PCAR: A Power Controlled Routing Protocol for Wireless Ad Hoc Networks

Xue Zhang*†, Ming Liu*, Haigang Gong*, Sanglu Lu† and Jie Wu‡
{zhangxue, csmliu, hggong}@uestc.edu.cn, sanglu@nju.edu.cn, jie@temple.edu
*School of Computer Science and Engineering
University of Electronic Science and Technology of China, Chengdu 611731, China
†State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, China
‡Department of Computer and Information Science, Temple University
Philadelphia, PA 19122, USA

***Abstract*—Power control and routing are two fundamental supporting techniques for wireless communications in ad hoc networks. However, most existing power control and routing proposals are not really efficient enough, due to separate considerations on them. In this paper, motivated by the observation on the necessity and feasibility of the combination of power control and routing, we propose a power controlled routing protocol PCAR (Power Controlled Ad hoc Routing). The basic idea is to develop power control on top of the distance-vector routing mechanism that is based on the classical distributed Bellman-Ford algorithm. Each node adjusts the transmission power automatically through the PIPC (Proportion-Integral Power Control) algorithm that we previously proposed. Both theoretical analysis and simulation results show that PCAR has a good performance.**

***Index Terms*—Power control, routing, wireless ad hoc networks**

I. Introduction

Wireless ad hoc networks have drawn more and more research interest over the past decade. However, there are still many issues that need to be addressed for efficient operation of such network systems. Power control and packet routing are perhaps two of the most important issues, since either plays an important role in wireless communications. Solutions for the two issues are intensely under development, such as [1]–[4] on power control, and [5]–[8] on packet routing. However, most previous work does not simultaneously take the two issues into account. This paper aims to develop a power controlled routing protocol based on joint considerations.

Power control is a prototypical cross-layer design problem since it affects all layers of the protocol stack from physical to transport [3]. Because holistic cross-layer design may lead to long term architectural issues, there arises the question of where in the network architecture should power control be located. Previous approaches attempt to solve the power control problem at layers varying from the MAC layer to the network layer. Kawadia and Kumar [3] distilled a basic principle that power control can be regarded as a network layer problem on account of the fact that placing it at a lower layer does not give the routing protocol the opportunity to determine the optimal next hop or the intended receiver.

We further believe that power control should be done in conjunction with routing, which is the central thesis of this paper. Our idea is based on the following observations. First, power control should guarantee network connectivity, but lower-layer protocols do not work efficiently. Second, it is usually more convenient to acquire necessary topological information from routing protocols than from other protocols; otherwise, an extra overhead must be paid for probing network topology. The last and most important reason is that routing not only very much depends on, but also significantly affects power control. They are correlated in too many ways to be considered separately.

Based on those observations, we propose a power controlled routing protocol PCAR (Power Controlled Ad hoc Routing) for wireless ad hoc networks. In [4], we presented PIPC (Proportion-Integral Power Control), a closed-loop power control algorithm that can be used in conjunction with any routing protocol as long as the routing table of a node provides enough knowledge of local network topology. In this paper, the basic idea of PCAR is to develop a distance-vector routing protocol with an employment of the PIPC algorithm.

The rest of the paper is organized as follows. In Section II, we summarize related work on power control and packet routing. Then we propose the PCAR protocol in Section III. Following that, we evaluate its performance by theoretical analysis in Section IV and by simulation study in Section V. We compare it with related protocols in Section VI. Finally, we conclude this paper in Section VII.

II. Related Work

Power control plays an important role in creating network topology. In order to obtain good topological properties, there arose a wave of power control proposals that advocated the employment of classical computational geometry structures, such as RNG (Relative Neighborhood Graph), GG (Gabriel Graph), DG (Delaunay triangulation Graph), YG (Yao Graph), and MST (Minimum Spanning Tree), among others. Santi presented a comprehensive survey in [9]. Algorithms based on classical graphs usually hold good properties; however, most of those proposals do not pay enough attention to difficulties that may be encountered in real systems. Comparatively,

978-1-4244-7265-9/10/$26.00 ©2010 IEEE

degree-based algorithms, LMA/LMN [1] for example, are more practical. However, most degree-based algorithms are too rough and crude since they usually adjust the transmission power by a fixed step. Motivated by those observations, we proposed closed-loop PIPC [4]. In this paper, we implement PIPC as an important part of the PCAR protocol.

Packet routing is an interesting and fundamental problem in wireless ad hoc networks. Either topological routing or geographical routing has drawn a significant amount of research interest over years. The next-hop topological routing methods can be categorized into two primary classes: link-state and distance-vector. Compared with the link-state method, the distance-vector method is preferable, because it is computationally more efficient, easier to implement and requires much less storage space. The distance-vector method is based on the classical distributed Bellman-Ford algorithm. It is well known that this algorithm can cause the formation of loops. Perkins and Bhagwat [5] solved this problem through the destination-sequenced distance-vector (DSDV) method. In this paper, PCAR avoids loops in the same way as DSDV.

Previous power controlled routing proposals are more related to this paper. Narayanaswamy et al. [10] proposed COMPOW. The basic idea is that all nodes use the same transmission power, which is minimized without destroying network connectivity. The drawback is that a relatively separate node may lead to a high transmission power. Kawadia and Kumar [11] proposed CLUSTERPOW and MINPOW. As with the COMPOW proposal, CLUSTERPOW and MINPOW are combinations of power control and routing. CLUSTERPOW has a good performance in the sense of spatial reuse, and MINPOW is optimal in the sense of energy efficiency. However, both CLUSTERPOW and MINPOW are too costly. The PCAR protocol proposed in this paper is also a joint consideration on power control and packet routing. Section VI presents a brief comparison with COMPOW, CLUSTERPOW and MINPOW.

III. PCAR Protocol

A. An Overview

PCAR is a power controlled routing protocol. It consists of two parts: Bellman-Ford routing and proportion-integral power control.

1) Bellman-Ford Routing: PCAR constructs routing tables through the classical distributed Bellman-Ford algorithm. As with DSDV [5], the PCAR protocol utilizes sequence numbers to avoid the formation of loops. PCAR can forward a packet along the shortest path in the sense of number of hops, since each PCAR node stores the next hop of the shortest path for each destination.

2) Proportion-Integral Power Control: We implement PIPC [4] in PCAR for power control. The power control algorithm accepts the routing table as its input. The derived topology is of course also reflected in the routing table, i.e., the derived topology is fed back to the input of the power control algorithm. PCAR adjusts the transmission power such that the derived topology is between RNG and GG.

B. Routing Table and Forwarding Algorithm

The routing table of each node u has the format

$$RT_u = \{(destID, destPos, transPower, hops, nextHop, entryStatus, seqNum, routeChanged, lastModified)\},$$

where RT_u denotes the routing table of node u, $destID$ is the identification of a destination node, $destPos$ is the position of that destination, $transPower$ is the transmission power of that destination, $hops$ is the number of hops from node u to that node, $nextHop$ is the next hop on the shortest path, $entryStatus$ is used to handle the asymmetry problem, $seqNum$ is the sequence number indicating timeliness, $routeChanged$ indicates whether the route is changed or not, and $lastModified$ records the last time the entry was modified. It is important to note that $lastModified$ is node u's time, and hence PCAR does not require that the network should be synchronized.

The forwarding algorithm at each node u is described as follows.

```
1: ForwardPacket(packet, u){
2:     if (u is the destination of packet)
3:         Send packet to the upper layer;
4:     else{
5:         Find the entry e in RT_u such that e.destID is the
             destination of packet;
6:         Forward packet to e.nextHop;
7:     }
8: }
```

C. Construction of Routing Tables

At the beginning, the routing table RT_u of each node u contains only one entry that is about itself. The entry is always stored in $RT_u[0]$, the number 0 row in u's routing table, and is initialized as follows.

```
01: InitRoutingTable(u){
02:     RT_u[0].destID = u.nodeID;
03:     RT_u[0].destPos = u.nodePos;
04:     RT_u[0].transPower = u.transPower;
05:     RT_u[0].hops = 0;
06:     RT_u[0].nextHop = u.nodeID;
07:     RT_u[0].entryStatus =BICONNECTED;
08:     RT_u[0].seqNum = 0;
09:     RT_u[0].routeChanged =YES;
10:     RT_u[0].lastModified = u.currentTime;
11: }
```

After initialization, each node may be triggered somehow to broadcast its routing table to its neighbors. The broadcasted part of RT_u, denoted by RT_u^B, is as follows.

$$RT_u^B = \{(destID, destPos, transPower, hops, nextHop, entryStatus, seqNum)\}.$$

When a node receives a neighbor's routing table, say RT_{neigh}^B, it updates its own according to the neighbor's knowledge. The procedure is described as follows.

01: ConstructRoutingTable(RT_u, RT^B_{neigh}){
02: for (each entry ent^B_{neigh} of RT^B_{neigh}){
03: if (no corresponding entry in RT_u){
04: Insert ent^B_{neigh} into RT_u as a new entry ent_{new};
05: $ent_{new}.nextHop = neigh.nodeID$;
06: $ent_{new}.hops++$;
07: $ent_{new}.routeChanged =$ YES;
08: $ent_{new}.lastModified = u.currentTime$;
09: }
10: else{
11: if ($ent^B_{neigh}.nextHop == u.nodeID$){
12: $ent^B_{neigh}.hops =$ INFINITY;
13: }
14: else {
15: $ent^B_{neigh}.nextHop = neigh.nodeID$;
16: $ent^B_{neigh}.hops++$;
17: }
18: /* Suppose the corresponding entry is ent_{crsp}. */
19: if ($ent^B_{neigh}.seqNum > ent_{crsp}.seqNum$ or
($ent^B_{neigh}.seqNum == ent_{crsp}.seqNum$ and
$ent^B_{neigh}.hops < ent_{crsp}.hops$)){
20: $ent_{crsp} = ent^B_{neigh}$;
21: $ent_{crsp}.routeChanged =$ YES;
22: $ent_{crsp}.lastModified = u.currentTime$;
23: }
24: } /* End of else at line 10. */
25: } /* End of for at line 02. */
26: After some time T_{BC}, node u broadcasts RT^B_u that is composed of all route-changed entries.
27: }

It should be noted that the procedure only shows the basic idea behind constructing routing tables. In reality, it is much more complicated than what is stated here.

D. Triggering Updates

The mechanism to trigger updates of routing tables is as follows.

(1) Each node updates $RT[0].destPos$ every T_{UP} (Time to Update Position) time. If the position change is greater than some predefined MIN_MOVEMENT_TO_UPDATE, then PCAR assigns the new position to $RT[0].destPos$, and executes
$RT[0].seqNum = RT[0].seqNum + 2$;[1]
$RT[0].routeChanged =$YES;
$RT[0].lastModified = currentTime$.

(2) If the routing table has changed for a time longer than some predefined T_{BC} (Time to Broadcast Change), then the node updates its position information as stated in (1), and immediately broadcasts all the route-changed entries to its neighbors. After that, the node clears $routeChanged$ for each broadcasted entry.

[1]It should be noted that, as considered in (5), the $seqNum$ of a route may be generated by other nodes if the destination node is lost. In order to let the destination have a higher priority, $seqNum$ is updated as follows: If a node updates its own sequence number, it executes $seqNum = seqNum + 2$, otherwise, $seqNum = seqNum + 1$.

(3) No matter whether the routing table changes or not, each node periodically broadcasts its routing table to its neighbors. It is worth noting that the broadcast cycle T_{BA} (Time to Broadcast All) should be much greater than T_{BC}.

(4) Each node runs the PIPC algorithm once a T_{PC} (Time to trigger Power Control) time. In order to avoid the situation where many neighboring nodes adjust the transmission power simultaneously, we use T_{PC} plus a random time as the power control cycle. If the new transmission power is the same as the old one, no action is taken. Otherwise, PCAR assigns the new power to $RT[0].transPower$, then executes
$RT[0].seqNum = RT[0].seqNum + 2$;
$RT[0].routeChanged =$YES;
$RT[0].lastModified = currentTime$.
It is worth pointing out that T_{PC} should be long such that the node has enough time to update its routing table. In reality, too frequent power switching make little sense due to increased switching cost.

(5) If an entry ent of the routing table has not been updated for a long time, longer than some predefined T_{LOST} (Time to believe LOST), then node $ent.destID$ is believed to be lost, but is not immediately deleted. In this case, PCAR executes
$ent.entryStatus =$LOST;
$ent.hops =$INFINITY;
$ent.seqNum++$;
$ent.routeChanged =$YES;
$ent.lastModified = currentTime$.

(6) If an entry ent with $ent.entryStatus =$LOST has not been updated for a time longer than some predefined T_{DEAD} (Time to believe DEAD), then it should be deleted since node $ent.destID$ may have failed or encountered something else wrong.

(7) For each node in the network, as soon as it receives the routing table broadcasted by a neighbor, it updates its own routing table according to the neighbor's routing table. The basic idea of the updating process is already presented in III-C.

E. Handling Asymmetry

Each usable data link should be symmetric, i.e., node u is a neighbor of v if and only if node v is a neighbor of u. Asymmetric communication links are impractical, because many communication primitives become unacceptably complicated. In PCAR, the transmission power may differ significantly from one node to another, which generates a great number of unidirectional links. Even if all nodes use a common transmission power, there may still exist many unidirectional links due to asymmetry of radio path loss in the real-world environment. It is well known that some MAC protocols have taken care of the asymmetry problem. For example, IEEE 802.11 solves the problem through the RTS/CTS handshake for unicast. However, to the best of our knowledge, no MAC protocols take asymmetry into account

for the case of broadcasting. PCAR constructs routing tables through broadcasts, which may well result in asymmetric links. To handle that problem, PCAR utilizes $entryStatus$ whose value may be one of LOST, INLINKED, BICONNECTED, and INLINKED_MULTIHOP_BICONNECTED. Suppose that there is an entry with respect to node v in u's routing table. LOST indicates that node u cannot hear anything of node v. INLINKED implies a unidirectional link from v to u (whether bidirectional is not known as yet). BICONNECTED means that each link on the established path from u to v is bidirectional. INLINKED_MULTIHOP_BICONNECTED represents that u and v are BICONNECTED through two or more hops and there exists a unidirectional link from v to u. Therefore, only those BICONNECTED and INLINKED_MULTIHOP_BICONNECTED entries are available when forwarding packets. Suppose node u has received the routing table of node v, the asymmetry problem at each node u is handled as follows.

```
01: HandleAsymmetry(RT_u,RT_v^B){
02:     if (there exists an entry in RT_v^B whose destID is u,
            and ((entryStatus ==INLINKED or
                  INLINKED_MULTIHOP_BICONNECTED)
                 or (entryStatus ==BICONNECTED and
                                        hops == 1))){
03:       /* Link (v,u) is bidirectional.*/
04:       for (each BICONNECTED or INLINKED_
              MULTIHOP_BICONNECTED entry of RT_v^B)
05:         Update RT_u as ConstructRoutingTable() does;
06:     }
07:     else{
08:       /* Link (v,u) is unidirectional.*/
09:       Add the INLINKED property to the entry with
            respect to v in RT_u;
10:       Update other properties of that entry.
11:     }
12: }
```

Line 10 includes a process that is something like ConstructRoutingTable described in III-C. It is important to note that the handling-asymmetry algorithm should be integrated into the procedure of constructing routing tables. But in this paper, we introduce them separately; otherwise, one is so easily confused.

F. Power Control

PCAR employs the PIPC algorithm to adjust transmission power such that the derived topology is between RNG and GG. The RNG derived from a given graph $G = (V, E)$ consists of all links $(u, v) \in E$ such that the intersection of the two circular areas centered at u and v and with radius $d(u, v)$ does not contain any node w from V, where $d(u, v)$ is the Euclidean distance between u and v. The GG contains link (u, v) if and only if $disk(u, v)$ contains no other nodes of V, where $disk(u, v)$ is the disk with edge (u, v) as its diameter. For each node, the transmission radius in RNG and GG is denoted by $rngRadius$ and $ggRadius$, respectively. The transmission power of each node is adjusted such that the transmission radius is in the range of $[rngRadius, ggRadius]$. The PIPC algorithm at each node u is described as follows.

```
1: PIPC(RT_u(i)){
2:     Compute rngRadius(i) and ggRadius(i);
3:     Estimate control error e(i);
4:     Determine parameters K_P(i), K_I(i), and ζ(i);
5:     Δp(i) = K_P(i)(e(i) − e(i − 1)) + K_I(i)e(i) + ζ(i);
6:     Add Δp(i) to u.transPower;
7: }
```

In the PIPC algorithm, $RT_u(i)$ is the routing table of node u at digital time i, e is the control error, K_P is the proportion gain, K_I is the integral gain, and ζ is a modification term. We will not explain PIPC anymore, so please refer to [4] for more details.

IV. Properties of PCAR

Property 1: PCAR is loop free.

PCAR employs the same approach to avoid loops as DSDV. Perkins and Bhagwat [5] have proved that the routing method is loop free.

Property 2: The routing table of a node is stable under PCAR if $e(i)$ exactly reflects the control error at any digital time i.

It is obvious that the routing table is stable if the transmission power is stable. The detail proof of the stability of the transmission power was presented in [4].

Property 3: When PCAR has converged into a steady state over a given network G, the derived topology $PCAR(G)$ holds the following properties:

(1) Connectivity: $PCAR(G)$ is strongly connected if G is strongly connected;
(2) Symmetry: PCAR always forwards packets over bidirectional links;
(3) Sparseness: The number of links in $PCAR(G)$ is in the order of the number of nodes if G is a UDG[2], i.e., $l = O(n)$, where l is the number of links, and n is the number of nodes;
(4) Planarity: $PCAR(G)$ contains no two intersecting edges if G is a UDG.

According to III-E, PCAR preserves symmetry. As for connectivity, sparseness and planarity, please refer to [4] for details.

Property 4: When PCAR has converged into steady state, packets are forwarded along the shortest path in the sense of the number of hops.

Property 4 is obvious since PCAR is based on the Bellman-Ford shortest-path algorithm and the path length is measured in the number of hops.

[2] G is referred to as a UDG (Unit Disk Graph) if the network is homogeneous, i.e., two nodes can communicate as long as their Euclidean distance is no more than a threshold.

TABLE I
PARAMETERS OF PCAR

Parameters	Values (seconds)	Descriptions
T_{UP}	15	Time to update position information.
T_{BC}	0.1	Time to broadcast changed entries.
T_{BA}	15	Time to broadcast all entries.
T_{PC}	105	Basic time to execute PIPC.
T_{LOST}	45	Time to believe a node lost.
T_{DEAD}	90	Time to believe a node dead.

TABLE II
FIXED SETTINGS OF THE INPUT FILE FOR GLOMOSIM.

Parameters	Values
SIMULATION-TIME	200M
NODE-PLACEMENT	RANDOM
NODE-MOBILITY	RANDOM-WAYPOINT
MOBILITY-WP-PAUSE	30S
MOBILITY-WP-MIN-SPEED	0
MOBILITY-POSITION-GRANULARITY	0.5m
PROPAGATION-LIMIT	-111dBm
PROPAGATION-PATHLOSS	TWO-RAY
TEMPERATURE	290.0K
RADIO-TYPE	RADIO-ACCNOISE
RADIO-FREQUENCY	2.4e9Hz
RADIO-BANDWIDTH	2000000bps
RADIO-RX-TYPE	SNR-BOUNDED
RADIO-RX-SNR-THRESHOLD	10.0
RADIO-ANTENNA-GAIN	0.0dBm
RADIO-RX-SENSITIVITY	-91dBm
RADIO-RX-THRESHOLD	-81dBm
MAC-PROTOCOL	802.11
ROUTING-PROTOCOL	PCAR
NETWORK-PROTOCOL	IP
NETWORK-OUTPUT-QUEUE-SIZE-PER-PRIORITY	100

V. SIMULATION STUDY

For static networks, we have presented a simulation study on power stability and derived topologies under PIPC [4]. It shows that simulation results coincide very well with theoretical results. However, it is difficult to theoretically predict how PCAR behaves over highly-dynamic wireless ad hoc networks. Therefore we simulate PCAR on Glomosim 2.03 to investigate its adaptability to dynamics. Parameters of PCAR are shown in Table I, and fixed settings of Glomosim are shown in Table II.

We randomly deploy 100 nodes in a $1000m \times 1000m$ region. The transmission power of each node is adjusted between -10dBm and 20dBm. Since there are usually only a limited number of available power levels in real systems, we round the transmission power to one of $\{0, \pm 2, \pm 4, \pm 6, \pm 8, \pm 10, 12, 14, 16, 18, 20\}$. We use the RANDOM-WAYPOINT mobility model. Figure 1 demonstrates how the average transmission radius varies with time in comparison with the average $rngRadius$ and $ggRadius$, where the unit time T is 15 seconds, and the maximum speed MOBILITY-WP-MAX-SPEED is 10 meters per second. It shows that PCAR has a good effect on power control by utilizing the PIPC algorithm, since the average transmission radius is between $rngRadius$ and $ggRadius$ for the most part. It should be noted that, in the beginning of the simulation, PCAR needs some time to construct routing tables.

We further investigate the delivery ratio of PCAR. We randomly deploy 100 nodes in a $500m \times 500m$, $1000m \times 1000m$, and $2000m \times 2000m$ region, respectively. Figure 2 shows how the delivery ratio varies with MOBILITY-WP-MAX-SPEED. The total number of packets is 11990. The delivery ratio decreases with respect to MOBILITY-WP-MAX-SPEED. It is also somehow related to node density in mobile networks. The denser the nodal distribution is, the lower the delivery ratio is. If the average node density under uniform random distribution is higher than $1/2500m^2$, the delivery ratio of PCAR is still greater than 70% even if the maximum speed MOBILITY-WP-MAX-SPEED is as high as 20 meters per second. It shows that PCAR has a good adaptability to network dynamics.

VI. COMPARISON WITH RELATED PROTOCOLS

To the best of our knowledge, COMPOW [10], CLUSTERPOW [11], and MINPOW [11] are more practical and more related to PCAR than other proposals. In this section, we compare them with PCAR in terms of control mode, end-to-end latency, space complexity, software structure, and adaptability to heterogeneity. Table III presents a quick summary.

PCAR, COMPOW, CLUSTERPOW, and MINPOW are all power controlled routing protocols, but PCAR differs very much from the other three. First of all, PCAR is closed-loop, whereas the other three are open-loop. As a result, PCAR has a much better adaptability to network dynamics than the other three. PCAR and COMPOW employ the per-node control strategy, i.e., the transmission power at a certain time does not differ from packet to packet. But in COMPOW, all nodes use a common transmission power that is minimized without destroying network connectivity, which has a drawback as a relatively separate node may lead to a high transmission power. CLUSTERPOW and MINPOW employ the per-packet control, i.e., the transmission power is adjusted for each packet. Therefore, it may differ from packet to packet even over a static network. It seems that the per-packet control may well have a better performance, for example, MINPOW can forward packets along the minimum energy path. However, this is not necessarily the case, because per-packet control requires frequent power switching, which results in a high end-to-end delay and an extra overhead. CLUSTERPOW constructs complicated routing tables for per-packet control, and MINPOW pays a high price, since it has to figure out the minimum required transmission power over each link. As for storage space, PCAR and MINPOW have an $O(n)$ space complexity since they are based on the distributed Bellman-Ford algorithm, where n is the number of nodes. COMPOW and CLUSTERPOW maintain routing tables at each power level. As a result, the space complexity is $O(mn)$,

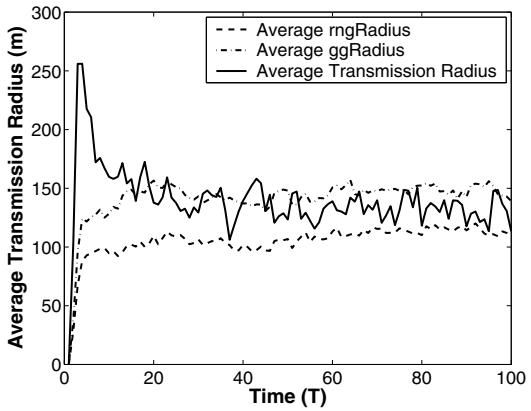

Fig. 1. Average transmission radius varies with time.

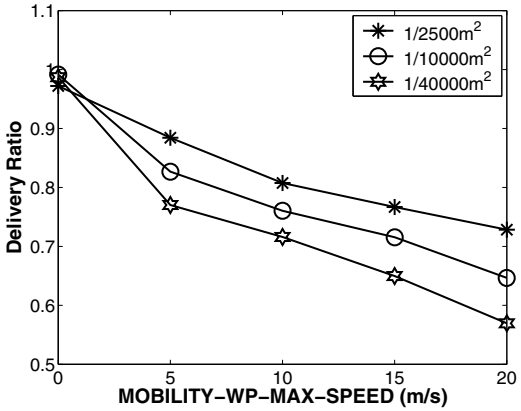

Fig. 2. Delivery ratio varies with speed.

TABLE III
COMPARISON WITH RELATED PROTOCOLS.

Protocols	Control Mode	End-to-end Latency	Space Complexity	Software Architecture	Heterogeneity
PCAR	Closed-loop, per node	Low	$O(n)$	Loosely coupled	Possible
COMPOW [10]	Open-loop, per node	Low	$O(mn)$	Tightly coupled	No
CLUSTERPOW [11]	Open-loop, per packet	High	$O(mn)$	Tightly coupled	No
MINPOW [11]	Open-loop, per packet	High	$O(n)$	Loosely coupled	Possible

where m is the number of available power levels. From the perspective of software architecture, PCAR is better than COMPOW and CLUSTERPOW, because power control and routing are not so tightly coupled in PCAR as in COMPOW or CLUSTERPOW. Finally, PCAR does not depend on any assumption of homogeneity, whereas neither COMPOW nor CLUSTERPOW can be applied to heterogenous networks. In summary, PCAR has a better performance.

VII. CONCLUSION

Motivated by the observation on the necessity and feasibility of the combination of power control and routing, we propose a new PCAR (Power Controlled Ad hoc Routing) protocol for wireless ad hoc networks. The basic idea is to implement the previously proposed PIPC (Proportion-Integral Power Control) algorithm based on the classical distributed Bellman-Ford routing mechanism. We carry out a full-scale evaluation of PCAR, including a theoretical analysis of its properties over static networks, a simulation study of its adaptability to network dynamics, and a comparison with other related power controlled routing protocols. It shows that PCAR has a good performance.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grants No. 60903156, No. 60903158, No. 60903155, No. 60703114, No. 90718031, and No. 60721002, the National Basic Research Program of China (973) under Grant No. 2009CB320705, and US NSF grants CNS 0422762, CNS 0434533, CNS 0531410, and CNS 0626240.

REFERENCES

[1] M. Kubisch, H. Karl, A. Wolisz, L. Zhong, and J. Rabaey, "Distributed algorithms for transmission power control in wireless sensor networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2003, pp. 16–20.

[2] N. Li and J. C. Hou, "Topology control in heterogeneous wireless networks: problems and solutions," in *Proceedings of the IEEE Conference on Computer Commnications (INFOCOM)*, 2004, pp. 232–243.

[3] V. Kawadia and P. R. Kumar, "Principles and protocols for power control in wireless ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 76–88, 2005.

[4] X. Zhang, Z. Li, S. Lu, D. Chen, and X. Li, "Proportion-integral power control for wireless ad hoc networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2008.

[5] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proceedings of the ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM)*, London, UK, 1994, pp. 234–244.

[6] M. Lim, A. Greenhalgh, J. Chesterfield, and J. Crowcroft, "Landmark guided forwarding," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, Boston, Massachusetts, USA, 2005, pp. 169–178.

[7] U. G. Acer, S. Kalyanaraman, and A. A. Abouzeid, "Weak state routing for large scale dynamic networks," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2007, pp. 290–301.

[8] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2007, pp. 169–180.

[9] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Computing Survey*, vol. 37, no. 2, pp. 164–194, 2005.

[10] S. Narayanaswamy, V. Kawadia, and R. S. Sreenivas, "Power control in ad-hoc networks: theory, architecture, and implementation of the COMPOW protocol," in *Proceedings of the European Wireless Conference*, 2002, pp. 156–162.

[11] V. Kawadia and P. R. Kumar, "Power control and clustering in ad-hoc networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2003, pp. 459–469.