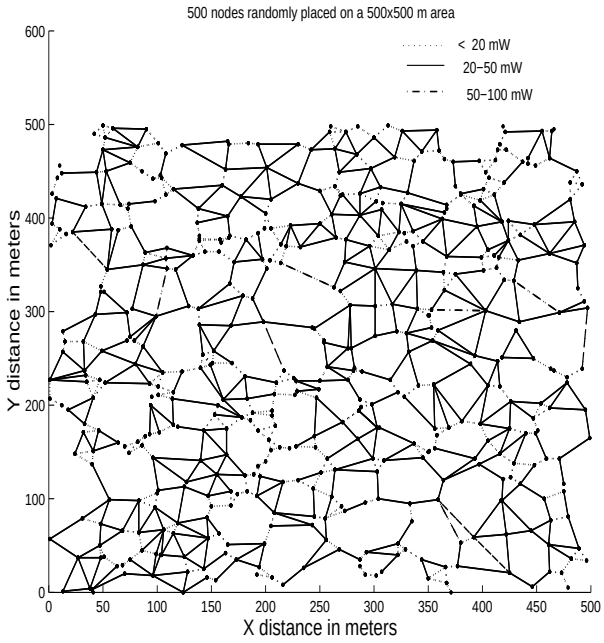


Fig. 11. The graph of optimal power routes with a transmit power required for the link as the cost.

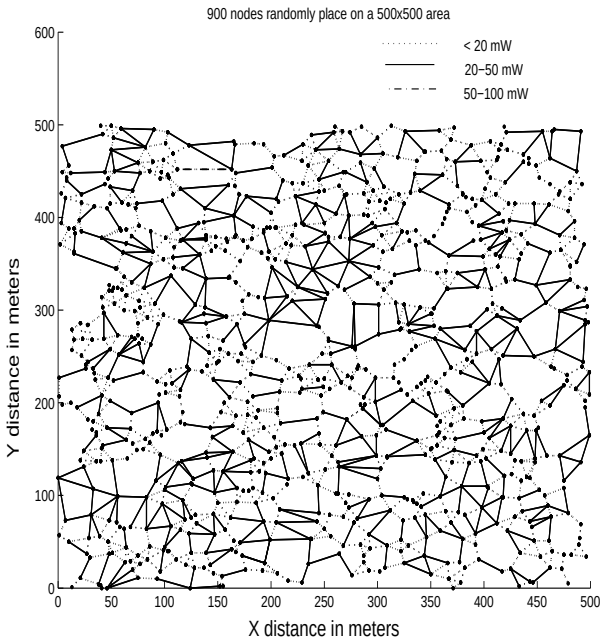


Fig. 12. The graph of optimal routes for 900 nodes. All other simulation parameters were the same as in Fig. 11.

IV. The MINPOW Routing and Power Control Protocol

MINPOW minimizes the total transmit power on a route. It is essentially distributed Bellman-Ford algorithm with sequence numbers, and transmit power as the link cost instead of the hop count 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, 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 has prevented an implementation of such schemes in the real testbed. We provide an implementation of MINPOW at the network layer using only hello packets, and no support from the physical layer for estimating cost. Our method works for both proactive as well as reactive routing protocols.

More thought can be given on estimating or measuring the link cost. The exact transmit power required to traverse the link can be obtained in two ways.

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 $\frac{\gamma\rho^\alpha}{c}$. This can be taken to be the link cost. Thus, if a route from a source to a destination consists of h hops, of distances ρ_i, $i = 1, 2, \ldots, h$, then the power cost of the route is $\frac{\gamma}{c}\sum_{i=1}^{h}\rho_i^\alpha$. We can ignore the scaling $\frac{\gamma}{c}$ and just fix the power cost of the route to be $\sum_{i=1}^{h}\rho_i^\alpha$.

A simulation was performed to discover the optimal routes using this methodology for estimating link cost, with $\alpha = 2$, for 500 nodes on a 500x500 m area. The graph formed of edges which lie along some power optimal path is shown in Fig. 11. The links in the graph have been coded according to the transmit power requirement. This envisages a practical scenario where the hardware is capable of discrete power setting only. This graph is planar and tends to use smaller power hops as the node density increases. Indeed Fig. 12, which is the same simulation as in Fig. 11 but for 900 nodes, shows that most of the hops are now 20 mW hops, the lowest power level available in this simulation. It was proved in [2] that such a graph could always be chosen to be planar for $\alpha \geq 2$ and for $\alpha > 2$ it can be chosen to be a subgraph of the one for $\alpha = 2$.

However, there are a few problems in estimating the link cost in this manner. The first difficulty is that distance information is not always available. For this to be done in a distributed manner, the nodes have to be equipped with location detection equipment like GPS. It should also be noted that just the geographical co-ordinates can be de-

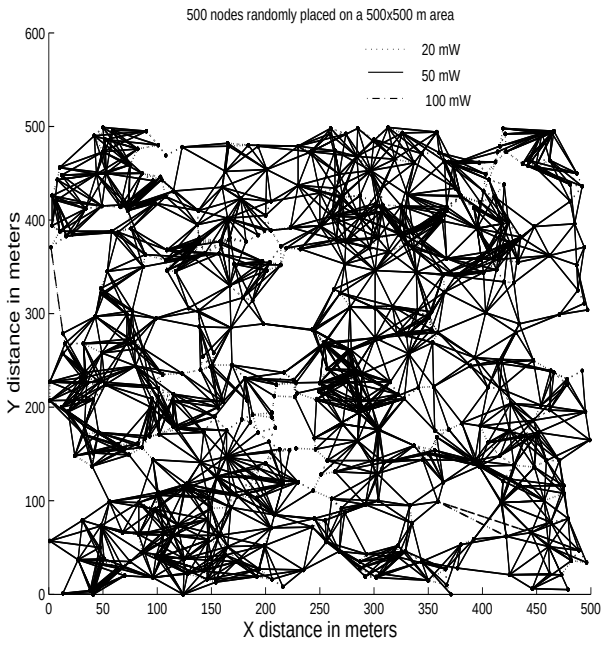


Fig. 13. The graph of optimal power routes is no longer planar if the cost is discrete.

ceptive 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. For example, the presence of a wall between two nodes may increase the transmit power cost of a link, even though geographical information may suggest a very low cost. Another problem 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 typically varies between 2 and 4 when going from an indoor building environment to an open outdoor environment.

A second way to estimate the exact link cost is by direct measurement at the physical layer. Per packet signal strength measurements, if available, can be used to estimate the power required for a successful transmission on that link. However signal strength measurements are hardware dependent. The measurement of the signal strength at the receiver does not give a proper estimate of the power the transmitter might have used for that transmission, since it might be using a different hardware. Another problem with signal strength measurements is their wide fluctuation with the channel state. There are also practical difficulties. There are hardly any drivers which support per packet signal strength measurements and expose that information to user-space programs.

We use neither of the two approaches, the first because the exact propagation model is unknown and distance information is not always available or useful, and the second because the required physical layer support is not available or standardized. We should also use the fact that

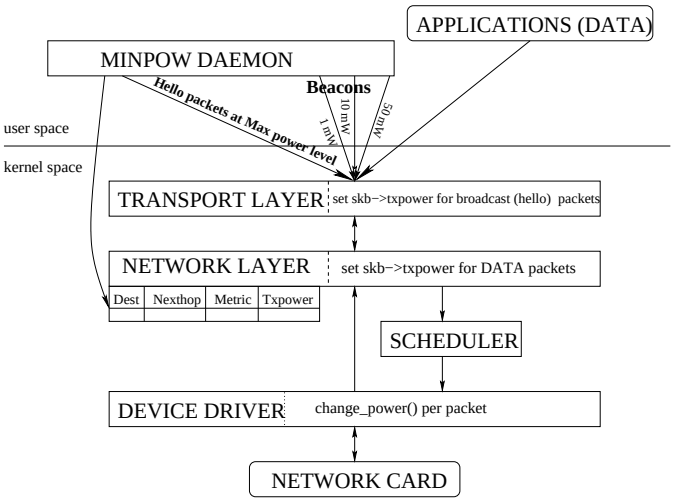


Fig. 14. The MINPOW Software Architecture.

typically there are only a few discrete power level setting available, e.g the Cisco Aironet 350 cards have only 6 distinct transmit power levels. Thus we should use a discretized link cost, rather than using the exact value of the transmit power required for successful transmission. The link cost used should be rounded above to the nearest transmission power level that the hardware is capable of. This is the real optimal solution as this is the power ultimately used for transmission. The simulation of Fig. 11 was repeated with the cost now discretized. The graph of optimal power routes for this case is shown in Fig. 13. This graph is no longer planar but the clustering of nodes is more prominent.

A. MINPOW implementation

We now provide an implementation of MINPOW which uses a discretized transmit power cost as discussed above. We have modified our DSDV implementation to implement MINPOW. To estimate the link cost every node now sends hello packets at each of the transmit power levels available, all of them having 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 power level at which it was transmitted, and the sequence number of the corresponding maximum power level hello packet. The neighbors receiving the hello packets set the link cost to be the transmit power value in the lowest power beacon that they successfully received. This link cost is then used in the distance vector algorithm for computing the routes. The software architecture for this MINPOW implementation is illustrated in Fig. 14.

The method suggested above works for both proactive as well as reactive routing protocols. Most reactive routing protocols, e.g. AODV [31], 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 itself are sent at maximum power, but the nodes use the transmit power as the cost, and also specify the transmit power to be used for the next hop.

Our implementation does not need any measurement support from the physical layer. We do need an extra txpower field in the kernel routing table, and also per packet power change support from the network driver. All these features were also needed for Clusterpow. Thus the MINPOW implementation is primarily in user space, with only the absolutely essential modifications in the kernel and the device driver to support per packet transmit power switching.

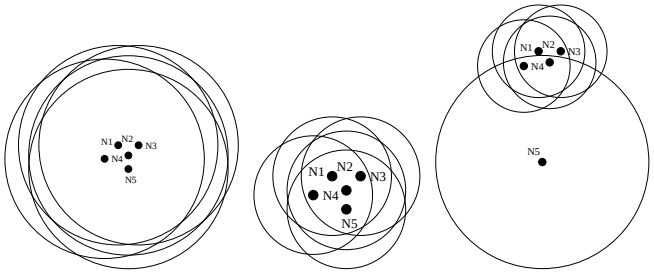
B. MINPOW properties

To summarize, the MINPOW protocol has the following properties:

1) It provides a globally optimal solution with respect to total transmit power.
2) MINPOW provides loop free routes. This is true because distributed Bellman-Ford with sequence numbers is loop free, provided the link cost is non-negative, 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 is done through hello packets only at the network layer.
4) The suggested architecture works well for both proactive (table-driven), as well as reactive (on-demand) routing protocols.
5) QoS support is not possible in MINPOW, as it is in the Clusterpow architecture.

V. Experimentation

The correctness for our MINPOW and Clusterpow implementations was tested in few simple 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 Fig. 15(a). When Clusterpow was allowed to run, the kernel routing tables at all the 5 nodes were built, such that the txpower field for all the entries was 1 mW, and as expected, all the nodes were now using 1 mW , as shown in Fig. 15(b). Same result was obtained for MINPOW as well. Now one of the nodes N5, was moved away from the others so that it could be reached only at 100 mW, as seen in Fig. 15(c). The routing table entry for this outlying node N5, at nodes N1-N4, was now



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

Fig. 15. Experimentation.

automatically modified to use a power level of 100 mW, while N5 now 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 we faced during our efforts for more extensive experimentation.

1) Even though, the Cisco Aironet 350 cards that we are using, support multiple power levels, they are not designed for per packet power switching. As we noted in Sec. II-A, the firmware automatically forces a reset when the power level is changed. Apart from the latency, frequent power changes causes these cards to crash temporarily, every now and then. 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 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 [33] to suggest a MAC protocol using power control. Since an 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; the carrier silences most of the nodes in the network.

VI. Concluding Remarks

We present 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 power control. Unlike most 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 provides a natural architecture for implementing DiffServ type QoS, where power can be traded off for latency. MINPOW provides a globally optimal routing solution with respect to transmit power. 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.

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] 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.

[5] 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.

[6] S. Singh, M. Woo, and C. S. Raghavendra, "Power aware routing in mobile ad hoc networks," in *Proceedings of 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*, Rome, Italy, ACM Press, 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] 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.

[13] 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.

[14] 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.

[15] 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, pp. 193–204, April 2002.

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

[17] 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.

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

[19] Z. Haas, "A new routing protocol for the reconfigurable wireless networks," in *ICUPC*, 97.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable routing strategies for ad hoc wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1369–1379, 1999.

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

[27] J. Deng and Z. Haas, "Dual busy tone multiple access (DBTMA): A new medium access control for packet radio networks," in *IEEE ICUPC*, 1998.

[28] R. Rozovsky and P. R. Kumar, "Seedex: A mac protocol for ad hoc networks," in *Proceedings of The ACM Symposium on Mobile Ad Hoc Networking and Computing, MOBIHOC*, 2001.

[29] 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.

[30] 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.

[31] 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.

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

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