Community-Based Asynchronous Wakeup Protocol for Wireless Peer-to-Peer File Sharing Networks

Andrew Ka-Ho Leung and Yu-Kwong Kwok
Department of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam Road, Hong Kong
Corresponding Author: Yu-Kwong Kwok (Email: *ykwok@hku.hk*)

***Abstract*— Ubiquitous Peer-to-Peer (P2P) networking is widely expected to be manifested in a wireless environment in the near future. However, to realize such an interesting mobile computing platform, energy efficiency is one of the most critical resources management issues yet to be tackled. Unfortunately, energy efficient wireless P2P networking is still a relatively less explored topic as it is quite challenging to tackle the energy management problem without centralized control. In this paper, we meet this research challenge by proposing a new distributed protocol, called Community-Based Asynchronous Wakeup Protocol, CAWP, for energy conservation in wireless P2P file sharing networks. Simulation results show that our proposed CAWP is found to be highly effective in that it can remarkably increase the energy efficiency of the participants in a wireless P2P system.**

***Index Terms*—distributed systems, wireless networking, P2P systems, file sharing, energy efficiency, network protocols.**

I. Introduction

Recently, we have witnessed that more and more wireless devices have become part of our life. In addition to the basic function of providing connectivity among people, wireless devices like 3G mobile phones and WLAN- or Bluetooth-enabled PDAs also provide instant web access services and tetherless personal area network coverage to a user. When these users meet each other and establish an ad hoc network, various interesting applications such as peer-to-peer (P2P) file sharing can be carried out. Indeed, given the highly pervasive nature of *wired* P2P systems [6], [2], [21], it is widely envisioned that *wireless* P2P will logically be the next step which allows mobile and ubiquitous file resources sharing.

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 of limited energy. Furthermore, mobile devices are found to have ever increasing energy consumption rates [14], more so in a P2P system. Indeed, to tackle the energy efficiency problem of mobile devices in a judicious manner is very challenging and has baffled researchers for years [1], [12]–[19].

In this paper, we meet this challenge by proposing a new energy conserving protocol, namely *Community-Based Asynchronous Wakeup Protocol, CAWP.* Specifically, we note that in most P2P file sharing applications, peers are always looking for their favorite file objects to download from other peers, and usually a peer having similar *file preferences* turns out to be the one who owns the target file objects and can send them to the original requester. This motivates us to design and propose a distributed "community-based sleep-and-wake" protocol for energy conservation in wireless P2P networks.

Putting a device into sleep mode is found to be a useful way for energy conservation [22], [5], [10]. In our protocol, mobile users are grouped into "virtual communities" in which they "sleep-and-wake" together but in an asynchronous manner. The grouping is done with respect to their file preferences. For example, users with similar favorite singers are grouped into the same community in music file sharing network and match potential requester and sender of files. This design remarkably conserves the energy of mobile users. More importantly, in our study we find that this intentional sleeping does not have significant negative impact on the file searching successful rate, as shown in Section IV of this paper.

The remainder of this paper is organized as follows. In the next section we discuss the background and related work on energy conservation of wireless networks. In Section III we describe our Community-Based Asynchronous Wakeup Protocol, CAWP. In Section IV we present our simulation results. We provide some concluding remarks in the last section.

This research was supported by a grant from the Research Grants Council of the HKSAR under project number HKU 7157/04E.

Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05)
0-7695-2375-7/05 $20.00 © 2005 IEEE

II. Background

A. The Significance of Sleep Mode

We first describe the terminology used in the literature which is highly related to the energy consumption rate and could characterize mobile wireless devices. In general, four modes of operation can be defined for a mobile wireless device: (1) Transmit mode, in which a device transmits packets and consumes the largest amount of energy; (2) Receive mode, in which the device listens and extracts payload data from received packets; (3) Idle mode, in which the device listens but does not decode the data; and (4) Sleep mode, in which the device totally shuts down its transmitter and receiver.

Feeney and Nilsson have done an investigation on power consumption of WLAN cards and the results are reported in [9]. In the IEEE 802.11 WLAN standard, the maximum transmit power of a mobile device is around 1 Watt (30 dbm). The four modes of operation have different levels of power consumption and the ratio (Transmit:Receive:Idle:Sleep) is around $1 : 0.6 : 0.5 : 0.08$. For example, experiments done by Chen *et al.* on a Cabletron 802.11 network card show that the power consumption levels for the four modes are 1400 mW, 1000 mW, 830 mW and 130 mW respectively.

B. Related Work

Now we can see that putting a device into sleep mode is an effective way to reduce the energy consumption and elongate the battery life of the device. However, the biggest challenge here is that once the mobile device is put into sleep mode, it cannot hear any more packet "on air" and it may miss some packets intended for it. Thus, mobile devices are needed to "wakeup" appropriately so as to maintain network connectivity. In other words, mobile devices must have some "sleep-and-wake" built-in protocols in their design.

There are two categories of "sleep-and-wake" protocols suggested in the literature. The first one is *scheduled protocol* and the second one is *on-demand protocol*. In addition to energy conservation, sleep-and-wake protocol is also related to Medium Access Control (MAC) layer design or Topology Control.

For *scheduled protocol*, there are two subtypes. The first subtype is *synchronous scheduling*. IEEE 802.11 Power Save Mode (PSM) [7] conserves energy by putting the mobile devices into sleep mode periodically. In order to maintain connectivity with other nodes, all the participating nodes need to synchronize their clocks so that a common wakeup time can be set for all the nodes.

In PAMAS [19], RTS and CTS packets are exchanged in a control channel separated from the data channel. The RTS/CTS packets contain the length of packet transmission, from which other nodes (except the intended receiver of packets) can estimate for how long should it power off their receiver. S-MAC [22] also employs periodic sleep and listening, network nodes are synchronized by transmitting synchronization (SYNC) packets before any RTS/CTS and DATA/ACK exchange. Together with overhearing avoidance and message passing, S-MAC is self-identified as an energy conserving MAC protocol for wireless sensor networks.

Scheduled protocol is suitable for *single hop* localized network because all mobile nodes need to synchronize their clocks. Synchronization of clocks of a large number and scattered mobile nodes is found to be non-trivial and costly [8]. Thus, this subtype of scheduled protocol is not scalable.

The second subtype is *asynchronous scheduling*. Tseng *et al.* pioneered this area of work in [20]. They propose modifications to IEEE 802.11 PSM to support sleep-and-wake protocol in an asynchronous manner. In [15], Paruchuri *et al.* propose the "Random Asynchronous Wakeup Protocol" (RAW). RAW is based on the concept that when a region of the wireless networks has a high density of nodes, some of the nodes can be put in sleep mode to conserve energy without affecting the network connectivity. The nodes employ *random* wakeup schedules and therefore, it removes the need of synchronization and simplifies the design. One of the drawbacks of this design is that a node may be sleeping when some packets are transmitted to it. Packets need to be queued and delay is introduced. To reduce the delay, the authors suggest a set of criteria to define the forwarding of packets to neighbor nodes of the destination. However, RAW is designed for fixed but not mobile network.

Based on block design combinatorics, Zheng *et al.* suggest an Asynchronous Wakeup Protocol (AWP) [23] in which nodes wakeup according to a *Wakeup Schedule Function, WSF*. These WSFs have an interesting property that any shifted version of a WSF would still have one or more active period of nodes overlapped, this reduces the complexity involved in synchronization of all nodes and avoids the problem of network partition due to different sleep-and-wake schedules of nodes.

Another way to solve the "packet-miss-during-sleeping" problem is packet-buffering. In [11], packets intended for a sleeping device is buffered in a base station, the sleeping device can wake up periodically to poll the base station for any stored packets.

The second category of sleep-and-wake protocol is on demand protocol. SPAN [5] is one of these types. In SPAN, some nodes are elected as "coordinators" to

tain connectivity and preserve capacity of the network. Coordinators stay awake and perform routing of packets while other redundant nodes are allowed to sleep to conserve energy. The coordinators are elected in a distributed manner and the role of coordinator is rotated among different nodes to introduce fairness. In order words, the mobile devices toggle between coordinator mode and redundant mode in an "on demand" manner.

ASCENT, Adaptive Self-Configuring sEnsor Networks Topologies [3], is an on demand protocol. Specifically, ASCENT puts different mobile devices into one of the four states: Active, Test, Passive, Sleep. The nodes would monitor the neighbor density and data loss level to decide they should be in which mode. ASCENT is also designed for wireless sensor networks.

Unfortunately, none of the above protocols take ubiquitous, wireless P2P as their target platform and propose a tailor-made energy-conserving protocol. Thus, we aim at providing some insight by suggesting an *application level-driven, community-based asynchronous wakeup protocol* for mobile wireless P2P file sharing networks. Our design philosophy is similar to *Clique-Based Randomized Multiple Access* (CRMA) [10], which also groups users into "cliques" and uses pseudo-random number generator to determine in which time frame(s) is each clique active. However, their focus is to resolve conflicting active time frame selections for different cliques.

III. Our Proposed Approach

A. Overview

Our protocol belongs to the asynchronous scheduling category and uses cross-layer design concept. With knowledge from application layer, the protocol guides the formation of "virtual communities" among mobile users. We define *community* as a set of two or more mobile users that perform certain computations in a particular manner. For example, in a music file sharing network, users with similar preferences, 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 interest (this is the usual reason why 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.

In order to maintain the connection between members of different community so as to avoid network partition, we apply the concept of *block design combinatorics* studied in [23], wakeup schedules of different community are guaranteed to have partial overlapping so that network connectivity can be preserved.

B. Formation of Community

CAWP is designed for wireless P2P file sharing platform. The concept of "community" is inspired by the fact that, no matter which file sharing protocol is used, peers that share files with each other (senders and requesters) usually have similar preferences. For example, in a music file sharing network, user A is asking for a song X by singer Y because singer Y is one of the idols of user A. Who will have file X? Normally we would download and keep a music file in our disk because we like it. Thus, in this case another user B who also in favor of music by singer Y may have that file X and can share it with user A.

Indeed, "community" usually refers to two or more entities with a particular bias or similar preferences. Now the question is how to identify and form communities in a wireless P2P network.

1) Similarity Measurement—Traditional Concept: We employ *similarity measurement* in the field of *Information Retrieval (IR)*. IR concerns about the study of the process of storing organizing, indexing, and accessing (usually a large number of) objects. *Google* search engine is an example of Information Retrieval System. In IR, there are many well defined measures to study the level of similarity between objects. In CAWP, we employs the concept of some of these techniques to measure the similarity between users and classify them into communities. The implementation of these techniques is given in the next section and here we first discuss the traditional concept of similarity measurement.

First we introduce the original idea in the context of comparing *word documents*. Given a number of word documents, we can count the frequency of occurrence of different terms in the documents. As an example, let $D = \{d_n, 0 \leq i \leq N\}$ denotes a set of N documents. We pick up K terms, denoted by $t_k, 1 \leq k \leq K$ and count their number of of occurrence in these N documents.

Now we study the *Similarity,* $SIM(i, j)$, of two documents d_i and d_j. The basic concept is that we want to determine the similarity between two documents by comparing the number of terms found in both documents, where f_{ik} is the total number of songs sung by singer s_k.

For example, two news articles belong to the same topic (say, economics) should have common terms like "investment", "job", "employment rate", etc., while another article about health would consist of other patch of terms such as "medicine", "diseases", etc. This way of measuring similarity is commonly used in the literature, interested readers may seek more details in [17].

However, some terms we use in daily life are frequently used. For example, terms such as *is, am, are, it*, etc., are always present in documents no matter what topics of documents we are comparing. Therefore, different terms are of different *importance* in the determination of similarity. Thus, we define *weight* of a term and use "weight" instead of simply "frequent of occurrence" to compare terms found in documents. There are several ways of calculating this "weight" [17]. In particular, one easy and common method is to define a weighting function w_{ik} for each term t_k, corresponding to a document d_i of which the value increases as the frequency of occurrence of t_k in d_i increases but the value decreases as the total number of documents containing it increases (to ignore the effect of common terms like is, am, are, it, etc.). Using logarithmic representation, the weighting function is commonly calculated as:

$$w_{ik} = f_{ik}[\log_2(N) - \log_2(f_k^{doc}) + 1] \quad (1)$$

where f_{ik} is the frequency of occurrence of term t_k found in document d_i. f_k^{doc} is the number of documents that the term t_k can be found. Given that there are totally K terms under consideration, *similarity,* $SIM_{doc}(i,j)$, between two documents d_i and d_j is defined as:

$$SIM_{doc}(i,j) = \frac{\sum_{k=1}^{K}(w_{ik}w_{jk})}{\sum_{k=1}^{K} w_{ik} + \sum_{k=1}^{K} w_{jk} - \sum_{k=1}^{K}(w_{ik}w_{jk})} \quad (2)$$

This expression is known as *Jaccard coefficient* in the literature [17].

2) Similarity Measurement in CAWP: Now we borrow the idea of similarity measurement for word documents and apply it into P2P file sharing network, say, music file sharing network. 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 of songs stored in these music files construct a singer set $S = \{s_k, 0 \leq k \leq N\}$. We compare the similarity *between any two users* [1] u_i *and* u_j based on the number of songs they own for different singers. If two users both have many songs by singer A, 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/her song. Therefore "both having songs by singer X" is not considered as an important feature that two users are similar. This phenomenon is similar to the traditional concept that frequent terms are regarded as "less important" in the weighting function.

To conclude, we define the weight w_{ik} of a singer s_k in user u_i as:

$$w_{ik} = f_{ik}[\log_2(N) - \log_2(f_k^{users}) + 1] \quad (3)$$

[1] We use terms users, mobile devices and nodes interchangeably

where f_{ik} is the total number of songs by singer s_k found in user u_i's disk. f_k^{users} is the number of users that own songs sung by singer s_k. Then, *similarity,* $SIM_{users}(i,j)$, between two users u_i and u_j in the P2P file sharing network is again defined as:

$$SIM_{users}(i,j) = \frac{\sum_{k=1}^{K}(w_{ik}w_{jk})}{\sum_{k=1}^{K} w_{ik} + \sum_{k=1}^{K} w_{jk} - \sum_{k=1}^{K}(w_{ik}w_{jk})} \quad (4)$$

given that there are totally K singers under consideration. The selection of this K singers should be diversified enough to reflect the interest of different user. The selection of the value of K also depends on the computation power of mobile devices. Using the K weight values, we can construct a K-dimensional vector for each user. This vector is a *Preference Vector* representing user's interest:

$$V_{pref} = \{w_{i1}, w_{i2}, \ldots, w_{iK}\} \quad (5)$$

3) Definition of Community: In CAWP, users belong to community C if their level of similarity is higher than certain threshold SIM_{th}. We emphasize that after joining one community, a user would not join another community (reason will be given in the next section) and each member of the community would have the responsibility to "promote" this community.

The key concept of CAWP is that members of the same community would employ the *same* wakeup schedule. The rationale behind our design is that we group users together, let them sleep-and-wake together, increase the chance of successful file exchange, conserve energy and minimize interference by offsetting active periods of users from different communities. Implementation details, related problems and solutions are given in the following sections.

C. Asynchronous Wakeup in CAWP

A requirement of using wakeup scheduling such as IEEE 802.11 PSM and SMAC is that all users need to synchronize with each other. This is a complex task when the scale of network is getting larger and larger. To avoid this problem, Zheng *et al.* [23] propose the use of Asynchronous Wakeup Protocol, AWP. The main feature of their design is that, using theory in *Block Design Combinatorics*, they employ *Wakeup Schedule Function, WSF*, which guarantees mobile nodes using different schedules must have their active period overlapped in at least one time slot. For details, the reader is referred to [23]. Now we quote the result as below, using a so-called "(7, 3, 1)-design":

Given that different nodes's clocks are not synchronized, start at random time instant. Consider a total number of seven time slots, each with a fixed duratio

summed up to a time frame of length T. If each node wakes up and becomes active in three of the time slots, namely x^{th}, y^{th}, z^{th} time slots respectively (x, y, 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 Figure 1.)

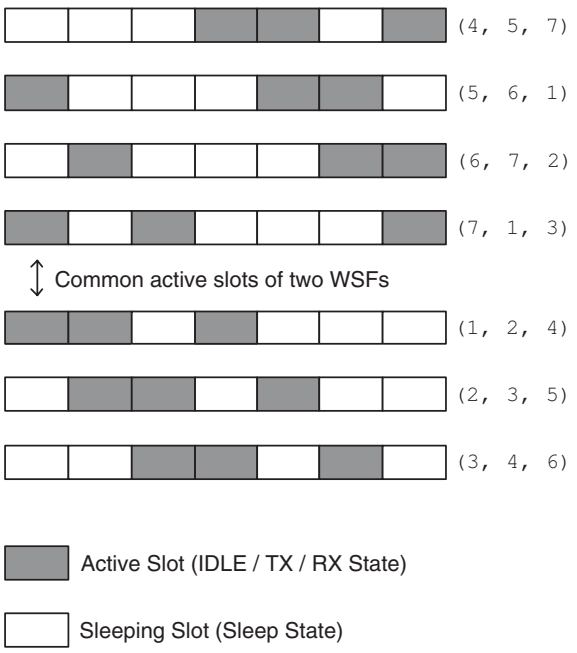

Fig. 1. '(7, 3, 1)-design' of WSF.

$$\{(1,2,4);(2,3,5);(3,4,6);\\ (4,5,7);(5,6,1);(6,7,2);(7,1,3)\} \quad (6)$$

Based on this "(7, 3, 1)-design", together with a neighbor discovery protocol given in [23], any two neighbors must be able to find each other.

D. Protocol Details and Implementation

1) Basic Concept: To summarize, CAWP is inspired by AWP. We extend AWP by adding the concept of "community". In order to determine the similarity between users from different communities in the network and group them into communities, we employ traditional similarity measurement techniques in Information Retrieval and Jaccard coefficient.

There are two purposes of grouping users into communities. Firstly, doing so can allow users with similar interest to be "active together". The reason is that we believe users are mostly decided to communicate with users who are with similar preference in terms of file sharing. By grouping them into virtual communities, we are able to increase their chance of getting desired files. Secondly, the grouping also reduces the interference among users with different interests. Because communities would choose different wakeup schedules, their active periods are offset. When one group of users are operating, other groups which are not interested in their file transfer would simply get into sleep mode.

Of course, we do *not* rule out the possibility that a user interested in a file that is outside his / her community. 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 Equation (6)), they 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 larger delay since they do not employ exactly the same schedules. For example, in (7, 3, 1)-design, the worse case is two communities overlap in the last time slot of a time frame (e.g., 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.

Therefore, network originally connected would *not* be partitioned after CAWP is used.

2) Protocol Layering: ¿From a layering protocol perspective, CAWP is in-between the MAC layer and Network Layer. As just mentioned, packets have to be routed and buffered by the Network Layer according to the sleep-and-wake information given from its lower layer, the CAWP layer. CAWP layer relies on the MAC layer to resolve contention. Signaling mechanism like RTS/CTS packets exchange is needed.

3) Distributed Community Formation: The first step of CAWP is formation of communities. Similar to many scheduling protocol aforementioned (e.g., SMAC), A user first listens for a certain amount of time. This value depends on density of nodes, applications running on top of the network, etc. After this time is expired, an announcement message is broadcast (if this user does not receive other announcement message in the listening period). This user is similar to the "synchronizer" in SMAC. The message contains the preference vector shown in Equation (5), showing the weight of different singers with respect to its disk. The message also contains the WSF 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 Jaccard coefficient for them is higher than certain threshold, they are regarded as belonging to the same community. The receiver of this announcement message would empl

same WSF. At this point the community has two members. Otherwise, if the receiver finds that the similarity is lower than the threshold, it continues to listen until it finds a community to join. If it cannot join a community after it finishes it listening period, it sends announcement message itself and ask any other members to join.

To reduce contention, listening periods of different nodes are random values range from 0 to $T_{announce}$.

Each user would join only one group. The key concept is that we assume the "preferences" of a user would not change very fast although number of files owned by users keep changing. Specifically, preference is an attribute that only changes slowly and gradually relative to mobility of the user. This ensures a larger extent in reducing energy consumption because joining more groups simply means more frequent wakeup. This is different from the concept in SMAC but can still solve the "Border nodes problem". Nodes within range of more than one user of similar interest but trying to establish communities independently simply pick up one to join.

4) Neighbor Discovery: Unlike AWP, CAWP does not need to transmit beacon messages in every active time slot to let neighbors know that they are active. When communities are set up, each node already employed the same WSF with other members in the same community, they know already that there will be someone to listen to their file searching request hereafter. In order to maintain connections with other communities, each node also records the WSF of neighbors in other communities. This information, together with the ID of the sender of announcement message and the total number of members, is stored in a *neighborhood table*. For each community, announcement message would be re-sent by the members every r time frames. r is a random value smaller than R where R is predefined. This is to allow every member to "promote" their community so that new nodes enter the network can join their corresponding communities. The second purpose is to adopt to network dynamics. Since all nodes are moving, members originally belong to the same community and using the same WSF may leave each other when moving. Therefore, after r time frames, not only new nodes would find a community to join, old members need to check if they can still hear the same WSF broadcasting in its proximity, if the number of members found is keep decreasing after nr time frames (n is a predefined integer), a node would decide to re-join 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 PC before they went out. Now our protocol needs only one more step—to calculate the preference vector shown in Equation (5) using the PC at home, download it into the mobile device together with music files. It should be noted that this calculation can be done by PC, and therefore, it does not increase overheads of our protocol 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 these information estimated from 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 devices is a finite addition and subtraction only, which does not cause feasibility problem. Since usually the human being's set of preferences is a long term behavioral attribute and and changes slowly compared to network dynamics and mobility of users, the community that a member joins need not be changed frequently. As mentioned before, a regular update and re-join process is needed only when too many members have moved away.

IV. Performance Results

A. Simulations

Now we study the performance of CAWP using simulations. In our simulations, 100 users (mobile devices) are scattered in a 100m $\times$ 100m area and are allowed to move freely. There are 60 different file objects in the network and 5 different singers. The application layer throughput of each mobile device is 0.6 Mbps. The threshold value of Jasscard coefficient used in comparing similarity levels of users so as to make community-joint decision is set as -1.1. The maximum file size of each song is assumed to 5 MB (similar to normal sizes of MP3 music files circulated in the Internet).

B. Traffic Pattern and Mobility Model

The traffic pattern of the system is simulated with the use of inter-request arrival times. We assume that the calls arrive according to a Poisson distribution where the call arrival times and the inter-arrival times between calls are mutually independent. This implies that inter-arrival times τ are exponentially distributed (mean = 100 sec) [13], [16].

Our mobility model assumes users' velocities follow a Gaussian distribution [13] with mean = 3 km/hr, (i.e., 0.83 m/s) and variance = 0.54. This simulates movement of pedestrians.

C. Energy Consumption Model

The wireless devices are assumed to have four possible modes of operation: Transmit, Receive, Idle and Sleep. The energy consumption ratio of the four modes is set as 1 : 0.6 : 0.5 0.08, as indicated in [15] 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 first three P terms represent power consumption in Transmit, Receive, Idle and Sleep modes, respectively, $P_{\text{overheads}}$ represents the energy consumption in transmission and reception of control overheads and the T terms represent corresponding time durations that the mobile devices are in different modes.

D. Transmit Power and Path Loss

The transmit power of mobile devices is assumed to be fixed at 1 Watt. For path loss in radio propagation, we adopt the *Okumura-Hata Model* which is commonly used in the literature [16] to estimate the path loss:

$$\begin{aligned}\text{Path Loss in dB} &= 69.5 + 26.16\log \text{freq}_c - 13.82\log h_{\text{ant}} \\ &-\text{corr}(h_{\text{ant}}) + (44.9 - 6.55\log h_{\text{ant}})\log d\end{aligned} \tag{7}$$

where freq_c is the carrier frequency (in MHz) of the transmitted signal, h_{ant} is the height of the antenna of the mobile devices. Furthermore, $\text{corr}(.)$ is a correction term and can be taken as zero in our case. Finally, d is the propagated distance (in km).

E. Performance Results

Since our CAWP is inspired by AWP [23], we compare the performance of CAWP with AWP. We also compare the case when CAWP is used and the case when no sleeping protocol is used.

First we consider the case that CAWP is used and the case that no sleeping protocol is used and all mobile devices are always "on" and at IDLE state if no file transmission is taken place, i.e., file sharing networks without any sleeping protocol. Figure 2 shows the difference in average energy of mobile devices between the two cases. We see that CAWP can significantly increase the energy efficiency of the network. The average energy of mobile devices is higher than the case without CAWP by around 15% after 3000 seconds of file transfer operation. We also see that this saving of energy has an increasing trend, as the network is running, the difference between the two cases becomes larger.

Then, we consider the file request successful rate. Since CAWP puts some nodes into sleep mode, the intuition is that this would reduce the chance for the holder of a file to receive a file searching request (since the holder may be in sleep mode when someone is transmitting a request). However, CAWP uses application level knowledge to intelligently group users into communities which allows users with similar interest to become active together. Thus CAWP does not suffer from high file request failure rate but instead has a higher successful file request rate than the case where all mobile nodes are always "on", as shown in Figure 3 (nearly 80% vs around 65%).

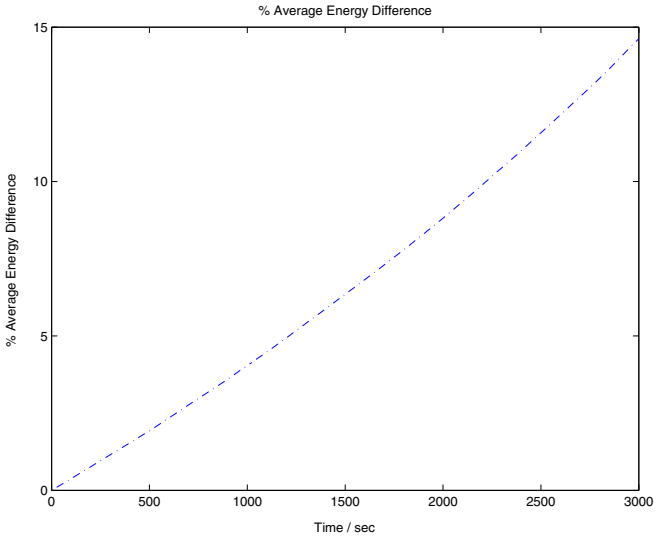

Fig. 2. Average energy of mobile devices.

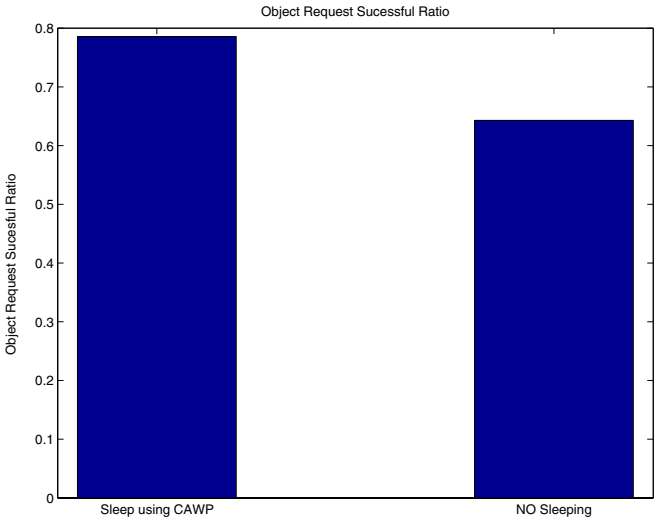

Fig. 3. Rate of successful file request.

Then we compare CAWP with AWP. The main difference between CAWP and AWP is the idea of community formation. As shown in Figure 4 CAWP increases the file request successful rate. It should be noted that only using AWP means users with similar interest may use di

WSF, they do not become active together, although they will have the chance to contact when overlapping time slots come. But this means the file request transmitted by a user in a time slot may not be heard by the one who has the same interest and holds the file if this holder is sleeping when the request packet arrives. CAWP satisfies over 90% of file searching request while in AWP only 80% of requests are satisfied.

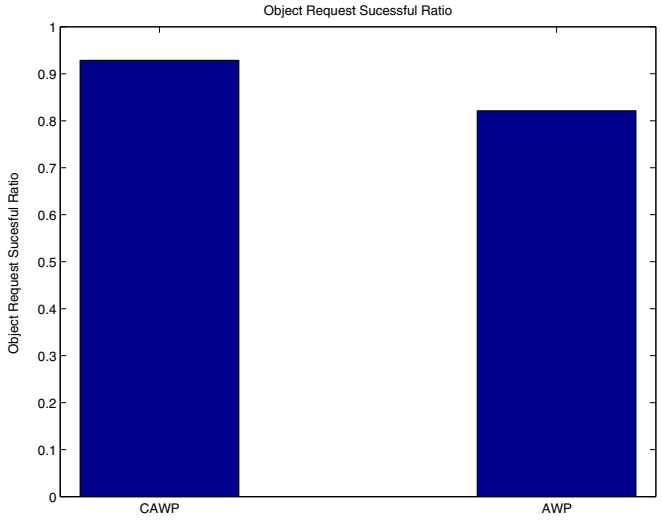

Fig. 4. Rate of successful file request.

For the same reason mentioned above, CAWP is able to reduce the delay involved in file transfer since users holding the requested file are supposed to be in the same community as, and thus become active together with the original requester of the file. As shown in Figure 5, CAWP could reduce the delay by around 20 seconds (one-third of the total transmission time).

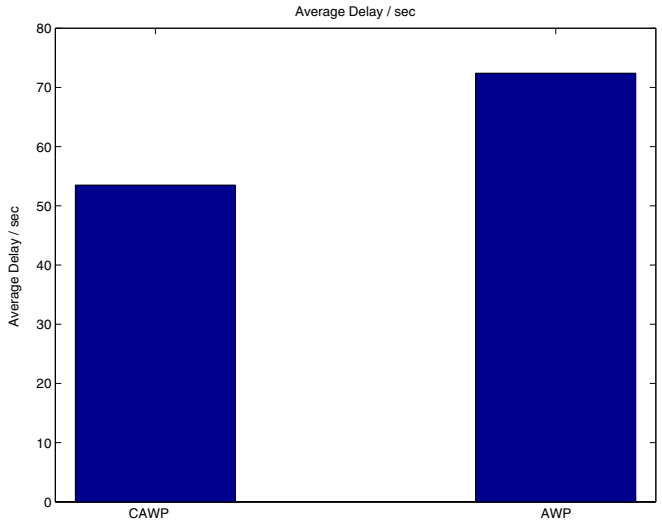

Fig. 5. File transmission delay

More future work on the use of similarity measurement technique (e.g., the use of coefficient other than Jasccard coefficient) is to be done to increase the improvement.

V. Conclusions

We have proposed a new sleeping protocol, namely Community-Based Asynchronous Wakeup Protocol (CAWP), which can increase the energy efficiency and reduce the rate of file request failure of a wireless P2P file sharing network, which is a yet-to-be intensively explored area. Our work sheds some insight into this important kind of mobile interaction system in the next generation wireless communication. With simulations we demonstrate that the protocol makes a step forward in achieving the goal of mobile and ubiquitous P2P data networks.

References

[1] V. Bharghavan, A. Demers, S. Shenker and L. Zhang, " MACAW: A media access protocol for wireless LAN's," *ACM SIGCOMM Computer Communication Review,* vol. 24, pp. 212–225, Oct. 1994.

[2] *BitTorrent*, `http://bitconjurer.org/BitTorrent/`, 2004.

[3] A. Cerpa and D. Estrin, "ASCENT: Adaptive self-configuring sensor networks topologies," *IEEE Transactions on Mobile Computing*, vol. 3, pp.272–285, July–Sept. 2004.

[4] J. H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," *Proc. IEEE INFOCOM 2000*, vol. 1, pp. 22–31, Mar. 2000.

[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*, vol. 8, pp. 481–494, Sept. 2002.

[6] *ICQ*, `http://www.icq.com/`, 2004.

[7] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *IEEE Std 802.11b–1999*, 1999.

[8] J. Elson, L. Girod and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *Proc. 5th Symposium on Operating System Design and Implementation*, pp. 147–163, Dec. 2002.

[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 2001*, vol. 3, pp. 1548–1557, Apr. 2001.

[10] P. G. Flikkema and B. West, "Clique-based randomized multiple access for energy-efficient wireless ad hoc networks," *Proc. IEEE WCNC 2003*, vol. 2, pp. 977–981, Mar. 2003.

[11] R. Kravets and P. Krishnan, "Application-driven power management for mobile communication," *ACM Wireless Networks*, vol. 6, pp. 263–277, July 2000.

[12] Q. Li, J. Aslam and D. Rus, "Online power-aware routing in wireless ad-hoc networks," *Proc. ACM MOBICOM 2001*, pp. 97–107, July 2001.

[13] J. G. Markoulidakis, G. L. Lyberopoulos, D. F. Tsirkas and E. D. Sykas, "Mobility modeling in third–generation mobile telecommunications systems," *IEEE Personal Communications*, vol. 4, no. 4, pp. 41–56, Aug. 1997.

[14] NEASIA, "Short battery life is top complaint among business mobile device users," available from http://www.nikkeibp.asiabiztech.com/wcs/frm/leaf?CID=onair/asabt/news/146420, 2004.
[15] V. Paruchuri, S. Basavaraju, A. Durresi, R. Kannan, and S. S. Iyengar, "Random asynchronous wakeup protocol for sensor networks," *Proc. IEEE BROADNETS 2004,* pp. 710–717, Oct. 2004.
[16] T. S. Rappaport, *Wireless communications: principles and practice*, Prentice Hall, 1996.
[17] G. Salton and M. J. McGill, *Introdution to modern information retrieval*, McGraw-Hill, 1983.
[18] S. Singh, M. Woo and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," *Proc. ACM MOBICOM 1998*, pp. 181–190, Oct. 1998.
[19] S. Singh and C. S. Raghavendra, "PAMAS—Power aware multi-access protocol with signalling for ad hoc networks," *ACM SIGCOMM Computer Communication Review*, vol. 28, pp. 5–26, July 1998.
[20] Yu-Chee Tseng; Chih-Shun Hsu; Ten-Yueng Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," *Proc. IEEE INFOCOM 2002*, vol. 1, pp.200–209, June 2002.
[21] X. Yang and Gustavo de Veciana, "Service capacity of peer to peer networks," *Proc. IEEE INFOCOM 2004*, Mar. 2004.
[22] W. Ye, J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," *Proc. IEEE INFOCOM 2002*, vol. 3, pp. 1567–1576, June 2002.
[23] R. Zheng, C. Hou and S. Lui, "Asynchronous wakeup for ad hoc networks," *Proc. ACM MobiHoc 2003*, pp. 35–45, June 2003.