On Topology Improvement of a Packet Radio Network by Power Control

Chi-Fu Huang and Yu-Chee Tseng, *Senior Member, IEEE*

***Abstract*—The packet radio network (PRN) is an attractive architecture to support wireless data communication. The *code assignment* problem in PRN is a classical problem that has been extensively studied. However, in this paper, we observe that the *power control* issue has been ignored by most works but may have significant impact on the PRN's performance. Given a set of PRN stations, the network topology can be changed by adjusting each station's transmission power. All existing works, nevertheless, assume that the network topology is given before solving the code assignment problem. In this paper, we regard code assignment as an independent problem and show how to improve the network topology by power adjustment without violating the original code assignment. The improvement in topology (such as more links in the network) may result in improvement in network throughput. Through simulations, we demonstrate that although the code assignment problem is NP-complete, our power adjustment schemes can easily improve the network performance by about 10% with polynomial costs.**

***Index Terms*—Code assignment, mobile computing, packet radio network (PRN), power control, wireless communication.**

I. Introduction

THE packet radio network (PRN) was first demonstrated in 1969 at the University of Hawaii [1] and since then has greatly increased its presence and importance for computer communications. A PRN consists of a number of *stations* placed in a geographically distributed area, where each station has a computer and a transceiver. Two stations are said to be *connected* if they are in each other's communication range. A PRN can be considered as a graph with a certain topology. It is sometimes referred to as a *multihop* PRN to reflect the fact that two stations may communicate indirectly by relaying stations.

The code assignment problem in PRN is a traditional issue that has been widely studied [2]–[8]. The problem is reviewed in Section II. A tree-based code assignment scheme is proposed in [3], where it is also shown that determining the least number codes for any network is NP-complete. The scheme is further extended to a distributed version, by adopting a concept called *traveling token*. Also based on traveling tokens, [4] proposes a distributed assignment scheme. Heuristics are proposed for regular and general PRNs in [2], [6], and [7]. A transmitter-oriented heuristic is presented in [6]. Reference [2] suggests assigning codes based on stations' degrees. Reference [7] tries to give a code to a station if it has the most neighbors already receiving codes. The concept of *maximum independent set* is used in [8] for broadcast scheduling, which can also be used for code assignment.

The above results are suitable for traditional PRNs with low or no mobility. Some recent protocols start to be able to tolerate mobility [9]–[11]. The protocol in [10] employs a polling mechanism. Once polled, an intending sender will use its sending code to transmit. In [9], the protocol assigns channels to stations dynamically. It requires that the channel assigned to a station be different from those of its two-hop neighbors. A *hop-reservation* median access control (MAC) protocol based on very slow frequency-hopping spread spectrum is proposed in [11].

While tackling the code assignment problem in PRN, we observe an interesting point: the *power control* issue has been ignored by most works but may have significant impact on PRN's performance. Given a set of PRN stations, the network topology can be changed by adjusting each station's transmission power. All existing works, nevertheless, assume that the network topology is fixed. Then, based on the given network topology, the code assignment problem is addressed. Conceivably, if hosts' transmission powers are not fixed, the network topology (and thus the code assignment result) may change. Take an extreme case as an example. If all stations' powers are tuned infinitely large, the network will be fully connected but the number of codes required will be equal to the network size. By reducing powers, the network connectivity reduces, but the number of codes required also reduces. Note that stations do not necessarily have equal transmission powers. Our goal is to improve the network topology, by means of power control, to improve a PRN's performance.

The purpose of this paper is not to propose a new code assignment solution. Instead, we show, given a PRN in which each station already received a code, how to adjust the powers of stations to obtain a "better" network without violating the original code assignment. By adjusting powers, we try to control/improve the topology of a PRN. So our result can be regarded as building on top of those code assignment solutions. Three schemes—*distance-based*, *degree-based*, and *load-based*—are proposed. On top of these, we also propose a *code randomization* mechanism to further improve performance. Distributed versions of these schemes are also introduced. Through simulations, we demonstrate that although the code assignment problem is NP-complete, and thus improving the code assignment is computationally very expensive, using our power adjustment schemes can easily improve the network performance by about 10% with polynomial costs.

Some works have addressed the power issue, but under different scenarios. Assuming a contention-based channel model

Manuscript received October 29, 2002; revised April 6, 2003. This work was sponsored by the MOE Program for Promoting Academic Excellence of Universities under Grants A-91-H-FA07-1-4 and 89-E-FA04-1-4.

The authors are with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan, R.O.C.

Digital Object Identifier 10.1109/TVT.2003.814229

(such as ALOHA or CSMA), [12] and [13] show how to determine the optimal transmission ranges in PRN, where stations' powers can be equal and nonequal, respectively. Reference [14] also considers topology control; its goal is to obtain a connected or biconnected network such that the maximal power used by all stations is minimum. Energy-efficient communication for sensor networks is addressed in [15]. In the area of mobile ad hoc networks (MANETs), power issues have been studied on the MAC layer [16] and routing layer [17]–[19]. Energy-efficient broadcasting and multicasting for MANETs is addressed in [20]–[22]. Reference [23] discussed how to assign codes to hosts so as to minimize either the total power consumption or the congestion factor. However, the network topology, in terms of communication pairs, is still predefined and cannot be changed during the optimization process.

The PRN has evolved since its first appearance. One example is Metricom Inc.'s Ricochet,[1] which has been installed in many major U.S. cities. Stations can be placed on poletops and packets may go through multiple stations before reaching a router, which is connected to the Internet. A similar product is Nokia's RoofTop Wireless Routing solution,[2] where wireless routers are placed on rooftops in a residential area. These routers form a mesh network, and to indoor users each router acts as a residential gateway.

The remainder of this paper is structured as follows. Section II gives the preliminaries and motivations, followed by formal definitions of the power adjustment problems to be solved. Section III proposes several centralized power-adjustment schemes assuming that only a single code is assigned to each station. Distributed versions of those schemes are in Section IV. Section V extends results to the case where multiple codes can be assigned to a station. Simulation results are presented in Section VI. Conclusions are drawn in Section VII.

II. Preliminaries and Motivations

A. Review of Code Assignment Schemes for PRN

A PRN consists of a number of stations. Traditionally, it is assumed that the connectivity between stations is predetermined. Based on this connectivity, the network can be considered as a static graph. (However, this is not the case if transmission powers, and thus connectivities, are adjustable.)

Suppose that the topology of a PRN is static. Then codes have to be assigned to stations for them to communicate with each other. Note that "code" is a logical term and could be a sequence of *time slots* appearing periodically under the time-division multiple-access (TDMA) mode, a *frequency band* under the frequency-division multiple-access (FDMA) mode, or an *orthogonal code* under the code-division multiple-access (CDMA) mode. A code can be regarded as a resource unit that a station can use freely without worrying about the interference problem. Following the *sender-based* model that is widely assumed in many works [2], [5], [7], [8], we need to consider two types of collisions when assigning codes to stations: *primary* and *secondary*. A primary collision occurs when two stations using the same code can hear each other; a secondary collision occurs when two stations using the same code can be heard by a third station (sometimes known as the *hidden-terminal problem*). Note that from a receiver's point of view, for TDMA and CDMA, one transceiver is sufficient. But for FDMA, either it has to tune to the proper channel at the proper time or multiple transceivers are needed.

Both primary and secondary collisions should be avoided while assigning codes. The PRN can be modeled as an undirected graph $G = (V, E)$, where V denotes the set of stations and E denotes the set of wireless links. So the problem becomes one of assigning a code to each station such that no two stations at a distance of one or two share the same code. Since codes represent wireless resources, the typical goal of code assignment is to minimize the total number of codes used. This problem can be further translated to the *vertex coloring problem* in graph theory. Specifically, G can be translated to another graph $G' = (V, E')$ such that between each pair of vertices in V there is an edge in G' if their distance is one or two in G. The original vertex coloring problem is known to be NP-complete. The code assignment problem after translation to the vertex coloring problem remains so [2], [24]. As a result, most existing results are based on heuristics.

The following reviews are all based on G'. Reference [24] formulates the code assignment problem as a graph coloring problem. Vertices are assigned codes in a decreasing order of their identifiers (ID). When being examined, a vertex is assigned the smallest possible code that is not yet used by any of its neighbors. The basic idea is to utilize codes as compact as possible. However, the characteristics of the network topology are not fully exploited—if IDs are given without any relation to the network characteristics, the results of this procedure are analogous to those obtained through a random choice of the next vertex to be colored at each step [2].

To remedy the random choice problem, [2] differentiates vertices by their degrees. Increasing and decreasing orders of vertex degrees are investigated. When examining each vertex, the same strategy of choosing the smallest possible code as in [24] is adopted. The experiments in [2] demonstrate that using a decreasing order of degrees is a better choice because vertices with higher degrees tend to be more constrained in their choice of colors (due to a more crowded neighborhood). If they are assigned codes at later time, one may encounter the danger of requiring more colors.

The sequence in which vertices are examined is further investigated by [7]. A concept called *saturation* is proposed. In this scheme, vertices are given codes based on a priority defined by the number of different codes that have been occupied by a vertex's neighbors. The more occupied codes, the higher the priority. In cases of ties, vertices with more neighbors already owning codes (the same code occupied by different neighbors is counted multiple times) are assigned first.

B. The Power Adjustment Problem

The above discussion has assumed that the topology of the PRN is already known, based on which the code assignment problem is solved. In this paper, we assume that the network topology is adjustable by controlling stations' transmission powers. As a result, there are two parameters to be decided

[1]Metricom, Los Gatos, CA: http://www.metricom.com.

[2]RoofTop: http://www.wbs.nokia.com.

together: transmission powers and codes. The transmission powers determine the network topology, and under this topology the code assignment should have no primary and secondary collisions. The ultimate goal is to maximize the network performance. Note that our goal in this paper is not to propose a new code assignment solution. Instead, we show how to improve a network's topology by power adjustment, which may in turn improve the network performance, even if the code assignment part is unchanged.

In this paper, we assume that the transmission power of a station should fall in a reasonable range $[P_{\min}, P_{\max}]$. For example, in IEEE 802.11 [25], the transmission power should be no less than 1 mW and no greater than 100 mW. The typical transmission distance ranges a few hundred meters. In Bluetooth [26], a device can transmit in three power classes. Class 1 should output 100 mW (+20 dBm), with a minimum of 1 mW (0 dBm). Class 2 should output 2.4 mW (+4 dBm), with a minimum of 0.25 mW (−6 dBm). Class 3 has the lowest power, with a nominal output of 1 mW. The transmission distance could be a few to a few tens of meters. However, note that energy efficiency is not a concern in this paper: we mainly focus on network throughput.

Below, we give the formal problem statements, in two versions.

Definition 1: Single-Code Assignment With Power Adjustment (SAPA): Given a set of stations $\{1, 2, \ldots, n\}$, where each station i is placed in a location $L_i, i = 1, \ldots, n$, our goal is to assign each station i, $i = 1, \ldots, n$, a code based on the sender-based rule and a transmission power level P_i satisfying $P_{\min} \leq P_i \leq P_{\max}$ such that the network throughput is maximized.

Definition 2: Multicode Assignment With Power Adjustment (MAPA): Given a set of stations $\{1, 2, \ldots, n\}$, where each station i is placed in a location $L_i, i = 1, \ldots, n$, and a set of integers $\{t_1, t_2, \ldots, t_n\}$, our goal is to assign each station i a set of t_i codes based on the sender-based rule and a transmission power level P_i satisfying $P_{\min} \leq P_i \leq P_{\max}$ such that the network throughput is maximized.

Note that MAPA is an extension of SAPA by allowing more than one code for each station. The motivation is to take into account the difference of traffic loads among stations. When all t_is are equal, MAPA degenerates to SAPA. Given any network, the code assignment problem has been proved to be NP-complete even when all stations share a common transmission power [2], [24]. If we impose that $P_{\min} = P_{\max}$, SAPA will be reduced to the code assignment problem. Thus, SAPA and MAPA, which extend the code assignment problem, are both computationally intractable.

Also note that the metric throughput is a complicated notion and may depend on many factors, such as traffic loads and patterns. A universal definition cannot be given easily. In this paper, we choose to inject packets between randomly selected sources and destinations into the network for evaluation. Then the end-to-end throughput is used as the performance metric.

III. Centralized Solutions for SAPA

In this section, we propose several centralized solutions for the SAPA problem. We are given a set of stations. The goal is to determine the transmission power and code of each station, but stations do not necessarily have the same transmission power. This is achieved in two stages. In the first stage, we test different transmission powers but all stations' powers still remain uniform. This gives an initial network topology. In the second stage, we further improve the topology by modifying individual stations' powers. The procedure is outlined below. The power of station i is denoted as $P_i, i = 1, \ldots, n$.

1) Let $T = P_{\min}$.
2) For $i = 1, \ldots, n$, let $P_i = T$. Based on this power setting, construct a graph corresponding to the topology of the PRN.
3) Apply any heuristic for code assignment on the current topology of the PRN.
4) Evaluate the network quality factor Q (discussed below) of the current network topology and keep a record of the evaluation result.
5) Let $T = T + \delta$. If $T \leq P_{\max}$, then go to Step 2).
6) From the above evaluation records in Step 4), pick the power setting that gives the best network quality factor Q. Let the induced topology graph be G.
7) Based on G, conduct one of the proposed power adjustment schemes (discussed below).

In the above steps, we gradually increase the powers of stations by a hop of δ each time. The value δ will depend on how fine one expects to tune the powers. For each selected power, code assignment is involved to determine a code for each station. Then the network is evaluated for its quality factor Q. Note that Q is for us to choose a proper initial power level to be used by all hosts [further adjustment will be made in Step 7)]. Below, we propose two alternatives to define Q.

- $Q_1 = (1/\text{Num}_{\text{code}} \times D_{\text{avg}}^{\alpha})$, where Num_{code} is the total number of codes used and D_{avg} is the average hop count between each pair of stations. Parameter α defines a weight on the factor D_{avg}. Intuitively, a bigger Q_1 implies a better topology, since it is desirable to use a smaller number of codes and have a shorter average hop count.
- Q_2 is defined to be the network throughput based on the given topology and code assignment.

Note that Q_1 can be computed easily. Q_2 may be obtained from simulations by generating random traffic into the networks. In Section VI, we will compare these two alternatives.

Let T be the best common power selected in Step 6). By setting $P_i = T, i = 1, \ldots, n$, a topology G and a code assignment for each station are already obtained. In Step 7), power adjustment will be applied to individual stations. In the following, we propose several solutions for Step 7).

A. Distance-Based Scheme

Let P_i and c_i be power and code of station $i, i = 1, \ldots, n$. In the distance-based scheme, we will greedily increase the powers of individual stations to increase the network connectivity. By *network connectivity*, we simply count the number of links in the graph. The intuition is that a network with more links may have higher throughput. However, doing so is under the constraint that no primary and secondary collision should occur.

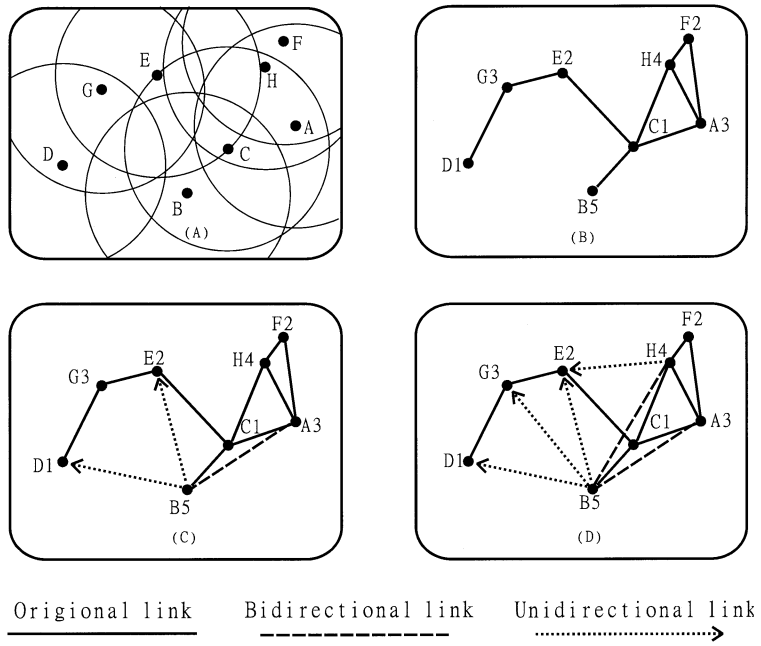

Fig. 1. An example of the distance-based scheme: (a) the original G and stations' transmission ranges, (b) the induced G and code assignment, (c) topology after adding link (B, A), and (d) topology after adding link (H, B).

The scheme works as follows. We first collect all station pairs that are not connected in G. These pairs are sorted in an ascending order according to their distances. Then we sequentially check each pair in the list for the possibility of adding it into G without changing their codes. Note that adding a link implies increasing the two end stations' powers. This is repeated until no more pairs can be added. In the following steps, the distance of stations i and j is represented by $\text{dist}(i,j)$, and the minimum transmission power required for two stations distanced by d to communicate is denoted by $\lambda(d)$.

a) Let L be the list of all station pairs (i,j) such that link $(i,j) \notin G$ and $\lambda(\text{dist}(i,j)) \leq P_{\max}$. Sort L in ascending order of the distance between each pair of stations.

b) Define a collision array $\text{col}[1,\ldots,n]$ such that $\text{col}[i]$ is the set of codes used by station i itself and all neighbors of i, i.e.,

$$col[i] = \{c_i\} \cup \{c_j | (i,j) \in G\}.$$

Intuitively, any station that is not adjacent to i in G and that intends to establish a link with i must not use any code in $\text{col}[i]$; otherwise, primary or secondary collision will occur.

c) Pick the first entry (i,j) in L. Check the following two conditions:

$$\forall k : \lambda^{-1}(P_i) < \text{dist}(i,k) \leq \text{dist}(i,j) \Longrightarrow c_i \notin \text{col}[k]$$

$$\forall k : \lambda^{-1}(P_j) < \text{dist}(j,k) \leq \text{dist}(i,j) \Longrightarrow c_j \notin \text{col}[k].$$

If both conditions hold, this means that adding a link between stations i and j will not suffer from primary and secondary collisions. If so, let $P_i = P_j = \lambda(\text{dist}(i,j))$ and update the collision array as follows:

$$\text{col}[k] = \text{col}[k] \cup \{c_i\}$$

for all k such that $\lambda^{-1}(P_i) < \text{dist}(i,k) \leq \text{dist}(i,j)$

$$\text{col}[k] = \text{col}[k] \cup \{c_j\}$$

for all k such that $\lambda^{-1}(P_j) < \text{dist}(j,k) \leq \text{dist}(i,j)$.

Intuitively, the first equation updates collision arrays interfered by i's higher powers (including j), while the second does so for those interfered by j's higher powers (including i).

d) Remove (i,j) from L. If L is not empty, go to Step c).

For example, Fig. 1(a) shows a PRN, where the circles (all of the same radius) indicate the transmission ranges of stations. The current topology G is shown in Fig. 1(b), where the number associated with each station is its code. After calculating list L, trials in Table I will be made. Note that trials 3 and 5 deserve special attention.

The resulting network is shown in Fig. 1(d), where two bidirectional links and four unidirectional links are newly added. As can be seen, station C, which was originally an articulation point and could be a heavily loaded bottleneck, is now not so any more. This is expected to relieve the network congestion significantly. Note that having unidirectional links is inevitable so long as one allows asynchronous transmission powers. We call those unidirectional links *side-effect links* and will not use them for routing packets in our performance simulation. However, guaranteed by the collision test in Step c), these side-effect links will not cause primary and secondary collisions.

List L could be as long as $O(\binom{n}{2}) = O(n^2)$. Sorting L takes time $O(n^2 \log n^2)$. The collision array can be as large as $\binom{n}{2}$ and can be easily computed in time $O(n^2)$. Steps c) and d) take time $O(n)$ (since at most n elements in the collision array need to be checked). Steps c) and d) will be looped at most $\binom{n}{2}$ times. So the overall time complexity of this scheme is $O(n^3)$. (The analysis for the subsequent schemes is similar and thus will be omitted.)

B. Degree-Based Scheme

The previous scheme uses distance as the metric to determine which link should be checked and added into the network first.

TABLE I
AN EXAMPLE OF THE DISTANCE-BASED SCHEME

Trail	Attempted Link	Result	Note
1	(H, E)	Failure	Increasing station H's power can be granted, but increasing station E's power will cause a secondary collision at station H.
2	(F, C)	Failure	Increasing F's power will cause a secondary collision at C.
3	(E, B)	Failure	Increasing B's power can be granted. But increasing E's power will cause a secondary collision at H (this is because $dist(E, H) < dist(E, B)$).
4	(D, B)	Failure	Increasing D's power causes a secondary collision at B.
5	(B, A)	Success	No collision will occur. So a link will be added between B and A, as shown in Fig. 1(c). Array entries $col[B]$ and $col[A]$ should be updated properly. Note that two directional links, one from B to D and one from B to E, as shown by dotted arrows in Fig. 1(c) will be added. So code 5 used by station B should be added to $col[D]$ and $col[E]$ too.
6	(F, E)	Failure	Primary collisions occur at both ends.
7	(E, D)	Failure	Secondary collision occurs at E.
8	(G, B)	Failure	Secondary collision occurs at B.
9	(G, C)	Failure	Secondary collision occurs at both ends.
10	(E, A)	Failure	Secondary collision occurs at both ends.
11	(H, B)	Success	No collision will occur. So a new link between H and B is added, as shown in Fig. 1(d). Four array entries, $col[B]$, $col[E]$, $col[G]$, and $col[H]$, should be updated.
			The rest of trials will all fail

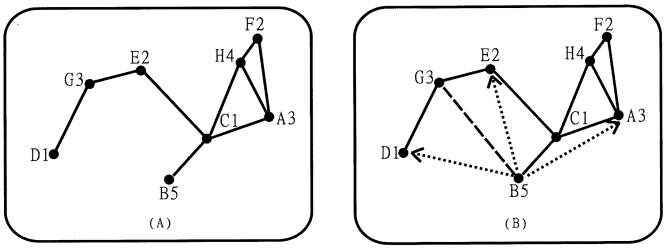

Fig. 2. An example of the degree-based scheme: (a) the original G and (b) the resulting topology.

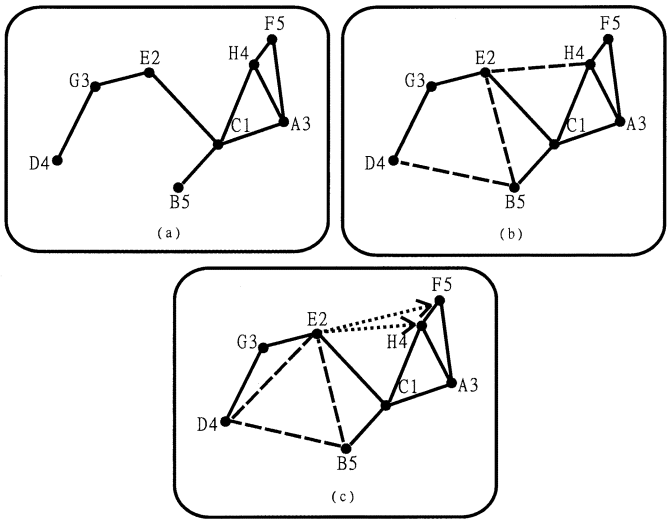

Fig. 3. Applying code randomization before power adjustment: (a) a new assignment by changing stations D's and F's codes, (b) the topology after performing the distance-based scheme, and (c) the topology after performing the degree-based scheme.

In this section, we propose to use stations' degrees in G as the metric. The rationale is: stations with lower degrees are weaker in communication capability and thus are more likely to become bottlenecks. Thus, adding links of weaker connectivities is more important.

The process to adjust powers is similar to the previous scheme, except that the order in which the potential links are checked is different. So we only briefly summarize the steps as follows. Note that similar to the distance-based scheme, the degree-based scheme can only increase powers of stations when no collision will occur. So the total number of codes will not be changed.

a) L is still the set of potential links to be added but is sorted differently according to stations' degrees as the primary key and distances as the secondary key, both in ascending order. Note that since each pair $(i, j) \in L$ has two stations, the lower value of the degrees of i and j is used for sorting.
b) Calculate the collision array (same as the distance-based scheme).
c) Pick the first $(i, j) \in L$ for possible power adjustment (same as the distance-based scheme).
d) Remove (i, j) from L. Also, if link (i, j) is added into the network in Step c), we should properly adjust the positions of all remaining links' in L that are incident to i or j (since both i's and j's degrees have been increased by one). Then go to Step c), if necessary.

Fig. 2 shows an example based on the same network in Fig. 1. The following sequence of trials will be made:

1) failure on (B, E), (D, B), (E, H), and (F, C);
2) success on (G, B);
3) failure on (A, B), (G, C), (H, E), and (C, F).

In this example, only one bidirectional link and three unidirectional links are added. Although the number of links being added to the network is less than that of the earlier distance-based scheme, the scheme has a different flavor by trying to make weaker stations stronger, in terms of connectivity. In fact, in our simulations (to be shown later), we do find many situations where this scheme will outperform the distance-based scheme.

C. Load-Based Scheme

As commented earlier, how well a network performs highly depends on the traffic pattern. In this section, we propose to

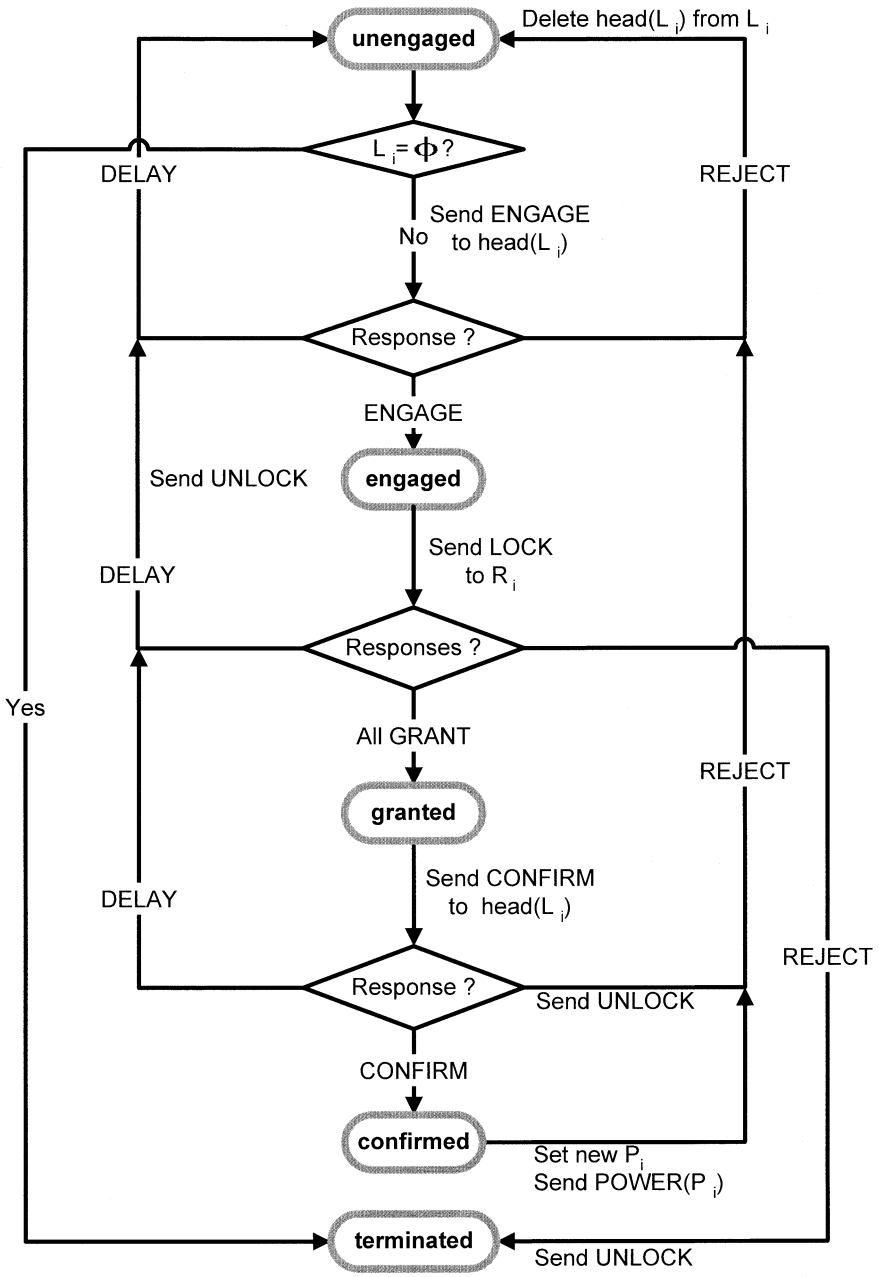

Fig. 4. State transaction diagram for power adjustment of station i.

use stations' traffic loads (instead of distances or degrees) as the metric to determine which station should increase its power first. Stations with higher traffic loads are more likely to become bottlenecks. So improving their connectivity is more desirable. We outline the procedure below. We assume that each station's load is already known.

a) L is the set of potential links to be added, which is sorted in descending order of links' loads as the primary key. (The secondary key could be distances or degrees.) The load of a link is the maximum of the loads of the two incident stations.
b) Calculate the collision array.
c) Pick the first $(i, j) \in L$ for possible power adjustment.
d) Remove (i, j) from L. Then go to Step c), if necessary.

D. Code Randomization

In this section, we propose a randomization mechanism that can be added on top of the above schemes to improve their performance. The main idea is to change the codes assigned to stations in Step 3). Let us use an example to motivate the idea. Observe Fig. 3, which represents the same network G in Figs. 1 and 2, but with different code assignment. The codes used by stations D and F are changed to 4 and 5, respectively. If we use this network G and apply the distance- and degree-based schemes on it, three bidirectional links will be added to the network, as shown in Figs. 3(b) and (c), respectively. This improves

the connectivity of the network as compared to the earlier examples, which shows the potential benefit of changing codes.

Indeed, as we surveyed several code assignment algorithms in the literature [2], [7], there is a tendency of favoring some set of codes over the others. The reason is quite obvious—the goal of code assignment is to use as few codes as possible. So the same code is likely to be used by stations that are physically close to form a compact code usage pattern. However, this is disadvantageous to our power adjustment, because when adding links, there will be more chance to find primary or secondary collisions.

Recall the network G, in which each station already has a code. Here we propose a simple *randomization technique* to change the code assignment. We sequentially pick each station in G and try to reselect a new code for it that has not been used by any of its two-hop neighbors. By so doing, the compact code usage pattern in the original assignment will be disturbed. However, the total number of codes used is not increased. The procedure is formally presented below. Each station i is already assigned a code c_i. This procedure should be run before the power adjustment [Step 7)] is executed.

i) Let C be the set of all codes used by the network after Step 3).
ii) Sequentially pick each station in an arbitrary order. For each station i, randomly pick a code from the set

$$C - \{c_j | \text{station } j \text{ is a 1-hop or 2-hop neighbor of } i \text{ in } G\}.$$

Then change c_i to this randomly selected code.

Note that in Step ii), the code that a station can select is the set of all codes excluding those that are used by its 1-hop and 2-hop neighbors. This set cannot be empty since it includes at least the station's current code. So Step ii) always succeeds and the randomization process will not increase the number of codes used.

IV. Distributed Solutions for SAPA

In this section, we extend our centralized power adjustment solution in Section III to a distributed protocol. Each station can adjust its power independently, and only local information needs to be collected. Distributed solutions are generally more favorable, accounting for concerns such as reliability, fault tolerance, load balance, and difficulties and costs in collecting global information.

Our protocol will allow multiple stations to adjust their powers simultaneously. To enable stations to make distributed decisions, at least two properties should be guaranteed. The first one is *correctness*, in the sense that no station's decision will cause primary and secondary collisions. In our protocol, we will let stations contend with each other to increase their powers. The contention is done by trying to lock neighboring stations. Only when all necessary stations are locked can a station adjust its transmission power. The second property is to ensure *freedom of deadlock* while locking neighboring stations. This is similar to the deadlock issue in operating system design, where one should avoid holding resources while waiting for more resources. In this paper, we will avoid this problem by preventing stations from entering a waiting status. Whenever a station cannot successfully lock all necessary stations, it simply unlocks all stations and retries later.

In the following, we first translate our scheme in Section III into a distributed one. Note that in the second step, a distributed code assignment protocol is required. Again, since we consider code assignment as an independent issue in this paper, any existing protocol may be adopted (e.g., [2], [4], and [5]).

1) When the network is at its initialization state, each station i picks an initial power T as its transmission power P_i such that $P_{\min} \le T \le P_{\max}$. (Here T can be a global constant, or a variable independently picked by each individual station based on statistics or its past experience.)
2) Apply a distributed code assignment protocol on the network based on the initial power. Let c_i be the code assigned to station i.
3) After a code is obtained, each station broadcasts its code to all its one-hop neighbors. On hearing a neighbor's code, a station should update this information into its collision array col[i].
4) Perform our distributed power adjustment protocol on each station (see the subsequent discussions).

Below, we will elaborate on Step 4). Our protocol is based on message exchange. We assume that there is a control channel (similar to that in [4]) for this purpose, and message delivery is reliable and has bounded delays.[3] Also, we assume that each station i is aware of all stations in its surroundings that it cannot reach directly by power T but may reach if it transmits with the maximum power $P_{\max}$. These stations are maintained in a list L_i and are sorted in an ascending order according to their distances to i. (This information may be obtained in the initialization stage, or by having each station sending a HELLO message from time to time with the maximum power.)

Station i will sequentially pick each station in L_i and try to increase its transmission power to reach it, until no further increment is possible. To ensure correctness and freedom from deadlock, each increment of power will involve four stages: unengaged, engaged, granted, and confirmed (see Fig. 4). In the following, we explain how station i proceeds from one stage to another:

a) *unengaged* $\Rightarrow$ *engaged:* Initially, station i stays in the unengaged state. Then it can check its list L_i. If $L_i \neq \emptyset$, an ENGAGE packet can be sent to the first station in L_i, denoted as head(L_i). This is to ask for head(L_i)'s cooperation to increase both sides' powers. After sending out the packet, station i should wait for a reply from head(L_i). If station head(L_i) also intends to increase its power, a positive ENGAGE packet will be replied; otherwise, a negative DELAY or REJECT packet will be replied. How station head(L_i) responds is based on the following rules:

i) ENGAGE: if head(L_i)'s state = "unengage" and head($L_{\text{head}(Li)}$) = i;
ii) DELAY: if head(L_i)'s state = "unengaged" and head($L_{\text{head}(Li)}$) $\neq i$;
iii) REJECT: if head(L_i)'s state = "terminated."

[3]We assume that these are supported by the underlying protocol. Reliable broadcast is beyond the scope of this work. Network-wide broadcast is addressed in [27]–[29]. Reliable one-hop broadcast is discussed in [30], [31].

If an ENGAGE packet is responded, station i will enter the "engaged" state. This happens when station head(L_i) also intends to increase its power and station i is the first station in head(L_i)'s list. If a DELAY packet is responded, station i will return to the "unengaged" state. This happens when station head(L_i) has the intention to increase its power but there are other stations in front of i in head(L_i)'s list. In this case, station i may retry later or wait for head(L_i)'s ENGAGE packet. If a REJECT packet is received, station i will delete head(L_i) from L_i and then return to the "unengaged" state to try the next station. This happens when head(L_i) is unwilling/unable to increase its power.

b) *engaged* $\Rightarrow$ *granted:* After entering the "engaged" state, station i should calculate a *request* set R_i, where

$$R_i = \left\{ r | \lambda^{-1}(P_i) < \text{dist}(i,r) \leq \text{dist}\left(i, \text{head}(L_i)\right) \right\}.$$

Intuitively, R_i includes those stations that may be interfered by i if i does increase its power. So permissions from these stations should be obtained. Then, a LOCK (c_i, $\{i, \text{head}(L_i)\}$) packet will be sent to each station in R_i, where c_i is to tell the other side the code currently used by station i and set $i, \text{head}(L_i)$ is to indicate the pair of stations that are sending the locking request. On a station $r \in R_i$ receiving the LOCK packet, it replies based on the following rules.

i) GRANT: If $c_i \notin \text{col}[r]$ and ($Z_r = \emptyset$ or $Z_r = \{i, \text{head}(L_i)\}$), a GRANT packet can be returned to station i. The first condition means that there is no collision. The second condition means that station r is currently not locked by any request, or is currently locked by the same pair of stations $\{i, \text{head}(L_i)\}$. Here Z_r is a local variable of station r to register its status. Initially, $Z_r = \emptyset$. After the GRANT packet is sent, station r should update $Z_r = \{i, \text{head}(L_i)\}$ to indicate that it has been locked by the pair $\{i, \text{head}(L_i)\}$.

ii) DELAY: If $c_i \notin \text{col}[r]$ and ($Z_r \neq \emptyset$ and $Z_r \neq \{i, \text{head}(L_i)\}$), a DELAY packet will be replied.

iii) REJECT: If $c_i \in \text{col}[r]$, collision will occur and the locking request will be rejected.

If station i successfully collects a GRANT packet from each station in R_i, it will enter the next "granted" state. If any of the replies is a REJECT, no further power increment is possible and station i will enter the "terminated" state. If the response contains any DELAY packet, station i will go to the "unengaged" state and retry later. In addition, in the latter two cases, an UNLOCK packet will be sent to all stations that have replied a GRANT to station i to release them. On receiving the UNLOCK packet, the receiver r can clear its Z_r to $\emptyset$.

c) *granted* $\Rightarrow$ *confirmed:* After entering the granted state, station i should contact with station head(L_i) for its locking result. A CONFIRM packet will be sent to head(L_i). After sending out the packet, station i should wait for a reply from head(L_i). How station head(L_i) responds is based on the following rules:

i) CONFIRM: if head(L_i) also has received all necessary GRANTs;

ii) DELAY: if head(L_i) has received some DELAYs, but no REJECT;

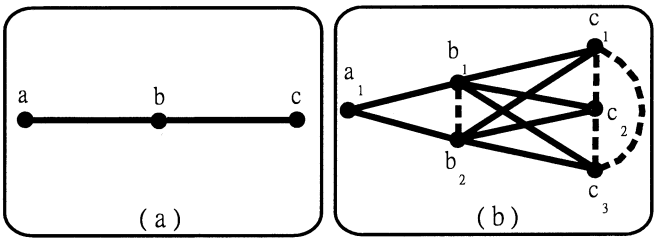

Fig. 5. An example of reduction from a MAPA problem to a SAPA problem.

iii) REJECT: if head(L_i) has received at least one REJECT.

If a CONFIRM packet is received, station i will enter the "confirmed" state. If a DELAY or REJECT packet is received, station i will go back to the "unengaged" state and an UNLOCK packet will be sent to all stations in R_i to release them. Note that in the case of receiving a REJECT packet, station i still has a chance to return to the "unengaged" state [as opposed to the "terminated" state in the earlier case b)] since this merely implies that $head(L_i)$ is unable to increase its power.

d) *confirmed* $\Rightarrow$ *unengaged:* Now everything is ready, and it is safe to increase station i's power. So, we set

$$P_i = \lambda\left(dist\left(i, head(L_i)\right)\right).$$

Then a POWER (P_i) packet is sent to each station in R_i to indicate that station i has increased its power, and station i can delete $head(L_i)$ from L_i. After doing so, station i can return to the "unengaged" state to contend for another power increment opportunity. On receiving the POWER packet, a station $r \in R_i$ should update its collision array to $\text{col}[r] = \text{col}[r] \bigcup \{c_i\}$. This POWER packet also serves as an unlocking message. So each receiving station can clear its Z_r to $\emptyset$.

Finally, we comment that the above discussion has shown a distributed version of the distance-based power adjustment scheme. The result can be easily extended to the degree-based and load-based schemes. Extending these results with code randomization is also straightforward. A station should try to lock necessary stations for their permissions to change its code to avoid primary and secondary collisions.

V. Extensions for MAPA

In MAPA, each station can request more than one code. The purpose is to take the unbalanced traffic loads among stations into account. In this section, we show how to extend our result from SAPA to MAPA.

The main technique is a reduction from a graph representing a SAPA problem to one representing a MAPA problem. Let us represent a PRN topology by an undirected graph $G_m = (V_m, E_m)$, where $V_m = \{H_1, H_2, \ldots, H_n\}$ is the station set and E_m is the link set (subscript m represents "multicode"). In V_m, each station H_i requires t_i codes. Now we translate this problem to a graph $G_s = (V_s, E_s)$ representing a SAPA problem (subscript s represents "single code"). Specifically, for

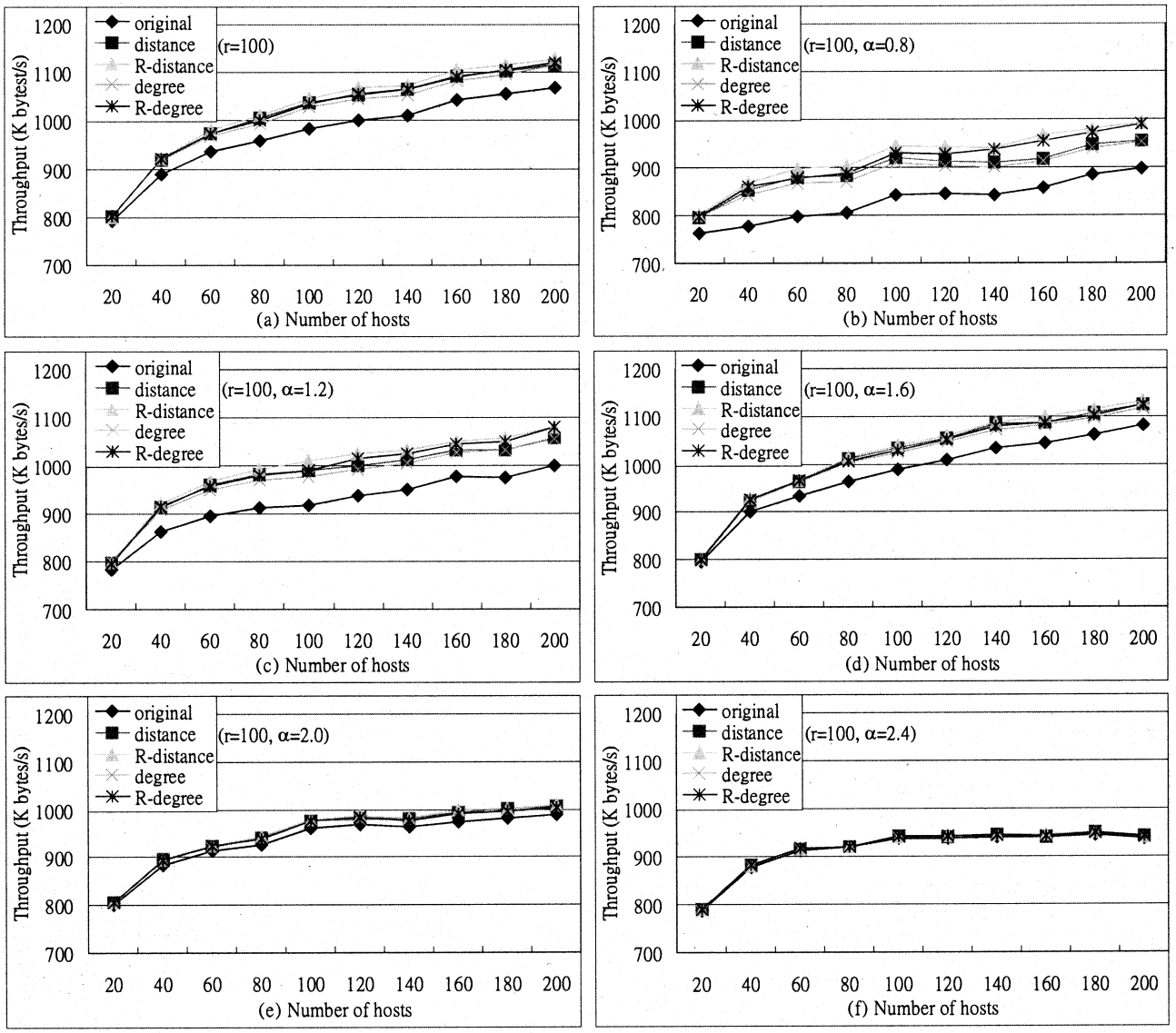

Fig. 6. Throughput versus host density using quality factor: (a) Q_2, (b) Q_1 with $\alpha = 0.8$, (c) Q_1 with $\alpha = 1.2$, (d) Q_1 with $\alpha = 1.6$,, (e) Q_1 with $\alpha = 2.0$, and (f) Q_1 with $\alpha = 2.4$.

each station $H_i \in V_m$, we introduce the following t_i stations into V_s: $H_{i,1}H_{i,2}, \ldots, H_{i,ti}$. Also, the link set is

$$E_s = \{(H_{i,k_1}, H_{i,k_2}) | H_i \in V_m, 1 \le k_1 \le t_i, 1 \le k_2 \le t_i, k_1 \ne k_2\} \cup \{(H_{i,k_1}, H_{j,k_2}) | (H_i, H_j) \in E_m, H_i \in V_m, H_j \in V_m, 1 \le k_1 \le t_i, 1 \le k_2 \le t_j\}.$$

Intuitively, in G_s, each station $H_{i,k}, k = 1, \ldots, t_i$, requires one code. These t_i stations, which represent H_i in G_m, will together require t_i codes. In the definition of E_s, the first set establishes a clique among stations $H_{i,k}, k = 1, \ldots, t_i$, which means that the codes assigned to these t_i stations should be distinct. The second set establishes a link between each pair of H_{i,k_1} and H_{j,k_2}, which indicates the fact that the two vertices H_{i,k_1} and H_{j,k_2} are physically adjacent.

Fig. 5(a) shows an example where we are given a network of three stations a, b, and c requiring 1, 2, and 3 codes, respectively. From these three stations, we introduce three sets: $\{a_1\}$, $\{b_1, b_2\}$, and $\{c_1, c_2, c_3\}$. Each set forms a clique. Also, from any station in one set, there is a link to any station in another set that is originally connected in G_m (e.g., there are six links between $\{b_1, b_2\}$ and (c_1, c_2, c_3). The resulting graph is shown in Fig. 5(b).

Theorem 1: Given a G_m and its corresponding G_s, if a code assignment to the G_s is optimal, the assignment mapped back to G_m is also optimal.

Proof: Suppose that there is an assignment A for G_s that is optimal. It is easy to see that the assignment mapped back to G_m, called A', has no primary and secondary collision. So assignment A' for G_m is correct. Suppose for contradiction that there exists another assignment B for G_m that uses fewer codes than A'. Then we can translate B to an assignment for G_s, called B'. Since B' uses fewer codes than A, we encounter a contradiction, which proves this theorem. □

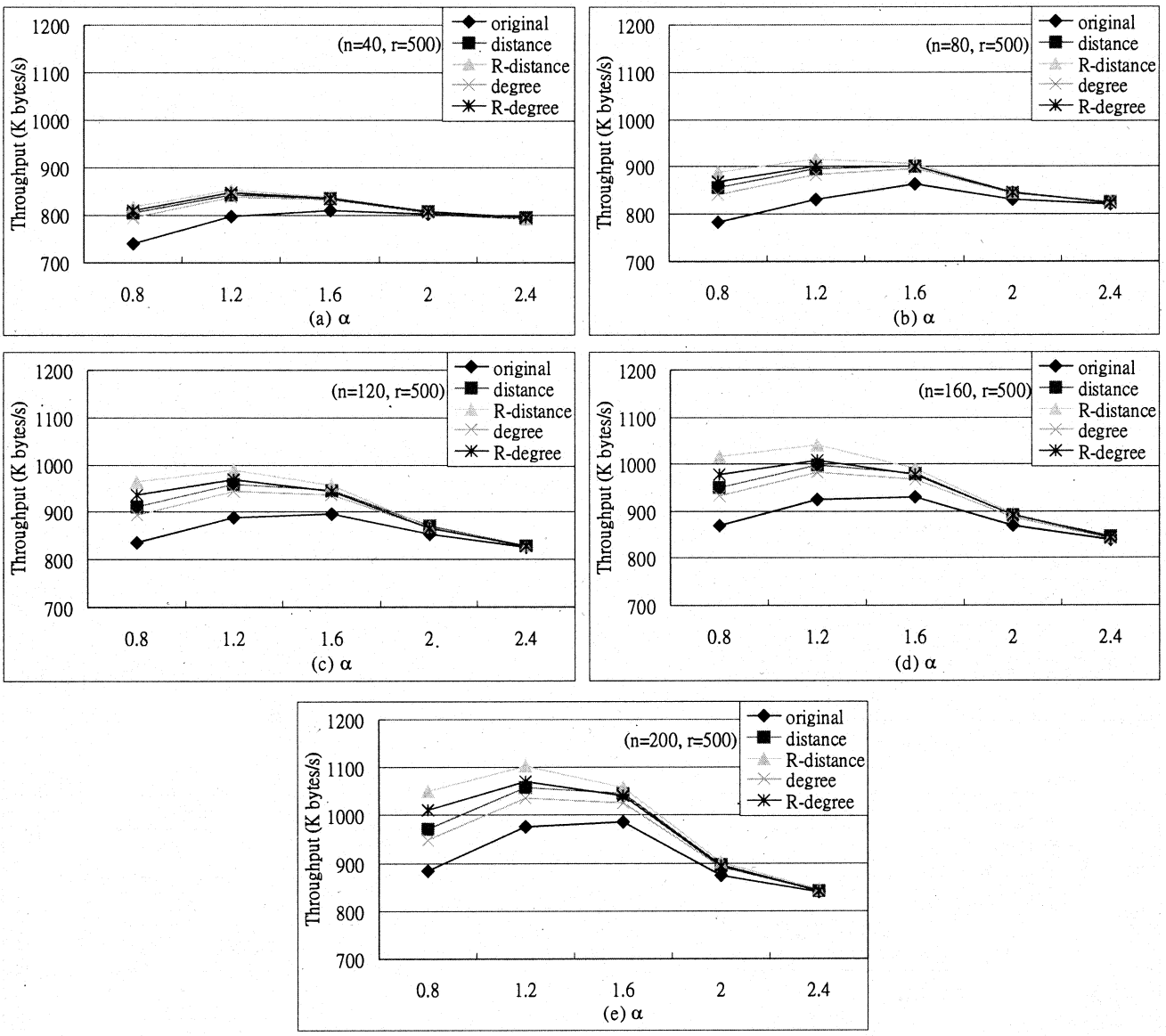

Fig. 7. Througput versus α under various host densities.

With the above reduction, power adjustment on MAPA can proceed similar to that in SAPA, except that when checking primary and secondary collisions, multiple codes may need to be checked for each station.

VI. Simulation Results

In this section, we demonstrate through simulations how power adjustment can improve network connectivity and network throughput. A packet radio network of n stations randomly spread in a 500 $\times$ 500 area is simulated, where n is a controllable parameter. The transmission distance of each station is tunable but no larger than 200 units ($\delta = 20$ units). In the simulations, we adopt the heuristic SATURATION-DEGREE-CODE-ASSIGNMENT [7]. Note that to exclude extreme cases and unfair comparisons, only connected networks are considered. If a network remains unconnected after increasing each host's transmission distance to the maximum of 200 units, we will regenerate a new network.

A TDMA channel model is simulated, where each time slot is 20 μs. The transmission rate is 10 Mbps, so 200 bits can be sent in one slot. Data packets, each of size 2 Kbytes, are injected into the network with a Poisson distribution of an arrival rate r. Each packet has a randomly chosen pair of source and destination. For each packet, the Dijkstra's shortest path algorithm is used to choose routes. Note that unidirectional links are not used for transmitting packets (if they are used, the performance improvement of our schemes could be even better). End-to-end throughput is measured, i.e., a packet successfully traveling from its source to its destination contributes 2 Kbytes to the throughput. All results presented below are from the average of 100 runs, where each run lasts at least 100 s.

A. Effect of Station Density

Fig. 6 shows the network throughput versus station density (with a fixed physical area, a larger n means higher density). In Fig. 6(a), hosts' initial power levels are determined by quality

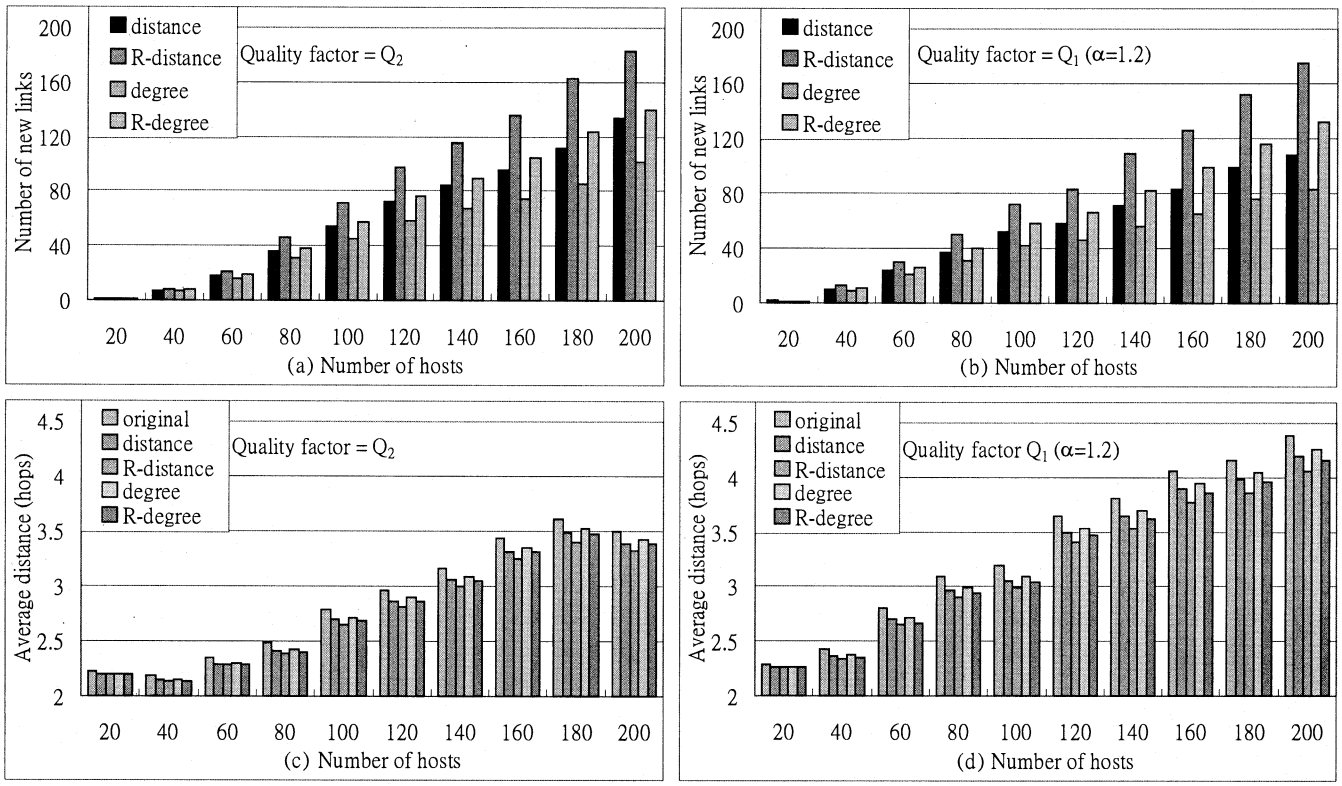

Fig. 8. Improvement in network connectivity: the number of new links and average distance between hosts.

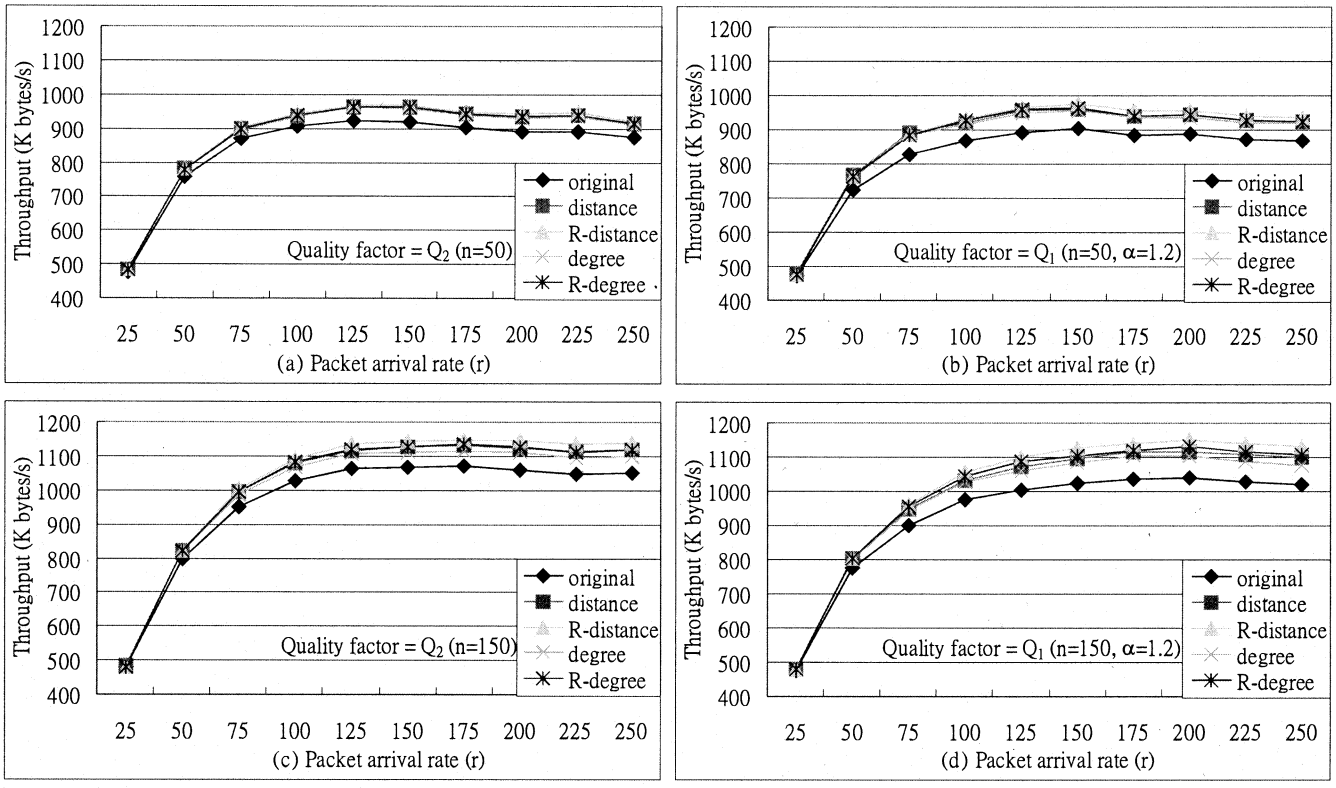

Fig. 9. Throughput versus traffic load.

factor Q_2, while in Fig. 6(b)–(f), they are determined by quality factor Q_1 with $\alpha = 0.8 \sim 2.4$. Note that an initial "R" means that the code randomization mechanism is adopted. Schemes with code randomization perform the best, followed by those

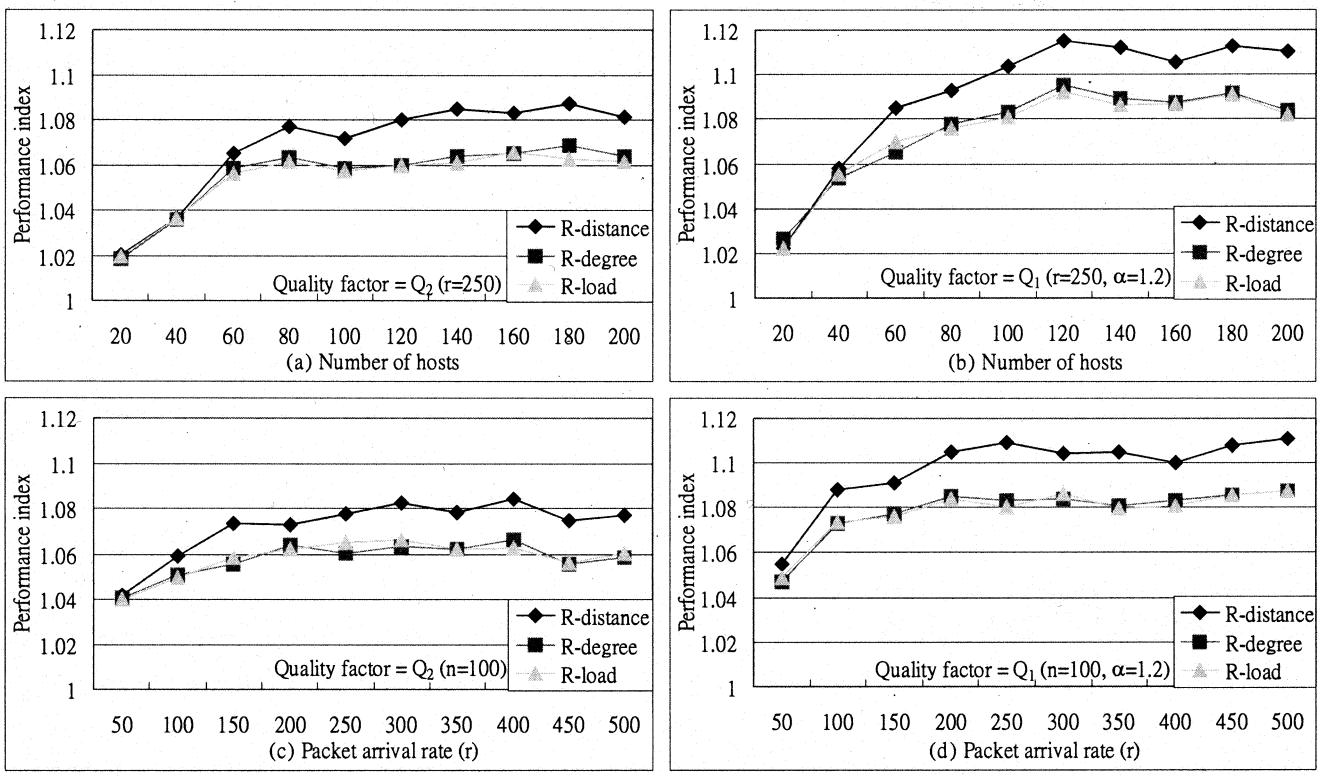

Fig. 10. Comparing the load-based scheme to other schemes: (a)-(b) effect of host density and (c)-(d) effect of traffic load.

without code randomization, and then by those without power adjustment (denoted by "original"). Overall, the R-distance scheme performs the best. Power adjustment is less effective when the network density is low. This is because when the network is sparse, a larger initial power will be picked, leaving less space for power adjustment. After $n \geq 50$, power adjustment can always benefit network throughput. This justifies the value of our schemes. The value of α also affects the result. An α that is too large (e.g., $\geq$2.0) will overemphasize the effect of the average distance between hosts, leading to very large initial transmission power levels of hosts. Again, this will leave little space for power adjustment. The effect of α will be further investigated in the next experiment.

B. Effect of α

Fig. 7 shows the network throughput versus different αs with a higher traffic rate $r = 500$. The observation is slightly different from the previous experiment: α should not be underemphasized or overemphasized. A too-small α will lead to small initial power levels and thus long network diameters. This is harmful to network throughput. Also, with low network connectivity, the total number of codes used is likely small. This implies that power adjustment would be difficult because primary and secondary collisions are likely to occur. Overemphasizing α is unwise too, as discussed earlier. From our experience, $\alpha = 1.2 \sim 1.6$ is likely to give the best performance. Higher traffic load (and similarly higher network density) tends to prefer a slightly larger $alpha$. By properly choosing $alpha$, using quality factor Q_1 is comparable to using Q_2, which is computationally more expensive.

C. Improvement of Network Connectivity

In this section, we are interested in how our schemes can improve network connectivity. In Fig. 8(a) and (b), we measure the numbers of new links being added into the network. In terms of this, the distance-based scheme outperforms the degree-based scheme. In most cases, adopting code randomization can significantly increase the number of new links. As there are more stations, more links can be added, which is reasonable. In Fig. 8(c) and (d), we measure the average distances between stations. We observe that as the network becomes denser, the average distance will increase slightly. Intuitively, in denser networks, a smaller initial power level will be selected so as to reduce the total number of codes used. With our schemes, the newly added links can help reduce the average distance between hosts.

D. Effect of Traffic Load

In this experiment, we try to vary traffic loads to observe how our topology improvement schemes can contribute to network throughput. As can be seen in Fig. 9, significant improvement can be obtained in most ranges of traffic loads. Comparing these figures, two conclusions can be drawn. First, using quality factor Q_1 with $\alpha = 1.2$ is comparable to using quality factor Q_2 under most situations. Second, as n increases, more benefit can be obtained by topology improvement, which implies that our schemes are more useful in denser networks. Note that under very high traffic loads, throughput will somehow get hurt because packets are likely to get dropped in the middle ways to their destinations, thus wasting some bandwidth.

E. Performance of the Load-Based Scheme

In the above simulation, the injected traffic is uniform for all stations. In this section, we repeat the above simulations for the load-based scheme. The traffic model is modified as follows. Each station is assigned a random number between 1 and 5, where a larger number implies a higher load. In our simulations, the probability that a station serves as a source or a destination is proportional to this number. While conducting power adjustment, stations with higher traffic loads are picked for power increase earlier than those with lower loads. Ties are broken based on the distance-based scheme. Fig. 10 shows the result, where the performance index implies the network throughput normalized to that without power adjustment. As can be seen, the load-based scheme is not particularly favorable in terms of performance. In most cases, it performs close to the degree-based scheme. We believe that the main reason is that our simulation concerns end-to-end traffic. Since a routing path typically consists of several relaying stations, favoring only source and destination hosts does not completely reflect the need for relaying stations.

VII. Conclusion

Power control is an important issue in almost all kinds of wireless architectures. We have developed several schemes to improve the topology of a PRN through power adjustment. The results have been successfully applied on top of earlier code assignment solutions. Interestingly, we have demonstrated that although code assignment is a computationally expensive job, it does not prohibit us from improving the performance of a PRN through power adjustment with polynomial costs. In addition, we have also shown how to reduce a multicode assignment problem to a single-code assignment problem and then use the proposed power adjustment schemes to improve the network performance. The temporal variation of traffic loads at individual stations is not considered in this paper. In that case, the code assignment and power levels need to be readjusted from time to time, which deserves further study.

References

[1] N. Abrahmson, "The ALOHA system—another alternative for computer communications," in *Proc. Fall Joint Computer Conf.*, 1970, pp. 281–285.

[2] A. A. Bertossi and M. A. Bonuccelli, "Code assignment for hidden terminal interference avoidance in multihop packet radio networks," *IEEE/ACM Trans. Networking*, vol. 3, no. 4, pp. 441–449, Aug. 1995.

[3] I. Chlamtac and S. Kutten, "Tree-based broadcasting in multihop radio networks," *IEEE Trans. Comput.*, vol. C-36, pp. 1209–1223, Oct. 1987.

[4] I. Cidon and M. Sidi, "Distributed assignment algorithms for multihop packet-radio networks," *IEEE Trans. Comput.*, vol. 38, pp. 1353–1361, Oct. 1989.

[5] L. Hu, "Distributed code assignment for CDMA packet radio networks," *IEEE/ACM Trans. Networking*, vol. 1, pp. 668–677, Dec. 1993.

[6] T. Makansi, "Transmitter-oriented code assignment for multihop radio networks," *IEEE Trans. Commun.*, vol. COM-35, no. 12, pp. 1379–1382, 1987.

[7] R. Battiti, A. A. Bertossi, and M. A. Bonuccelli, "Assigning codes in wireless networks: bounds and scaling properties," *ACM/Baltzer Wireless Networks*, vol. 5, pp. 195–209, 1999.

[8] T. H. Vuong and D. T. Huynh, "Broadcast scheduling in packet radio networks," in *Proc. ICCCN*, 1998.

[9] J. J. Garcia-Luna-Aceves and J. Raju, "Distributed assignment of codes for multihop packet-radio networks," in *Proc. MILCOM*, 1997.

[10] Z. J. Hass, "On the performance of a medium access control scheme for the reconfigurable wireless networks," in *Proc. MILCOM*, 1997.

[11] Z. Tang and J. J. Garcia-Luna-Aceves, "Hop-reservation multiple access (HRMA) for ad-hoc networks," in *Proc. INFOCOM*, 1999.

[12] T. C. Hou and V. O. K. Li, "Transmission range control in multihop packet radio networks," *IEEE Trans. Commun.*, vol. COM-34, pp. 38–44, Jan. 1986.

[13] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Trans. Commun.*, vol. COM-32, pp. 246–257, Mar. 1984.

[14] R. Ramanathan and R. R. Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *Proc. INFOCOM*, 2000.

[15] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. Hawaii Int. Conf. System Sciences*, 2000.

[16] S.-L. Wu, Y.-C. Tseng, and J.-P. Sheu, "Intelligent medium access for mobile ad hoc networks with busy tones and power control," *IEEE J. Select. Areas Commun.*, vol. 18, no. 9, pp. 1647–1657, 2000.

[17] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *Proc. INFOCOM*, 2000.

[18] J.-H. Ryu and D.-H. Cho, "A new routing scheme concerning power-saving in mobile ad-hoc networks," in *Proc. ICC*, 2000.

[19] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *Proc. MobiCom*, 1998, pp. 181–190.

[20] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, pp. 14–25, Jan. 2002.

[21] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proc. INFOCOM*, 2000.

[22] ——, "Algorithms for energy-efficient multicasting in static ad hoc wireless networks," *ACM/Baltzer Mobile Networks Applicat.*, vol. 6, no. 3, pp. 251–263, June 2001.

[23] Y. Wu, Q. Zhang, W. Zhu, and S.-Y. Kung, "Spreading code assignment in an ad hoc DS-CDMA wireless network," in *Proc. ICC*, 2002.

[24] I. Chlamtac and S. S. Pinter, "Distributed nodes organization algorithm for channel access in a multihop dynamic radio network," *IEEE Trans.Comput.*, vol. COM-36, pp. 728–737, 1987.

[25] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11, 1997.

[26] Bluetooth Specification Version 1.1, Feb. 2001.

[27] E. Pagani and G. P. Rossi, "Providing reliable and fault-tolerant broad-cast delivery in mobile ad-hoc networks," *ACM/Baltzer Mobile Networks Applicat.*, vol. 4, pp. 175–192, 1999.

[28] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. MobiCom*, 1999.

[29] C.-J. Su and L. Tassiulas, "Broadcast scheduling for information distribution," in *Proc. INFOCOM*, 1997.

[30] K. Tang and M. Gerla, "MAC layer broadcast support in 802.11 wireless networks," in *Proc. MILCOM*, 2000.

[31] J. Tourrihes, "Robust broadcast: improving the reliability of broadcast transmissions on CSMA/CA," in *Proc. Personal, Indoor and Mobile Radio Communications (PIMRC)*, 1998.

Chi-Fu Huang received the B.S. degree from Feng-Chia University, Taiwan, R.O.C., in 1999 and the M.S. degree from National Central University, Taiway, R.O.C., in 2001, both in computer science. He is currently pursuing the Ph.D. degree at the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan, R.O.C.

His research interests include wireless communication and mobile computing.

Yu-Chee Tseng (S'91–M'95–SM'03) received the B.S. degree from National Taiwan University, Taiwan, R.O.C., in 1985 and the M.S. degree from National Tsing-Hua University, Hsin-Chu, Taiwan, R.O.C., in 1987, both in computer science. He received the Ph.D. degree in computer and information science from The Ohio State University, Columbus, in 1994.

He was with D-LINK Inc. as an Engineer in 1990. From 1994 to 1996, he was an Associate Professor at the Department of Computer Science, Chung-Hua University. He joined the Department of Computer Science and Information Engineering, National Central University, in 1996, where he has been Full Professor since 1999. He was a Program Chair of the Wireless Networks and Mobile Computing Workshop, 2000 and 2001, an Associate Editor of *The Computer Journal*, and a Guest Editor for the *ACM Wireless Networks* special issue on "Advances in Mobile and Wireless Systems." He was a Guest Editor for the *Journal of Internet Technology* special issue on "Wireless Internet: Applications and Systems," Guest Editor for the *Wireless Communications and Mobile Computing* special issue on "Research in Ad Hoc Networking, Smart Sensing, and Pervasive Computing," Editor for the *Journal of Information Science and Engineering*. and Guest Editor for the *Telecommunication Systems* special issue on "Wireless Sensor Networks." His research interests include mobile computing, wireless communication, network security, and parallel and distributed computing.

Mr. Tseng was a Guest Editor for the IEEE TRANSACTIONS ON COMPUTERS special issue on "Wireless Internet." He received the Outstanding Research Award, 2001–2002, from the National Science Council, R.O.C.