On Localized Application-Driven Topology Control for Energy-Efficient Wireless Peer-to-Peer File Sharing

Andrew Ka-Ho Leung, *Student Member*, *IEEE*, and Yu-Kwong Kwok, *Senior Member*, *IEEE*

Abstract—Wireless Peer-to-Peer (P2P) file sharing is widely envisioned as one of the major applications of ad hoc networks in the near future. This trend is largely motivated by the recent advances in high-speed wireless communication technologies and high traffic demand for P2P file sharing applications. To achieve the ambitious goal of realizing a practical wireless P2P network, we need a scalable topology control protocol to solve the neighbor discovery problem and network organization problem. Indeed, we believe that the topology control mechanism should be *application driven* in that we should try to achieve an efficient connectivity among mobile devices in order to better serve the file sharing application. We propose a new protocol, which consists of two components, namely, *Adjacency Set Construction* (ASC) and *Community-Based Asynchronous Wakeup* (CAW). Our proposed protocol is shown to be able to enhance the fairness and provide an incentive mechanism in wireless P2P file sharing applications. It is also capable of increasing the energy efficiency.

Index Terms—Topology control, wireless networking, P2P systems, file sharing, energy efficiency, fairness, incentive, network protocols.

1 Introduction

According to a recent survey [36], Peer-to-Peer (P2P) applications generate 1/5 of the total Internet traffic. It is believed that this trend will continue. Furthermore, an Internet Services Provider (ISP) solution company [32] reported that the hottest P2P applications are file sharing applications such as BitTorrent [3] (which occupies 53 percent of all P2P traffic) and eDonkey2000 [7] (which occupies 24 percent). Apart from these *wired P2P file sharing*, some other *wireless* P2P applications have become part of our daily life. For example, people can now play numerous P2P Java online games, which are compatible with mobile phones so that players are allowed to interconnect in local area through Bluetooth, Wi-Fi, or wide area through 3G W-CDMA or 3.5G HSDPA networks (for example, Nokia 6500s and 6500c, Sony Ericsson W910i, K850i, HTC TyTN II, and so forth) [13]. Indeed, in many metropolitan cities such as Hong Kong and Tokyo, we can see that train commuters routinely play wireless games among each other by using popular devices such as PlayStation Portables (PSPs). Now, many mobile phones also already have 128-Mbyte or higher storage capability. Indeed, it is now a common practice to have P2P file transfer through Bluetooth or Wi-Fi on mobile phones and PDA. As such, wireless P2P file sharing is not only feasible but is also becoming pervasive.

- *A.K.-H. Leung is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Room 807, Chow Yei Ching Building, Pokfulam Road, Hong Kong. E-mail: khleung@eee.hku.hk.*
- *Y.-K. Kwok is with the Electrical and Computer Engineering Department, Colorado State University, Fort Collins, CO 80523-1373. E-mail: Ricky.Kwok@colostate.edu.*

Manuscript received 8 Nov. 2005; revised 13 Sept. 2006; accepted 24 Apr. 2007; published online 14 May 2007.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0326-1105.
Digital Object Identifier no. 10.1109/TMC.2007.1077.

Since such a wireless P2P network is likely to be ad hoc in nature, running P2P applications on top of it requires network developers to meet several research challenges. First, "ad hoc P2P" means that users are allowed to join and leave freely and, hence, a dynamic credit system is needed to monitor the behavior of peers so as to monitor "free riders" [11] who do not contribute to the community. Second, in such an ad hoc P2P wireless environment, energy efficiency is, beyond doubt, a crucial factor in the system design due to the fact that mobile wireless devices are inevitably energy limited. Indeed, it is challenging to tackle the energy efficiency problem of mobile devices in a judicious manner, and it has intrigued researchers for years, for example, [2], [31], and [30]. This motivates the need for a new energy-efficient *topology control (TC)* for effectively supporting P2P file sharing applications.

TC, in a traditional sense, comprises two components: neighbor discovery and network organization [25]. In neighbor discovery, one has to detect network nodes in its proximity and construct a neighbor set in which it could find possible next hops to establish communication linkages. On the other hand, network organization involves the decision of which communication links should be established with neighboring nodes. Typically, it involves the use of power management schemes such as *sleeping* and *transmit power control*. The former disables some communication links temporarily and the latter adjusts the transmission range. In summary, traditionally, the objective of TC is the preservation of network connectivity while improving the efficiency of transmissions. However, we believe that connectivity should be considered at the *application level*. Specifically, our suggested schemes are aimed at achieving an efficient connectivity among mobile devices in order to better serve the file sharing application. Indeed, our idea is that the underlying network-layer (or

1536-1233/08/$25.00 © 2008 IEEE Published by the IEEE CS, CASS, ComSoc, IES, & SPS

even link-layer) connections should be constructed in such a way that the file sharing application's performance is improved. The metric that we use to judge performance is the file request successful ratio.

The remainder of this paper is organized as follows: In Section 2, we discuss the background and related work on P2P networking. In Section 3, we describe our proposed TC protocol. In Section 4, we describe the implementation issues. In Section 5, we describe our simulation methodology. In Section 6, we present our simulation results and our interpretations. Finally, we provide some concluding remarks in Section 7.

2 Related Work

There is a plethora of previous work related to the TC problem.

The first category is TC for ad hoc wireless networks without any P2P concept [4], [5], [25]. The major focal points of these previous results are energy efficiency and connectivity. Some of these previous schemes can generate performance gains in terms of throughput and delay. Only [4] and [5] explicitly take fairness into consideration.

The second category is *wired* P2P, including P2P file sharing protocols. They are generally divided into three classes: centralized P2P file sharing protocols [23], decentralized unstructured P2P file sharing protocols [3], [8], and decentralized structured P2P file sharing protocols [28], [33]. Recently, a proximity-aware configuration of P2P network overlay topology has also been extensively studied [15]. Most of these protocols are designed for file exchange or file searching in a wired network. Specifically, these protocols do not participate in constructing the link-layer topology.

The incentive of sharing files in P2P networks is another important issue. One of the major research directions is the use of game theory to study the incentive problem [1], [12], [20], [38].

Sleeping is found to be a useful method to save the energy of wireless nodes. There are two categories of "sleep-and-wake" protocols suggested in the literature: *scheduled protocol* and *on-demand protocol*. Scheduled protocols include the IEEE 802.11 Power Save Mode (PSM) in infrastructure configuration [14], Power-Aware Multi-Access protocol with Signaling for Ad Hoc Networks (PAMAS) [30], and sensor-medium access control (S-MAC) [37]. Two well-known examples of on-demand protocols are SPAN [5] and ASCENT [4].

Unfortunately, none of the previously suggested protocols are designed for a wireless P2P system. Thus, we consider an *application-level-driven protocol* for mobile wireless P2P file sharing networks. The proposed protocol consists of two components: *Adjacency Set Construction* (ASC) and *Community-Based Asynchronous Wakeup* (CAW). ASC determines which neighboring nodes should be included in the "adjacency set." CAW groups users[1] into "communities." Members of the same community employ the same sleep-and-wake schedule.

1. We use "users," "mobile devices," and "nodes" interchangeably to mean wireless mobile devices.

3 Our Proposed Protocol

In this section, we describe in detail our proposed algorithms and their design rationale. Rather than suggesting a new routing protocol concerned about finding a path between a sender and receiver, we focus on the question of "*who is my neighbor?*" and the question of "*when should i sleep and wake up?*" for each individual node. Indeed, our proposed TC schemes can be used with any typical ad hoc routing algorithms.

3.1 Overview

As indicated in our brief survey presented in Section 2, many energy-efficient protocols have been proposed in the literature. Nevertheless, previous researchers usually neglect the importance of considering the *difference* of remaining energy levels between *individual nodes*. Indeed, in the pioneering work of Singh et al. [31], it only uses metrics for the total energy consumption in the whole path from server-peer (the peer who shares files) to client-peer (the peer who requests files).

In *ASC*, we find out which nodes could be the *next hop* when a node has to communicate with another node which is n hops away ($n \geq 2$). Specifically, we consider that the construction of neighbor set is related to

1. contribution levels of different nodes in the P2P file sharing network,
2. the popularity of file resources owned by individual nodes,
3. aggressiveness of the file-requesting node, and
4. the remaining energy levels of nodes.

Our protocol not only takes energy efficiency into consideration but also controls the topology of the file sharing network to introduce fairness.

In *CAW*, we form "virtual communities" among mobile users. We define *community* as a set of two or more mobile users that perform in a particular habit. For example, in a music file sharing network, users with similar preferences and fans of similar idols are recognized as members of the same "community." Members from the same community follow the same wakeup schedules. The rationale behind this is that file sharing is usually carried out between users with similar interests (this is the usual reason that a sender owns the favorite file that the requester is asking for). Using community formation, we not only increase the chance of getting a file but also allow a group of nodes to sleep and conserve energy when other communities are active.

3.2 "Who Is My Neighbor?"—ASC

Although the main function of routing protocols is to find out the appropriate path from the sender to the receiver, ASC constructs the next-hop set for each individual user locally and gives this information to the routing layer to construct the path. No matter which routing protocol is used, the question is "Why does someone need to help the source node to route the packets?" Our view is that, by taking some factors into consideration, one can decide the worthiness of helping others to route their packets. Thus, we propose *ASC* to address the issue. In the following, we first describe the system model.

Consider a P2P file sharing application running on top of a mobile wireless ad hoc network with N users. We denote each user by u_i and the set of users by U:

$$U = \{u_i | i = 1, 2, 3, \ldots, N\}. \quad (1)$$

We assume that there are in total M different file objects in the network:

$$F = \{f_j | j = 1, 2, 3, \ldots, M\}. \quad (2)$$

We use E_i to denote the remaining energy level (battery level) of a mobile user u_i. We then use a binary vector V_i to denote the file objects that user u_i possesses:

$$V_i = \{\delta_{ik} | k = 1, 2, 3, \ldots, v\},\ i = 1, 2, 3, \ldots, N, v = M, \quad (3)$$

where

$$\delta_{ik} = \begin{cases} 1 & \text{If user } i \text{ has file object } f_k \\ 0 & \text{If user } i \text{ does not have file object } f_k. \end{cases} \quad (4)$$

In this P2P file sharing platform, we quantify the popularity of each file object and represent it by using a weight with respect to a particular user. Different files are of different levels of popularity in the network: Some file objects (for example, latest pop music) are of higher ranking in the mind of some users. Thus, we represent the weight (rank) of an object f_k in user u_i's mind by a weight w_{ik}. Different file objects could have different weight values for different people.

Traditionally, network nodes transmit their packets to the "neighboring nodes," which are those nodes in close proximity of the source node. However, our idea is that even if a node, say, u_i, is within the transmission range of the source node u_s, it is not a "must" that u_i needs to be the next hop of u_s. Specifically, from an application point of view, the neighbor node u_i might refuse to be the next-hop data forwarder for u_s.

In ASC, we define the *adjacency set*, $\mathrm{Adj}(s)$, which represents the next-hop nodes of source node u_s, as:

$$\mathrm{Adj}(s) = \{u_i \mid d_{u_s \to u_i} \leq R \wedge u_i \text{ is willing to be the next hop of } u_s\},$$

where $d_{u_s \to u_i}$ denotes the distance between u_i and u_s, and R is the transmission range of u_s.

Now, an interesting question is: "*Is u_i willing to be the next-hop of u_s?*" To answer this question, u_i needs to consider several factors. The first factor is *fairness*. If u_i has already helped a lot in routing packets for other nodes and contributed in the maintenance of the P2P network for a long time, then it is desirable to exclude u_i in the adjacency set of u_s so that the relaying and routing task can be more evenly distributed among all nodes. The second factor is the *aggressiveness* of the source node u_s. If u_s always asks for files but does not contribute anything, then u_s is likely to be a selfish user, and this would reduce the willingness of other nodes to relay u_s's packets.

The third factor is the *popularity of file objects* held by u_s. Apart from the aggressiveness of u_s, u_i also needs to consider the expected number of files that *will be downloaded from* u_s. If u_s holds a large number of files or a smaller number of popular files, then there can be many other peers asking for files from u_s. Consequently, this would not only occupy u_s's bandwidth, but also create a burden for the neighbor(s) of u_s. The reason is that a neighboring node has to frequently route its packets outward. Moreover, being this kind of relay node requires a double expense on energy: A relay has to first receive files from the server-peer, then transmit the files to the client-peer, and both transmit and receive actions consume energy, whereas the server-node only transmits files and the client-node only receives files.

The final factor considered in our study is the *remaining energy levels of nodes*. As mentioned above, being a relay node requires more energy consumption than simply being a client-peer (receives files) or server-peer (transmits files). It is obviously undesirable to ask a peer with a very low remaining energy level to be the relay node. Furthermore, it is not enough to define a single threshold so that all peers with energy levels above the threshold act as relay nodes and those with low energy levels only enjoy service from the high-energy peers, which is simply unfair. We thus propose the use of a relative metric; that is, if a node u_i has a higher energy level than a node u_s, then we consider that u_i is more eligible to be the next hop for u_s and it is also more desirable for u_i to relay packets from u_s.

Now, we can formalize the above factors and define *preference metrics* to evaluate the worthiness of being an intermediate node to relay packets for the source node. We classify network activities into two types. The first type is "public service," where a node exchanges control packets with other nodes for maintaining network infrastructure, handling file searching query, helping in transferring files, and so forth. The second type is "private service," where a node downloads files from other nodes for its own use.

The contribution metric is defined as

$$m_{contrb} = \frac{T^i_{public}}{T^i}, \quad (5)$$

where T^i_{public} denotes the time duration in which node u_i is engaging in public service and T^i is the time duration for which u_i has joined this P2P network as one of the peers. Given this metric, we can allocate the relaying task more evenly among all nodes. Nodes that have already provided public service for a long period of time can be excluded from the adjacency set. As we have mentioned, we use "public service" and "private service" to classify network activities.

To assess the aggressiveness of u_s, we define the metric

$$m_{agg} = \frac{T^s_{private}}{T^i}, \quad (6)$$

where $T^s_{private}$ denotes the time duration in which u_s is engaging in private service. These two metrics use the behavior of u_i and u_s known so far to predict the upcoming traffic demand of u_i and u_s. The service time can be easily recorded by a local timer. As time goes by, these metrics would be changed. Therefore, our protocol would update these values in a distributed manner to reconstruct the adjacency set of u_s dynamically.

In a file sharing network, one of the most valuable resources is the file objects owned by network users. Some files are more popular and are requested more frequently by users. Indeed, work has been done on studying the relationship between the number of requests or number of replica of a particular file in a P2P network and the popularity of files [11]. Assume that we can give a rank r to

$$\log(prob(e)) = -\kappa \log(r) + \log(K)$$

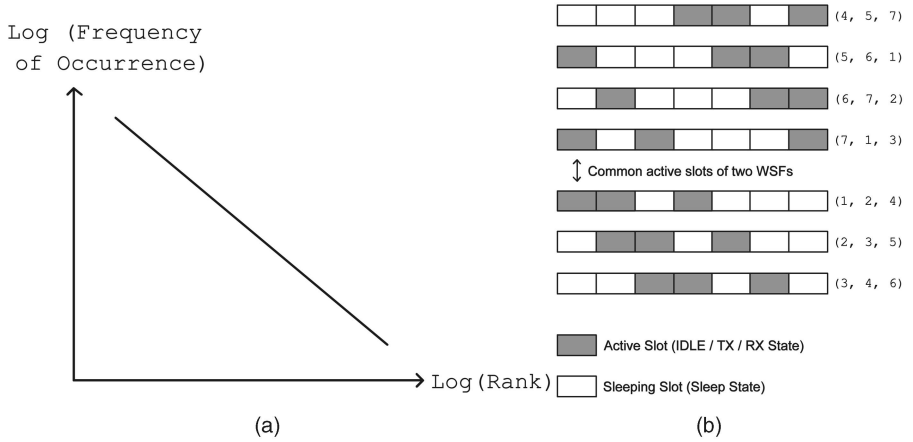


Fig. 1. Zipf's law and "(7, 3, 1) design" of the Wakeup Schedule Function (WSF). (a) Zipf's law. (b) "(7, 3, 1) design" of WSF.

a file f to represent its popularity. Then, according to the Zipf distribution [19],

$$prob(f) = \frac{K}{r^{\kappa}} \tag{7}$$

for some constant K and κ is close to unity, which means that "the probability that a file searching query is associated with the rth most requested file is inversely proportional to its rank value." For example, the most popular file $(\text{rank} = 1)$ would be the file most probably requested since the denominator of the fraction in (7) would be of the smallest value. This is known as the *Zipf's law*: The probability of occurrence of an event e (denoted by $prob(e)$) as a function of its rank r (determined by the frequency of occurrence) is a power-law function (see Fig. 1).

Here, we assume that κ equals 1. Our protocol uses the basic form of Zipf's law. If a more accurate value of κ is provided by network statistics, then the accuracy of CAW can be further improved:

$$\log(prob(e)) = -\kappa \log(r) + \log(K). \tag{8}$$

Based on our assumption that κ equals 1, we estimate that, if a user holds a larger number of popular files, then this peer is more likely to be a server-peer. Consequently, more energy would be required for being the relay node of this node. Let r_i be the average rank of the file objects held by user (node) u_i:

$$r_i = \frac{\sum_{j=1}^{J} r'(f_j)}{J}, \tag{9}$$

where $r'(f_j)$ denotes the rank of file f_j and J is the total number of files that user u_s holds.

We then define the *File Popularity Metric* with respect to user u_s as

$$\frac{K}{r_s^{\kappa}}. \tag{10}$$

If $\kappa = 1$, then the value of K can be determined as follows: We assume that there are in total M files in the P2P file sharing network. Then,

$$\begin{aligned} &\textstyle\sum_{m=1}^{M} prob(f_m) = 1 \\ &\Rightarrow \textstyle\sum_{m=1}^{M} \frac{K}{m^{\kappa}} = 1 \\ &\Rightarrow K = \frac{1}{\sum_{m=1}^{M} \frac{1}{m^{\kappa}}}. \end{aligned} \tag{11}$$

Thus, the File Popularity Metric can be defined as

$$m_{popu} = \frac{1/\sum_{m=1}^{M} \frac{1}{m^{\kappa}}}{r_s^{\kappa}}. \tag{12}$$

This probability is calculated based on the average rank of files held by u_s and it serves as a measure on the chance that a file searching query is associated with u_s.

Finally, we define the energy metric as

$$m_{eng} = \frac{E_i}{E_s}, \tag{13}$$

where E_i and E_s denote the remaining energy levels of u_i and u_s, respectively. A larger value of the expression represents a higher preference for u_i to act as the next hop of u_s.

A larger value of the aggressiveness metric means that the source node is likely to be a free rider and does not deserve other nodes to relay too much traffic for it. Finally, a larger value of the File Popularity Metric means that the source node has a higher expected level of outgoing traffic. All these three cases reduce the intention of other nodes to be the relay node. On the other hand, a higher energy level of u_i makes it preferentially the next hop of u_s.

Now, we refine the definition of adjacency set as

$$\text{Adj}(s) = \left\{ u_i \mid d_{u_s \to u_i} \le R \wedge \left\{ \begin{array}{l} m_{contrb} < \lambda_1, \text{or} \\ m_{agg} < \lambda_2, \text{or} \\ m_{popu} < \lambda_3, \text{or} \\ m_{eng} > \lambda_4 \end{array} \right\} \right., \tag{14}$$

where λ_1, λ_2, λ_3, and λ_4 are the threshold values used to determine whether u_i should be put into the adjacency set of u_s or not. The four metrics are *not* used simultaneously. In this paper, we study the effect of each of them separately by assuming that *all* nodes in the network use one of the four metrics in the ASC.

3.3 "When to Sleep and Wake Up"—CAW

We now describe the second component of our proposed TC scheme, namely, *CAW*. This is a sleeping protocol belonging to the asynchronous scheduling category (refer to the *Related Work* section for a classification) and using the cross-layer design concept. With knowledge from the application layer, we form "virtual communities" among mobile users. We define *community* as a set of two or more mobile users that perform in a particular habit.

The system model in CAW is similar to that in ASC. However, now, we further assume that those files are music files (one of the most common types of files being shared in P2P network nowadays), and we further assume that those music files are songs sung by different singers. Again, consider a number of music files circulated in a network, with a set of N users $U = \{u_n, 0 \leq i \leq N\}$. The singers construct a singer set:

$$S = \{s_k, 0 \leq k \leq Z\}, \tag{15}$$

where Z is an integer. We employ the *similarity measurement*, which is widely used in the field of *Information Retrieval (IR)* [29] to form communities.

We compare the similarity *between any two users* u_i *and* u_j based on the number of songs that they own for different singers. If two users both have many songs by singer A, then this increases the level of similarity between them. Of course, we reduce the weighting of a singer X if this singer is very popular, and nearly all users have his or her song. In this case, "both having songs by singer X" is not considered as an important feature to show that two users are similar. This phenomenon is similar to the traditional concept in IR where frequent terms are regarded as "less important" in the weighting function.

In summary, we define the *importance* q_{ik} of a singer s_k with respect to user u_i as [29]

$$q_{ik} = n_{ik}[\log_2(N) - \log_2(n_k^{users}) + 1], \tag{16}$$

where n_{ik} is the total number of songs by singer s_k found in user u_i's disk. n_k^{users} is the number of users that own songs by singer s_k. Given that there are in total K singers under consideration, the selection of these K singers should be diversified enough to reflect the interest of different users. The selection of the value of K also depends on the computation power of mobile devices. Using the K importance values, we can construct a K-dimensional vector for each user. This vector is a *Preference Vector* representing the user's interest:

$$Q_{pref} = \{q_{i1}, q_{i2}, , q_{iK}\}. \tag{17}$$

The *similarity* $SIM_{users}(i,j)$ between two users u_i and u_j in the P2P file sharing network is again defined as follows (normalized Jaccard coefficient)[2] [29]:

$$SIM_{users}(i,j) = \frac{\sum_{k=1}^{K}(q_{ik}q_{jk})}{\sum_{k=1}^{K} q_{ik}^2 + \sum_{k=1}^{K} q_{jk}^2 - \sum_{k=1}^{K}(q_{ik}q_{jk})}. \tag{18}$$

2. Our definition of a Jaccard coefficient has square terms in the denominator, which is a little bit different from that defined in [29] because we normalize the coefficient to be a value between 0 and 1. For details, please refer to http://mingo.info-science.uiowa.edu/padmini/230/Lectures/Vector1.html.

In CAW, users belong to a community C if their level of similarity is higher than a certain threshold SIM_{th}. We emphasize that, after joining one community, a user would not join another community. Each member of the community would have the responsibility to "promote" this community. The key concept of CAW is that members of the same community would employ the *same* wakeup schedule. The rationale behind our design is that we group users together and thus let them sleep and wake together to increase the chance of successful file exchange, conserve energy, and minimize interference by offsetting active periods of users from different communities.

A requirement of using wakeup scheduling such as the IEEE 802.11 PSM and S-MAC is that all users need to synchronize with each other. This is a complex task when the scale of network is getting much larger. To avoid this problem, Zheng et al. [39] propose the use of the Asynchronous Wakeup Protocol (AWP). The main feature of their design is that, using the theory in *Block Design Combinatorics*, they employ *WSF*, which guarantees that mobile nodes using different schedules must have their active period overlapped in at least one time slot. For details, readers are suggested to consult [39]. Let us quote the result as follows, using a so-called "(7, 3, 1) design."

Given that different nodes' clocks are not synchronized, start at a random time instant. Consider a total number of seven time slots, each with a fixed duration of I, summed up to a time frame of length T. If each node wakes up and becomes active in three of the time slots, namely, the xth, yth, and zth time slots, respectively (x, y, and z between 1 and 7), then they must have at least one time slot overlapped, even drifted, provided that (x, y, z) is one of the elements in this set (see Fig. 1):

$$\begin{aligned}\{&(1,2,4);(2,3,5);(3,4,6);\\ &(4,5,7);(5,6,1);(6,7,2);(7,1,3)\}.\end{aligned} \tag{19}$$

Based on this "(7, 3, 1) design," together with a neighbor discovery beacon protocol given in [39], any two neighbors must be able to find each other. In CAW, members of the same community would use the same WSF and become active or return to sleep "together." Of course, we do *not* rule out the possibility that a user is interested in a file that is outside his or her community, but the use of WSF in AWP such as the (7, 3, 1) design mentioned above already solves the problem. Using this wakeup schedule, network neighbors, no matter which communities they are in and what sleep and wake schedules they use (from the set shown in (19)), are guaranteed to hear each other within one time frame (seven time slots). The only problem is that users from different communities would notice the existence of each other in a larger delay since they do not employ exactly the same schedule. For example, in the (7, 3, 1) design, the worst case is that, if two communities overlap in the last time slot of a time frame (for example, WSF (6, 7, 2) and (7, 1, 3)), then, in every time frame, the packets would have to be buffered until the last time slot comes. The delay would become longer. However, the network originally connected would still *not* be partitioned.

To summarize, CAW is built on top of AWP by adding the concept of "community." The purpose of grouping users into communities is twofold. The first is to allow users with similar interests to be "active together." The reason is that

we believe that users mostly decide to communicate with users who have similar preferences in terms of file sharing. By grouping them into virtual communities, we tend to increase their chance of getting desired files. Second, this also reduces the delay. When one group of users is operating, other groups that are not interested in their file transfer would simply get into sleep mode.

Our TC mechanism is a protocol running in between the network layer and MAC layer. From this layering perspective, our proposed TC mechanism is similar to SPAN [5], which is also a TC protocol, but it only uses sleeping to conserve network nodes' energy.

One more point to note is that our protocol is mainly designed for mobile users in application scenarios such as people hanging out in a shopping mall or walking down the street. We do not target at higher mobility scenarios because, in those cases, current technologies such as 2.5 generation (2.5G) GPRS, the Enhanced Data GSM Environment (EDGE), or the third-generation Universal Mobile Telecommunications System (3G UMTS) could provide file download from mobile service operators through base stations. Instead, our protocol targets at file sharing in localized communities with pedestrian speed mobility.

4 Implementation Issues

In this section, we describe the issues pertaining to the implementation of our proposed TC mechanism.

4.1 ASC

In ASC, in order to construct the next-hop set (that is, the *adjacency set*) of each node, the nodes need to exchange a small amount of information to calculate the value of the preference metrics. This small amount of information can be incorporated into the request-to-send/clear-to-send (RTS/CTS) packet exchange so that extra control packet is not needed. This piggyback method is used in Muqattash and Krunz's distributed power control protocol [22]. The "connectivity set" in their protocol is similar to the adjacency set in ASC, but ASC unifies incentive, fairness, and energy-awareness factors and is tailor-made for file sharing applications. Their pioneering exploration reveals that a distributed algorithm with information exchange among peers is practical and does not have any significant negative impact on energy efficiency due to the energy spent on the transmission of a small amount of control overhead (relative to the file size, especially in ASC for P2P file sharing applications). As suggested by Muqattash and Krunz, when an RTS/CTS exchange is not frequent enough to update the connectivity, HELLO packets can be used to exchange information for the construction of an adjacency set.

One practical issue that has to be addressed is the trustworthiness of nodes. For example, how can we ensure that the timing values and energy-level values given by each node are real and obtain true values for the four metrics? First, it should be the devices to report these values, not manually. Provided that an ordinary user cannot hack into the kernel of the operating system of the devices, these values should be real. However, this security issue is not our main goal in this paper. Second, there is a field of research on the "reputation" and trustworthiness of nodes [10], [35]. We believe that it is possible to adopt the proposed scheme(s) to address the issue. After getting the required information (T^i_{public}, $T^s_{private}$, E_i, E_s, and so forth), each node determines a preference metric value for the nodes that it can connect to and lets these nodes know if it is willing to be their next hop.

Our metric's "rank" of file r_i could be based on Internet statistics. In our paper, we keep this open and do not restrict the method to obtain r_i: It could be estimated previously in wired Internet file transfer statistics, hit rate, download rate, or pop music ranking available on the Web according to Internet statistics. The key point is that our expression comes from Zipf's law, which employs the concept of "frequency of request."

4.2 CAW

The first step of CAW is the formation of communities. Similar to many scheduling protocols aforementioned (for example, S-MAC), a user first listens for a certain amount of time, which depends on the density of nodes, applications running on top of the network, and so forth. After this time is expired, an announcement message is broadcast (if this user does not receive any other announcement message in the listening period). This user is similar to the "synchronizer" in S-MAC. The message contains the preference vector described in (17), showing the importance of different singers with respect to this user. The message also contains the WSF that it decides to use. Other users receiving this message would be able to determine the level of similarity between itself and the sender of the message. If the value of the Jaccard coefficient for them is higher than a certain threshold, then they are regarded as similar to each other and grouped into the same community, and the receiver of this announcement message would employ the same WSF. At this point, the community has two members. Later on, the new member would rebroadcast the announcement message to let other nodes compare their preference vectors with that of the original proposer of the community. Otherwise, if the receiver finds that the similarity is lower than the threshold, then it continues to listen until it finds a community to join. If it cannot join a community after its listening period, then it sends an announcement message itself and asks any other members to join. To reduce contention, listening periods of different nodes are random values ranging from 0 to $T_{announce}$.

Each user would join only one group. The key concept is that we assume that the "preference" of a user would not change very fast although the number of files owned by users keeps changing (preference by nature only changes slowly and gradually relative to the mobility of a user). This is different from the concept in S-MAC but can still solve the "border nodes problem." Nodes within the range of more than one user of similar interest but trying to establish communities independently simply pick up one to join.

A user may have similar preferences with more than one neighbor, but the point is that our calculation of preference vectors already takes into account all file objects held by the users: The community formation is not done by asking the user to select a community manually (for example, ask the user, "which singers do you like most?") but is instead done by computation of the similarity coefficient. Two users are classified into the same community if their file objects (even if they have a number of favorite singers) are regarded as

"similar," that is, a Jaccard coefficient value higher than a certain threshold.

For each community, an announcement message would be resent by the members every c time frames. c is a random value smaller than C, where C is predefined. This is to allow members to "promote" their community so that new nodes entering the network can join their corresponding communities. The second purpose is to adapt to network dynamics. Since all nodes are moving, members originally belonging to the same community and using the same WSF may leave each other when moving. Therefore, after r time frames, not only would new nodes find a community to join, but old members would also need to check if they can still hear the same WSF broadcasting in its proximity. Consequently, a node would decide to rejoin other communities showing similar interest.

In fact, each user needs to calculate their preference vector before joining the network. In daily life, users always download music into their MP3 player or PDA from their personal computer (PC) before they go out. Now, our protocol needs only one more step: Calculate the preference vector described in (17) by using the PC at home and download it to the mobile device together with the music files. It should be noted that this calculation can be done by a PC; therefore, it does not increase the computation overhead needed on the energy-constrained mobile devices with limited computation power. The number of users who own files of a singer can be estimated by Internet usage statistics. A similar phenomenon is singers' ranking. This is a very common statistic: Music Web sites could provide this information estimated from the number of downloads of different songs to ease the calculation of preference vector and Jaccard coefficient.

Upon receiving other users' preference vectors, the similarity-level calculation done on each mobile device is finite addition and subtraction only, which does not cause feasibility problems. The final point to note is that, once a file transfer is started, the nodes overwrite their sleeping schedule and stay at the active state from that point onward until the file transfer is finished.

5 Simulation Configurations

We study the performance of the proposed protocols by using simulations. First, we describe the parameters used in our simulations. The simulator used in this paper is developed on Matlab. Some famous communication systems used in the industry also have their simulation platforms constructed on top of Matlab/Simulink such as [34].

The traffic pattern of the system is simulated with the use of interrequest arrival times. We assume that the calls arrive according to a Poisson distribution, where the call arrival times and the interarrival times between calls are mutually independent. This implies that the interarrival times τ are exponentially distributed. Our mobility model is based on a random waypoint model and assumes that users' velocities follow a Gaussian distribution [21] with $\text{mean} = 3\text{-}5\ \text{km/hr}$ (that is, 0.83-1.11 m/s) and $\text{variance} = 0.54$. This simulates the movement of pedestrians.

In previous sections, we define a weight matrix for different objects, showing the weights of different files in each user's mind. In the simulations, we also define an *Aggressiveness Matrix* for peers, listing the relative frequency of different users to request files. "Who requested a file" and "what file object is being requested" are decided according to these two matrices. It is *not* a random generation. We use the Zipf distribution for generating target files according to the popularity of files and use *Roulette's Wheel Selection* for generating the requests. Roulette's Wheel Selection is commonly used in *Evolutionary Algorithms*: Its details can be found in [6].

In CAW, the mobile devices are assumed to have four possible modes of operation: Transmit, Receive, Idle, and Sleep. In ASC, no sleeping is adopted. The energy consumption ratio of the four modes is set as $1 : 0.6 : 0.5 : 0.08$, as indicated in [24] and [9]. The energy consumption on a node is modeled as

$$P_{\text{Tx}}T_{\text{Tx}} + P_{\text{Rx}}T_{\text{Rx}} + P_{\text{IDLE}}T_{\text{IDLE}} + P_{\text{sleep}}T_{\text{sleep}},$$

where the four P terms represent the power consumption in Transmit, Receive, Idle, and Sleep modes, respectively, and the T terms represent the corresponding time durations that the mobile devices are in different modes.

For path loss in radio propagation, we adopt the *Okumura-Hata Model*, which is commonly used in the literature [26] to estimate the path loss.

In the simulations for ASC, 50 mobile devices are assumed to be scattered in different locations within around a 300 m $\times$ 300 m area and are allowed to move freely according to our mobility model. The devices are modeled to have similar wireless transmission parameters as those commonly found in the IEEE 802.11b wireless local area network (WLAN) adapters available in the market, which operate at the 2.4 GHz Industry, Science, Medicine (ISM) band, where the transmit power generally ranges from 10 dBm to 30 dBm, depending on the brand of the WLAN adapter. We set the transmit power as 14 dBm in our simulations.

The sizes of file objects are assumed to be less than 5 Mbytes. File objects with these sizes can be MP3 songs, ring tones for mobile phones, short movie trailers in relatively low resolution, and so forth. We assume a file searching engine that can search the requested file, as long as there exists a path between the holder of file *in active states* and the requester, according to the adjacency set of all nodes. The values of λ_1, λ_2, λ_3, and λ_4 are set as 0.250, 0.002, 0.016, and 1.500.

Here, we set up a benchmark for comparison to study the performance of ASC. Consider a P2P system again. The simplest way for a node to construct its next-hop set is to include every node within its transmission range. This scenario is the *benchmark* that we use to compare with a network running our proposed TC algorithm.

In the simulations for CAW, the simulation parameters setting is basically the same as that in ASC, except that the number of users and file objects are different. There are six different singers' songs owned by 100 users, and totally, there are 60 file objects. The threshold value of the Jaccard coefficient used in comparing the similarity levels of users to make a community-joint decision is set as 0.75.

There are two benchmarks for evaluating CAW. First, since our CAW is inspired by AWP [39], we compare the

TABLE 1
Important Simulation Parameters Used for ASC

Simulation area	300 m × 300 m
Simulation period	4000 seconds
Number of nodes	50
Number of distinct files	30
Mean velocity of nodes	0.83–1.11 ms^{-1}
Mean inter-request arrival time	100 seconds
Transmission radius of each mobile node	100 m
Transmission rate of mobile nodes	0.6 Mbps
Adjacency set update period of each node	300 seconds
$\lambda_1, \lambda_2, \lambda_3, \lambda_4$	0.250, 0.002, 0.016, 1.500

TABLE 2
Important Simulation Parameters Used for CAW

Simulation area	300 m × 300 m
Simulation period of each simulation	3000 seconds
Number of nodes	100
Number of distinct files	60
Mean velocity of nodes	0.83–1.11 ms^{-1}
Mean inter-request arrival time	100 seconds
Transmission radius of each mobile node	100 m
Transmission rate of mobile nodes	1 Mbps
Community update period of each node	300 seconds
Similarity threshold, SIM_{th}	0.75

performance of CAW with AWP. Second, we compare the case when CAW is used and the case when no sleeping protocol is used. In both cases, the asynchronous wakeup schedule is assumed to be a (7, 3, 1) design, which is originally used in AWP. The key simulation parameters are shown in Tables 1 and 2.

6 Performance Results

We consider several performance metrics, namely, file request successful ratio, delay performance of the file transfer, average individual contribution of each node (that is, the average amount of time that each node spends on being a sever or relay), and standard deviation (SD) of energy levels of node, which is used to find out whether the difference between energy levels of different nodes increases during file transfer: We interpret it as another aspect of the fairness measurement. For CAW, we also study the average values of the Jaccard coefficient and spatial density of the community.

More than 10 simulation trials are carried out for each test case. Since randomness is involved in the mobility model and the file request model is introduced above, data values obtained for each simulation are not exactly the same, even if the same setting is used. Thus, we show the maximum, minimum, and mean values for the file request successful ratio and delay performance obtained. Unless specified, the data values shown for a performance metric are mean values, computed with a confidence interval of 95 percent.

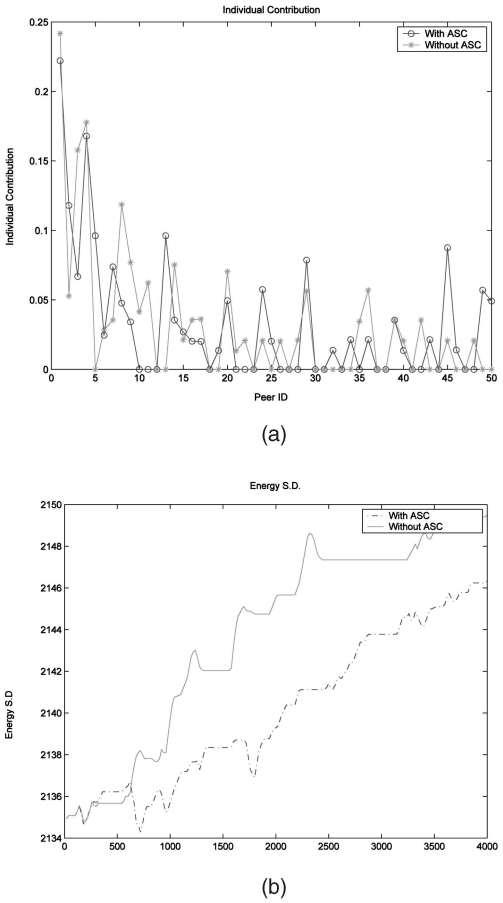


Fig. 2. Performance results for ASC: contribution and popularity metrics. (a) Individual contribution for ASC: contribution metric. (b) Energy SD for ASC: popularity metric.

6.1 Results for ASC

First, we consider the energy consumption for control purpose. We find that the amount of energy used for building the adjacency set using ASC equals only about 1 percent of the total energy consumption of nodes (see Fig. 6a). We would like to point out that, whether ASC is used or not, neighbor discovery and neighbor set construction are needed for an ad hoc network to make sense. The use of ASC does not involve a large amount of extra energy for communication between peers. The curve shown in Fig. 6a shows a sawtooth shape because the ASC adjacency set is updated periodically. The update process involves more control packet exchanges and results in higher power consumption for control purposes.

6.1.1 Contribution Metric

The main goal of using the contribution metric is to implicitly allocate the relay and server tasks among all the nodes evenly. Fig. 2a shows the individual contribution of nodes and the delay performance. We calculate the SD of the individual contribution of each node and find that the value is smaller when a contribution metric is used (0.046 versus 0.065), which means that the time contributed by

TABLE 3
File Request Successful Ratio for ASC: Contribution Metric

	With ASC	Without ASC
Maximum	0.78	0.75
Mean	0.70	0.63
Minimum	0.64	0.52

TABLE 4
Delay Performance for ASC: Contribution Metric (in Seconds)

	With ASC	Without ASC
Maximum	101.82	122.50
Mean	100.20	110.65
Minimum	97.78	92.00

TABLE 5
File Request Successful Ratio for ASC: Popularity Metric

	With ASC	Without ASC
Maximum	0.79	0.75
Mean	0.78	0.63
Minimum	0.76	0.52

TABLE 6
Delay Performance for ASC: Popularity Metric (in Seconds)

	With ASC	Without ASC
Maximum	69.21	122.50
Mean	67.03	110.65
Minimum	66.81	92.00

TABLE 7
File Request Successful Ratio for ASC: Aggressiveness Metric

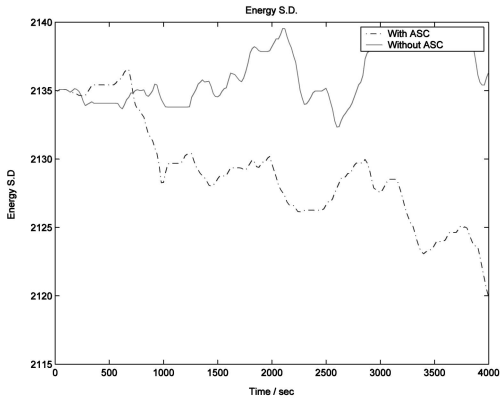
	With ASC	Without ASC
Maximum	0.85	0.75
Mean	0.82	0.63
Minimum	0.74	0.52

TABLE 8
Delay Performance for ASC:
Aggressiveness Metric (in Seconds)

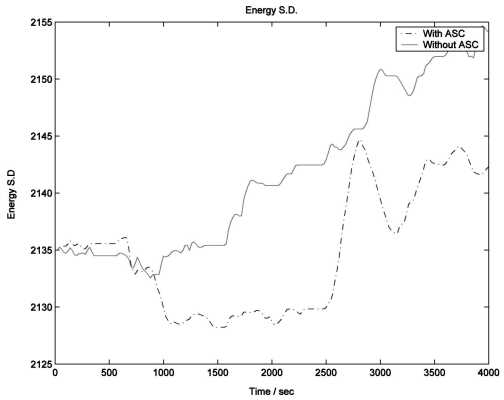
	With ASC	Without ASC
Maximum	90.51	122.50
Mean	88.20	110.65
Minimum	82.33	92.00

each node in acting as a server or relay between nodes is closer, and the public service is more evenly distributed. This is one aspect of fairness that we want to achieve. The second advantage is that, by more evenly distributing the relay and sever roles among nodes, a busy server or a busy relay is avoided. As a result, the file request successful ratio is increased and the delay involved in the file transfer is reduced (see Tables 3 and 4).

This implicit but intelligent "job allocation" is done by controlling the next hop of every single node based on the amount of public service already assigned to or done by each node.



(a)



(b)

Fig. 3. Performance results for ASC: aggressiveness and energy metrics. (a) Energy SD for ASC: aggressiveness metric. (b) Energy SD for ASC: energy metric.

6.1.2 Popularity Metric

We observe that the use of popularity could significantly increase the file request successful ratio and reduce the delay in file transmission (see Tables 5 and 6). We interpret these results in the following manner: Since more popular files are more frequently requested, by limiting the number of neighbors for a node who holds popular files, we can avoid a particular node that always acts as a server due to a large number of outgoing links. Thus, the server role becomes more evenly distributed among the nodes. This reduces the chance of request failure due to a busy server. The file request successful ratio is thus increased, and delay is reduced.

This more even distribution of server and relay jobs is also reflected in the energy levels of nodes. Consider Fig. 2b, where the SD of energy levels of nodes is smaller when the popularity metric is used (dotted line), which shows that the difference in energy levels between different nodes is controlled. This is regarded as one aspect of fairness that the use of the popularity metric can achieve.

6.1.3 Aggressiveness Metric

The aggressiveness metric intends to reduce the number of "next hops" of a user that gets used to downloading many files. This can free some original next hops of the user and allocate them for other parties to use. Consider Table 8. We

TABLE 9
File Request Successful Ratio for ASC: Energy Metric

	With ASC	Without ASC
Maximum	0.85	0.75
Mean	0.80	0.63
Minimum	0.77	0.52

TABLE 10
Delay Performance for ASC: Energy Metric (in Seconds)

	With ASC	Without ASC
Maximum	123.32	122.50
Mean	120.00	110.65
Minimum	117.56	92.00

see that one of the direct impacts of using the aggressiveness metric is a shorter file transfer delay. We interpret these results as follows: The metric reduces the number of "next hops" of some aggressive "downloaders." Thus, we free some relay nodes for other parties to use, and this can allocate the nodes for relay tasks in a better manner to cover the whole network area. This increases the choices of route for each node and increases the possibility of finding a shorter route. Since the topology of the whole network and route combinations are also varied by adjusting the size and content of the adjacency sets, the server selection process is also affected, and the file request successful ratio is higher (see Table 7). We also interpret this as a result of more even distribution of servers and relays.

The more even distribution using the aggressiveness metric has one more effect: The energy levels of nodes become less deviated, which we regard as a sort of fairness in energy sense (see Fig. 3a). Basically, the effect of the aggressiveness metric is similar to the popularity metric.

6.1.4 Energy Metric

Our intention of defining an energy metric is to reduce the difference between the energy levels of different nodes while keeping the network connected and the file requests satisfied ("energy-sense fairness"). Our way of doing this is similar as the use of other aforementioned metrics. We vary the adjacency set and topology of the network so that the server and relay tasks can be distributed in a fair manner.

Consider Fig. 3b. For the case in which the energy metric is used, the SD between the energy levels of different nodes are smaller (dotted line), which is our expected results. In doing so, we do not compromise the file request successful ratio and delay performance, as shown in Tables 9 and 10.

We have also examined the effect of different threshold values. Results are shown in Fig. 4. As discussed before,

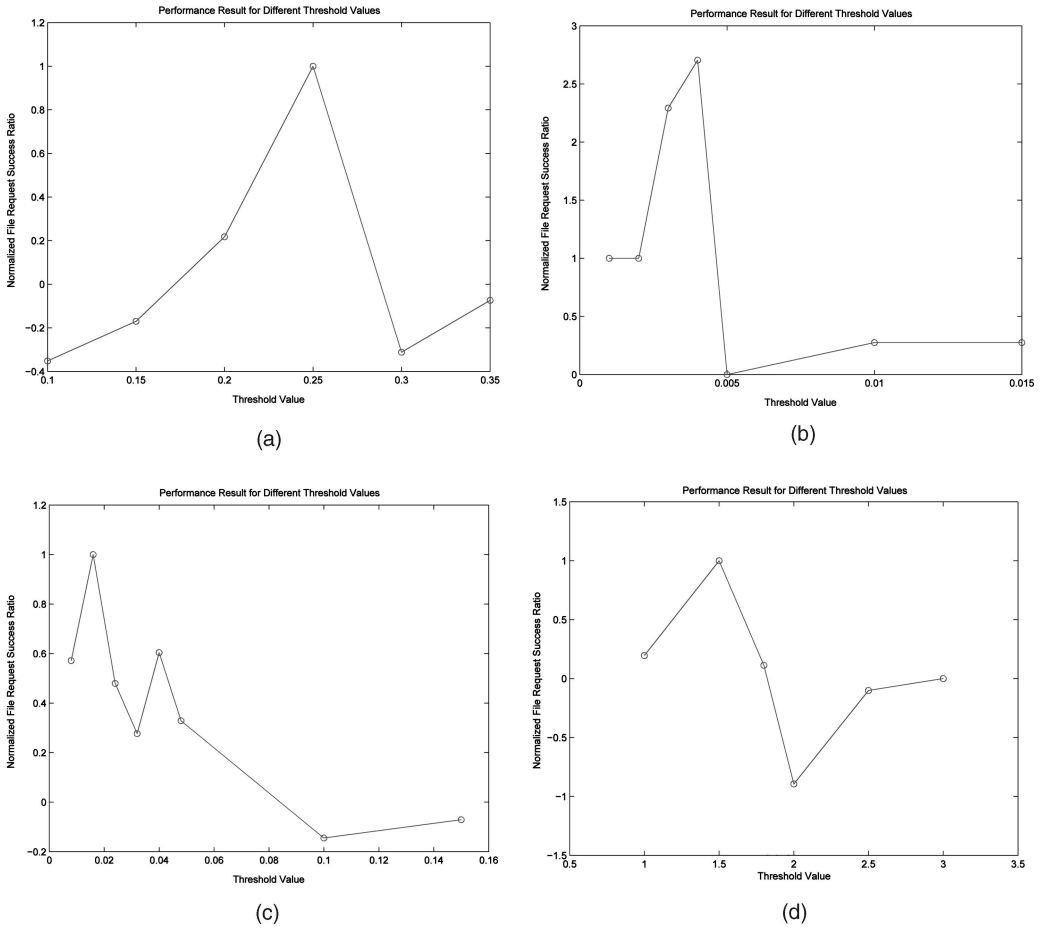


Fig. 4. Effect of different threshold values on the file success ratio. (a) Effect of the contribution metric threshold. (b) Effect of the aggressiveness metric threshold. (c) Effect of the popularity metric threshold. (d) Effect of the energy metric threshold.

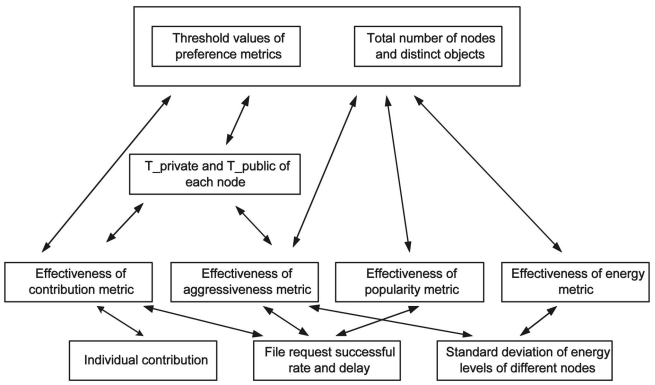


Fig. 5. Relationship among different system parameters and performance metrics in ASC.

TABLE 11
File Request Successful Ratio for CAW versus AWP

	With CAW	With AWP
Maximum	0.99	0.88
Mean	0.98	0.85
Minimum	0.89	0.81

TABLE 12
Delay Performance for CAW versus AWP

	With CAW	Without AWP
Maximum	110.20	143.50
Mean	100.50	122.80
Minimum	98.72	100.86

with varying threshold values, we do not observe a significant impact on delay performance. On the other hand, for the file request successful ratio, as shown in the result graphs, we can see that it is quite sensitive to some of the threshold value settings. Specifically, for a contribution threshold value of 0.25, the best performance is achieved. When the threshold value is smaller than 0.15 or larger than 0.3, the performance gain is minimal. For the aggressiveness metric, a small threshold value is suitable. This is also true for the popularity metric. For the energy metric, we do not see an obvious impact of the threshold values.

Finally, the relationship among different system parameters and the performance gain after applying ASC is summarized in Fig. 5.

6.2 Results for CAW

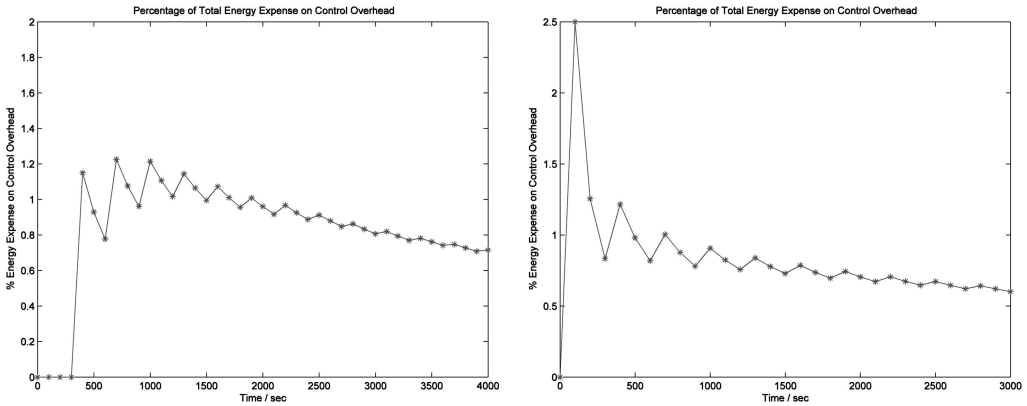
The main difference between CAW and AWP is the use of community. Now, we study the extra amount of energy used for community formation by using CAW compared with AWP. Consider Fig. 6b. We see that the average amount of energy used for community formation such as announcement packets broadcasting equals only about 1 percent of the total amount of energy expense in the file sharing process. This is reasonable since a control packet size is small compared with the file size involved in the file sharing process, which is in megabyte order. The energy used for the control purpose is higher when the network starts up due to the initialization of communities. After a short transient period, the energy expense on the control overhead reaches a more or less constant value. The curve shows a sawtooth shape because a node periodically updates its community membership, during which more control packets are exchanged.

6.2.1 Comparison with AWP

As shown in Tables 11 and 12, CAW outperforms AWP in the sense that it has a higher file request successful ratio and lower file transfer delay. We see that the file request successful ratio keeps higher than that in AWP when serving the same series of file download requests. After 3,000 seconds of file transfer, based on the total aggregated number of file requests and aggregated successful number of downloads, the file request successful ratio is around 95 percent (CAW) versus 85 percent (AWP). The average file transfer delay is shortened by around 20 seconds (16.7 percent). This is interpreted as a gain by grouping users with a similar interest together and letting them operate with the same schedule.

6.2.2 Comparison with the Case of No Sleeping Protocol

We also compare the performance of CAW with the case that no sleeping is used. We find that sleeping with CAW



(a) (b)

Fig. 6. Percentage of the total energy consumption used on control overhead. (a) ASC. (b) CAW.

TABLE 13
File Request Successful Ratio for CAW versus No Sleeping Protocol Is Used

	With CAW	Without AWP
Maximum	110.20	143.50
Mean	100.50	122.80
Minimum	98.72	100.86

TABLE 14
Delay Performance for CAW versus No Sleeping Protocol Is Used

	With CAW	Without AWP
Maximum	88.20	101.23
Mean	79.81	85.08
Minimum	75.22	80.71

does not reduce the file request successful ratio. After 3,000 seconds of file exchange, the aggregated successful number of file downloads is still higher for CAW. CAW also has better delay performance than no sleeping. Our interpretation is that CAW can group users with a similar interest together. Thus, it is easier for any user to find its target file because CAW has already ruled out users with different interests and let them be active in other time slots. This reduces the collision of simultaneous requests on a server-peer from users with different interests. Thus, the delay is reduced (see Tables 13 and 14).

Similar to other sleeping protocols, CAW is able to save mobile nodes' energy by sleeping. Our results show that, with CAW, the average energy level of nodes is higher than those without sleeping by 15 percent after 3,000 seconds of file transfer. This gain is expected to increase as more and more files are exchanged (see Fig. 7b).

Finally, the relationship among different system parameters and the performance gain after applying CAW is summarized in Fig. 8.

7 Further Discussions

In CAW, the similarity coefficient that we now use is the normalized Jaccard coefficient: The average value of the coefficient is found to be randomly fluctuating between 0.20 and 0.25 as files are exchanging for 100 users scattered in an area of around 300 m $\times$ 300 m (see Fig. 7a). We have started some new simulations on using other coefficients (for example, the Dice coefficient or cosine coefficient) to compare the similarity between nodes. Performance evaluation using these new coefficients is a new interesting topic.

Furthermore, the average spatial density of community in CAW is 25 communities in around a 300 m $\times$ 300 m area. We find that the spatial density has sensitive dependence

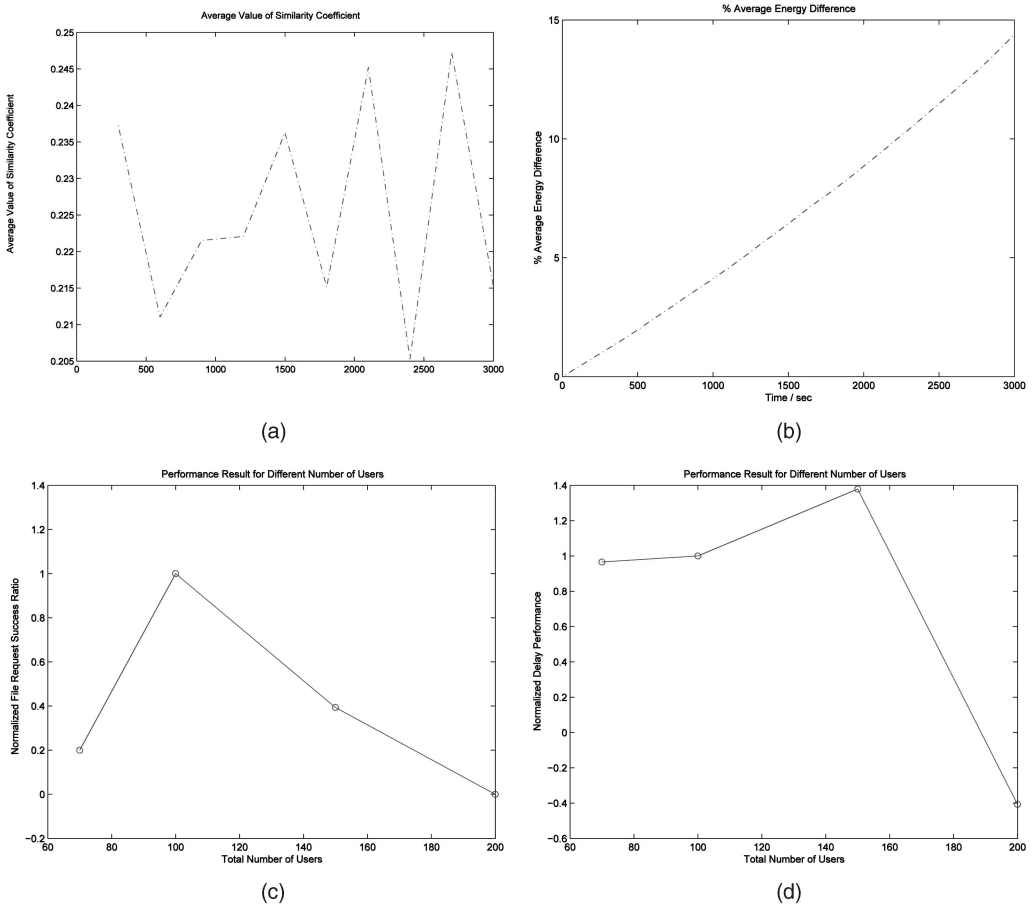


Fig. 7. Performance results for CAW. (a) Average value of Jaccard coefficient in CAW. (b) Average energy saved using CAW versus no sleeping. (c) Normalized file request successful ratio. (d) Normalized delay performance.

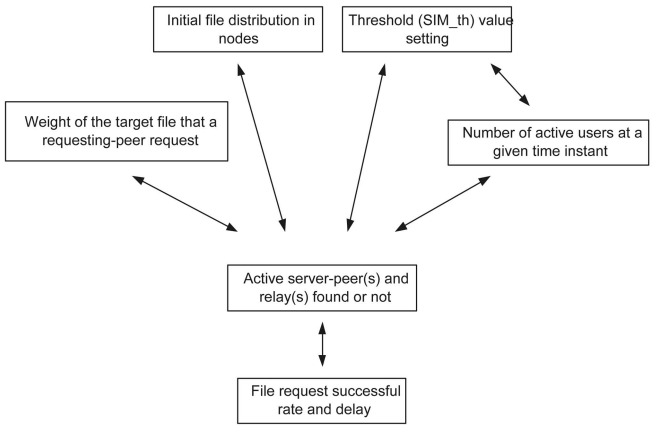


Fig. 8. Relationship among different system parameters and performance metrics in CAW.

on the mobility and threshold value of similarity coefficient used in CAW. Our protocol settings are supposed to be used in a lower mobility environment (for example, pedestrian walking down the street or hanging out in shopping malls) for a mean velocity of 3-5 km/hour. Tuning of the algorithms for higher mobility is another future research direction.

We have also studied the effect of varying the total number of users. The number of distinct files is kept at 60. Consider Figs. 7a and 7b, which show the normalized results. A result of 1 corresponds to the result obtained using 100 users and 60 distinct files. A value higher than 1 means better than the case with 100 users, and vice versa. The curves attain peaks for a certain number of users. Negative results are obtained on two ends of the x-axis. This shows that the performance of the CAW protocol gives good results for a certain suitable number of users for a given threshold value.

We have considered the change in the number of communities formed by the users by varying the value of the similarity threshold SIM_{th}. The results obtained are reasonable. For a lower value of SIM_{th}, more users are regarded as "similar with each other" and, thus, fewer communities are formed (see Figs. 9a and 9b). The average numbers of communities are 18 to 22 and 20 to 26 for SIM_{th} equals 0.05 and 0.35, respectively. Higher values of the threshold result in larger numbers of communities since the criteria become tighter for peers to be regarded as "similar." Consider Figs. 9c and 9d. The average number of communities is 22 to 28 when the value of SIM_{th} equals 0.55 or 0.75.

Finally, the "fairness" issue is closely related to the "availability" metric. We have investigated their interplay in detail and the results have been presented elsewhere [18].

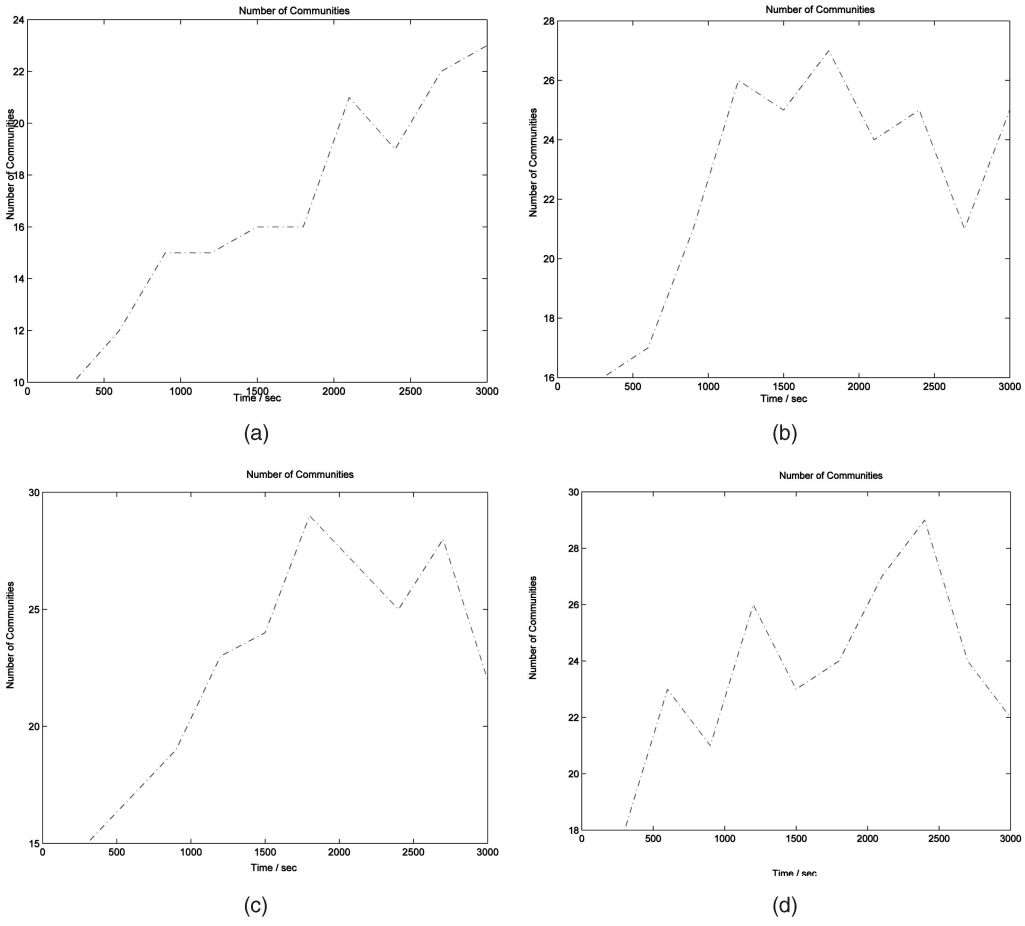


Fig. 9. Number of communities versus value of similarity threshold SIM_{th}. (a) $SIM_{th} = 0.05$. (b) $SIM_{th} = 0.35$. (c) $SIM_{th} = 0.55$. (d) $SIM_{th} = 0.75$.

8 Concluding Remarks

In this paper, we have proposed a new localized TC protocol, which is application driven. The proposed scheme is designed for a wireless P2P file sharing network. Our proposed scheme is based on several useful but simple policies, which we believe, when efficiently deployed in the environment considered in our study, could enhance the lifetime and the effectiveness of file sharing among the peers.

One limitation in our proposed algorithms is that we have not investigated an optimized "interplay" between the application layer and the physical layer, which is ideal for efficient file sharing in a wireless network. In our proposed algorithms, we have physical-layer control actions (for example, controlling who the neighbor is). We strongly agree that an important next step in this research is to propose an efficient "cross-layer" design for the file sharing network TC. Moreover, in the current paper, we have taken a more objective approach in considering the various topology configuration criteria. Indeed, it would be an interesting research problem as to how we should come up with a "weighted" combination of the several metrics.

Furthermore, how our proposed policies would impact the system evolution subject to different operating conditions (such as the existence of noncooperative peers, selfish peers, or even malicious peers) is an interesting further research topic. Most notably, the existence of "free riders" and "whitewashers" could possibly lower the lifetime of the file sharing network significantly because such users would definitely not contribute to the network by not acting as relay nodes. Furthermore, an even more detrimental situation would be having some malicious users who drop important control messages or fake them, possibly leading to the formation of an inefficient cluster.

Acknowledgments

The authors would like to thank the four anonymous reviewers for their critical and highly constructive comments on the paper. In particular, they are very thankful for Reviewer 3's detailed and insightful suggestions. Thanks are also due to Professor Terence Todd for his encouragement and professional handling of the paper.

References

[1] K.G. Anagnostakis and M.B. Greenwald, "Exchange-Based Incentive Mechanisms for Peer-to-Peer File Sharing," *Proc. 24th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '04),* pp. 524-533, Mar. 2004.

[2] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LAN's," *ACM SIGCOMM Computer Comm. Rev.,* vol. 24, pp. 212-225, Oct. 1994.

[3] *BitTorrent,* http://bitconjurer.org/BitTorrent/, 2005.

[4] A. Cerpa and D. Estrin, "ASCENT: Adaptive Self-Configuring Sensor Networks Topologies," *IEEE Trans. Mobile Computing,* vol. 3, pp. 272-285, July-Sept. 2004.

[5] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "SPAN: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *ACM Wireless Networks J.,* vol. 8, pp. 481-494, Sept. 2002.

[6] *Evolutionary Algorithms,* http://www.geatbx.com/docu/algindex-02.html, 2005.

[7] *eDonkey2000,* http://www.edonkey2000.com/, 2005.

[8] *FastTrack,* http://www.slyck.com/ft.php, 2005.

[9] L.M. Feeney and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," *Proc. IEEE INFOCOM,* vol. 3, pp. 1548-1557, Apr. 2001.

[10] S. Ganeriwal and M.B. Srivastava, "Reputation-Based Framework for High Integrity Sensor Networks," *Proc. Second ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '04),* pp. 66-77, Oct. 2004.

[11] Z. Ge, D.R. Figueiredo, J. Sharad, J. Kurose, and D. Towsley, "Modeling Peer-Peer File Sharing Systems," *Proc. IEEE INFOCOM,* vol. 3, pp. 2188-2198, Mar.-Apr 2003.

[12] D. Hales, "From Selfish Nodes to Cooperative Networks—Emergent Link-Based Incentives in Peer-to-Peer Networks," *Proc. Fourth Int'l Conf. Peer-to-Peer Computing (P2P '04),* pp. 151-158, Aug. 2004.

[13] *HKCSL,* http://one2free.hkcsl.com/eng/main/index.jsp, 2005.

[14] *IEEE Standard 802.11b, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* IEEE, 1999.

[15] D. Karger and M. Ruhl, "Finding Nearest Neighbors in Growth-Restricted Metrics," *Proc. 34th ACM Symp. Theory of Computing (STOC '02),* pp. 63-66, 2002.

[16] A.K.H. Leung and Y.K. Kwok, "On Topology Control of Wireless Peer-to-Peer File Sharing Networks: Energy Efficiency, Fairness and Incentive," *Proc. Sixth IEEE Int'l Symp. World of Wireless (WoWMoM '05),* pp. 318-323, June 2005.

[17] A.K.H. Leung and Y.K. Kwok, "Community-Based Asynchronous Wakeup Protocol for Wireless Peer-to-Peer File Sharing Networks," *Proc. Second Ann. IEEE Int'l Conf. Mobile and Ubiquitous Systems (MobiQuitous '05),* pp. 342-350, July 2005.

[18] A.K.H. Leung and Y.K. Kwok, "An Efficient and Practical Greedy Algorithm for Server-Peer Selection in Wireless Peer-to-Peer File Sharing Networks," *Proc. First Int'l Conf. Mobile Ad Hoc and Sensor Networks (MSN '05),* pp. 1016-1025, Dec. 2005.

[19] Z. Li and M.H. Ammar, "A File-Centric Model for Peer-to-Peer File Sharing Systems," *Proc. 11th IEEE Int'l Conf. Network Protocols (ICNP '03),* pp. 28-37, Nov. 2003.

[20] R.T.B. Ma, S.C.M. Lee, J.C.S. Lui, and D.K.Y. Yau, "A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks," *Proc. ACM Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS '04),* pp. 189-198, June 2004.

[21] J.G. Markoulidakis, G.L. Lyberopoulos, D.F. Tsirkas, and E.D. Sykas, "Mobility Modeling in Third-Generation Mobile Telecommunications Systems," *IEEE Personal Comm.,* vol. 4, no. 4, pp. 41-56, Aug. 1997.

[22] A. Muqattash and M.M. Krunz, "A Distributed Transmission Power Control Protocol for Mobile Ad Hoc Networks," *IEEE Trans. Mobile Computing,* vol. 3, pp. 113-128, Apr. 2004.

[23] *Napster,* http://www.napster.com/, 2004.

[24] V. Paruchuri, S. Basavaraju, A. Durresi, R. Kannan, and S.S. Iyengar, "Random Asynchronous Wakeup Protocol for Sensor Networks," *Proc. First IEEE Int'l Conf. Broadband Networks (BroadNets '04),* pp. 710-717, Oct. 2004.

[25] R. Rajaraman, "Topology Control and Routing in Ad Hoc Networks: A Survey," *ACM SIGACT News 33,* pp. 60-73, July 2002.

[26] T.S. Rappaport, *Wireless Comm.: Principles and Practice.* Prentice Hall, 1996.

[27] V. Rodoplu and T.H. Meng, "Minimum Energy Mobile Wireless Networks," *IEEE J. Selected Areas in Comm.,* vol. 17, pp. 1333-1344, Aug. 1999.

[28] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware '01),* pp. 329-350, Nov. 2001.

[29] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

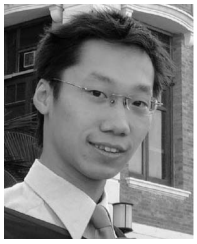
[30] S. Singh and C.S. Raghavendra, "PAMAS—Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks," *ACM SIGCOMM Computer Comm. Rev.,* vol. 28, pp. 5-26, July 1998.

[31] S. Singh, M. Woo, and C.S. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks," *Proc. ACM MobiCom,* pp. 181-190, Oct. 1998.

[32] Slyck.com, http://www.slyck.com/news.php?story=574, 2005.

[33] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Trans. Networking,* vol. 11, pp. 17-32, Feb. 2003.

[34] VirtualUMTS—Lucent Technologies, http://www.ee.ust.hk/%7Eeeknlau/AppliedResearch/VUMTS/Virtual%20UMTS.htm, 2005.
[35] W. Wang, X.Y. Li, and Y. Wang, "Truthful Multicast Routing in Selfish Wireless Networks," *Proc. ACM MobiCom,* pp. 245-259, Sept. 2004.
[36] Washington Times Online, http://www.washtimes.com/technology/20040303-094741-3574r.htm, 2005.
[37] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. IEEE INFOCOM,* vol. 3, pp. 1567-1576, June 2002.
[38] M.K.H. Yeung and Y.K. Kwok, "A Game Theoretic Approach to Energy Efficient Cooperative Cache Maintenance in MANETs," *Proc. 16th Ann. IEEE Int'l Symp. Personal Indoor and Mobile Radio Comm. (PIMRC '05),* Sept. 2005.
[39] R. Zheng, C. Hou, and S. Lui, "Asynchronous Wakeup for Ad Hoc Networks," *Proc. ACM MobiHoc,* pp. 35-45, June 2003.



Andrew Ka-Ho Leung received the BEng (First Class Honours) and MPhil degrees from the Department of Electrical and Electronic Engineering, The University of Hong Kong in 2003 and 2005, respectively. His research interests are in mobile computing and wireless networking. In pursuit of the MPhil degree, he received the Outstanding Teaching Assistant Award in 2003-2004 and a DuPont scholarship in 2004-2005 for his dedicated work. After graduation, he joined The Hongkong Electric Co., Ltd. and became a graduate member of the Hong Kong Institution of Engineers (HKIE). He is a student member of the IEEE.

Yu-Kwong Kwok received the BSc degree in computer engineering from the University of Hong Kong (HKU) in 1991 and the MPhil and PhD degrees in computer science from the Hong Kong University of Science and Technology (HKUST) in 1994 and 1997, respectively. He is now an associate professor in the Electrical and Computer Engineering Department, Colorado State University (CSU). Before joining CSU in August 2007, he was an associate professor in the Department of Electrical and Electronic Engineering at HKU, which he joined in August 1998. Prior to that, he was a visiting scholar for one year in the parallel processing laboratory at the School of Electrical and Computer Engineering at Purdue University. From August 2004 to July 2005, he served as a visiting associate professor in the Department of Electrical Engineering—Systems at the University of Southern California on his sabbatical leave from HKU. His research interests include distributed computing systems, wireless networking, and reconfigurable heterogeneous multiprocessors. He is a senior member of the IEEE and a member of the ACM, the IEEE Computer Society, and the IEEE Communications Society. He received the Outstanding Young Researcher Award from HKU in November 2004.