

Energy-Efficient Data Broadcasting in Mobile Ad-Hoc Networks^{**}

Le Gruenwald, Muhammad Javed, Meng Gu

The University of Oklahoma, School of Computer Science, Norman, OK 73019

Abstract

Energy saving is the most important issue in wireless mobile computing due to power constraints on mobile units. Data broadcasting is the main method of information dissemination in wireless networks as its cost is independent of the number of mobile hosts receiving the information. A number of data broadcasting techniques have been proposed for mobile wireless networks, where servers have no energy restrictions, but little research has been done to address the issue of data broadcasting in mobile ad-hoc networks where both servers and clients are nomadic. In this paper, we propose two groups of broadcast scheduling algorithms called adaptive broadcasting and popularity based adaptive broadcasting that consider time constraints on requests as well as energy limitation on both servers and clients. We also present the simulation experiments that compare the algorithms.

1. Introduction

In a mobile ad-hoc network (MANET), mobile units can communicate with each other directly via wireless links in the absence of a fixed wired infrastructure [6][10]. MANET is different from a wireless mobile network which usually consists of a static wired part, in which fixed hosts and base stations are interconnected through a high speed wired network, and a mobile wireless part, in which mobile units communicate with the base stations through wireless connections. A base station can only communicate with the mobile units moving within its coverage area called cell. Mobile units can communicate with each other only through at least one base station. Mobile units run on batteries while base stations are supplied by stable system power from static networks. In MANET, every mobile unit can move freely and communicate directly with another mobile unit as long as that mobile unit is in its communication coverage area.

In a mobile computing environment, bandwidth and power limitations impose significant restrictions on data management [8]. These limitations require frequent disconnection and inspire the need for energy-efficient data access methods. While research in mobile computing has received growing interest in recent years due to the large number of potential applications, research in mobile ad-hoc network is still in its infancy.

Data broadcasting is considered as a main method of information dissemination in mobile wireless networks and can also be adopted for information distribution in mobile ad-hoc networks. However, in a MANET environment, both servers and clients are mobile. Therefore, energy conservation issues must be considered in developing broadcasting strategies for both servers and clients. In this paper, we propose two groups of broadcast scheduling algorithms for MANET called adaptive broadcasting and popularity based adaptive broadcasting which address the issues of client and server energy limitation and timing constraints on requests. The performance of the proposed algorithms is analyzed using simulation.

This paper is organized as follows. Section 2 reviews related work. Section 3 describes our architecture. Sections 4 and 5 present our broadcasting algorithms. Section 6 analyzes the simulation experiments. Section 7 provides conclusions and future research.

2. Related Work

The existing studies on data broadcasting are limited to mobile network environments, where only clients are nomadic. They attempt to save energy for clients but not for servers. Below is a brief discussion of the three most recent data broadcast algorithms for mobile networks.

Askoy et. al. [1] have presented a large-scale low-overhead on-demand broadcasting model called RxW (Requests time Wait). In RxW, at each broadcast tick, the

^{**} This research is supported in part by National Science Foundation grant No. EIA-9973465

server chooses an item with the highest value of $(R * W)$ where R is the number of outstanding requests and W is the waiting time for the first request. The entry for this data item is then removed from the queue that keeps track of the number of requests and earliest request time for data. The algorithm makes no assumptions regarding access probabilities of data items. However, the size of the queue is equal to the size of the database, therefore, large databases will require a significant overhead in terms of time to find the highest value of $R * W$ and space to store frequency and time.

Xuan et. al. [16] have proposed two on demand broadcast scheduling strategies, which consider deadlines attached to requests to decide the next item to be broadcast. In the first strategy, the server always broadcasts a request with the earliest deadline first (EDF) and every request is scheduled once no matter how frequently the same request is encountered. The second strategy is called EDF-batch, in which the server broadcasts an item according to EDF but after broadcasting, it removes the other entries for the same request.

Datta et. al. [8] have proposed the protocols that dynamically change the contents of broadcast according to client requests. In these protocols the broadcast data and index are organized using the $(1,m)$ indexing strategy [9]. The server decides the data item to be included in the broadcast on the basis of priority, which is given by $(IF^N * PF)$, where IF is Ignore Factor, PF is the Popularity Factor, and N is an adaptive scaling factor. PF makes sure that most popular data items are included in the broadcast and IF makes sure that less popular long neglected data items are also broadcast. In the Constant Broadcast Size protocol, the broadcast size is fixed, and after each broadcast cycle, the server calculates the priority of data items, sorts them in descending order of priority and adds them to the broadcast until it is full. In the Variable Broadcast Size protocol, all the items with $PF > 0$ are added to the broadcast set.

3. Architecture

Depending on communication capacity, computing power, disk storage, size of memory and energy limitation, MHs can be classified into two groups: 1) computers with reduced memory, storage, power and computing capabilities called *Small Mobile Hosts (SMHs) or Clients*, and 2) classical workstations equipped with more storage, power, communication and computing facilities than the SMHs called *Large Mobile Host (LMHs) or Servers*. Each MH has a radius of influence. An MH can directly communicate with other MHs within its radius of influence. If two MHs are outside each other's radius of influence, they will be able to indirectly

communicate with each other in multiple hops using intermediate MHs [2] as shown in Figure 1.

To conserve energy, an MH can operate in Active, Doze or Sleep mode. In active mode, the MH's CPU is working and its communication device can transmit and receive signals. In doze mode, the CPU of the MH will be working at a lower rate, but it can receive and examine messages from other MHs; so the MH can be awoken by a message from other MHs [3]. In sleep mode, both the CPU and the communication device of the MH are suspended. Due to energy and storage limitations, we will assume that only LMHs will store the whole DBMS (Data Base Management System) and SMHs will store only some modules of the DBMS that allow them to query their own data, submit transactions to LMHs and receive the results.

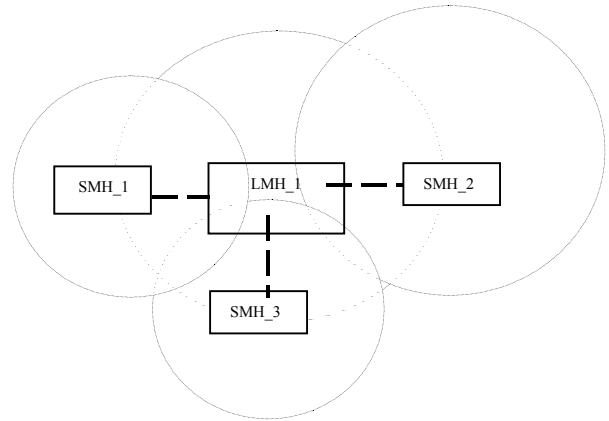


Figure 1. Architecture

This proposed architecture can be used to support many applications, such as battlefields and disaster recovery. In battlefields, portable computing devices with soldiers will work as SMHs while computers stored in tanks and humvees will work as LMHs. Note that tank/humvee computers are usually battery-powered so that they can easily be moved to other tanks/humvees, and these vehicles do not have to be turned on in order for their computers to function. These LMHs can store tactical information regarding enemy and other units in a database and the SMHs can communicate with the LMHs to get this information (e.g. location and strength of enemy units). In a disaster recovery operation, the palmtops carried by rescuers act as the SMHs and the computers in mobile hospitals can be viewed as the LMHs. The LMHs can keep the information about medical equipments in their databases, and the SMHs can query about the inventory.

4. The Proposed Adaptive Broadcast Scheduling Algorithms

The ultimate goal of broadcast scheduling is to minimize the average response time for clients' requests, but for MANET, broadcast scheduling must also address the issue of energy restriction on both servers and clients. In MANET, clients' requests will typically require fast responses with short soft deadlines [14].

Broadcast scheduling among the distinguished servers is possible in the assumed MANET. In MANET the movement patterns of the servers and the clients and the network topology are unpredictable. The simplest case is that in a certain area, there exist one server and multiple clients. These clients can communicate with the server if the server's radius of influence covers them. However, in the more complicated and practical cases, multiple servers and multiple clients exist in some areas. A client may communicate and receive broadcast from one or more servers depending on its location. Therefore, it is not energy-efficient that these servers broadcast the same data. On the other hand, a client who can communicate with one server only will prefer to have that server broadcast as many data items as possible so that it can benefit from broadcasting.

Based on the above considerations, we develop the broadcast scheduling algorithms applicable to the case where multiple servers and multiple clients exist in a certain area. We assume full data replication among the servers and the data in a server's local database can be divided into two groups: frequently requested data, called hot data and less frequently requested data, called cold data. The server's power level and location information are two hottest data items that are broadcast by all the servers. All requests from the clients are "read-only". Each request asks for one data item only.

Each server keeps the power level and location information of all the other servers in the network in its local database. It does periodic broadcasting for some hot data items only and offers cold data on-demand in one broadcast cycle. We assume one broadcast cycle takes T units of time and T is large enough to broadcast all hot data as in traditional periodic broadcast. A broadcast cycle should also have some time slots to serve data on-demand. The on-demand data items fill this part spontaneously. Thus, a broadcast cycle, in terms of time, consists of time to broadcast the indices and data, the time for on-demand data and maybe some idle time. The $(1, m)$ indexing scheme [9] is used in our algorithm. The periodic broadcast size varies within the range of T . The algorithm consists of two parts: Part 1 is performed by the leader server and Part 2 is performed by all the servers in the system.

4.1. Part 1: Leader Protocol

When a broadcast cycle starts, the server with the highest power level will be the leader and schedule data broadcast. The other servers will follow the instructions from the leader to arrange their broadcasting schedules. The idea is that the server with the highest power level will try to schedule data broadcasting for the other servers. It will partition the hot data, except for the power level and location information, into portions. The amount of data in each portion, which will be assigned to a server for broadcasting, will depend on the power level of the server. The server with less energy available broadcasts a smaller portion of hot data but these data are more frequently accessed than those broadcast by the server with more energy. This way the server with less energy will not have to accommodate too many on-demand requests. The algorithm is presented below.

Adaptive Broadcast Scheduling Algorithm (Part 1 - Leader Protocol): for the server with the highest power level:

Let $P_1, P_2, \dots, P_i, \dots, P_n$ denote the power levels of all n servers. i represents a server's id number. Let $d = \{d_1, d_2, \dots, d_j, \dots, d_N\}$ denote the set of the more frequently requested data items, except power levels and locations of servers. N is the total amount of data items in set d . Sort P_1, P_2, \dots, P_n in increasing order, with $P_1 < P_2 < \dots < P_n$;

If the power level of the server who is running this algorithm is the maximum power level among all the servers $\{$

for $i = 1, \dots, n$

Calculate the ratio $R_i = P_i/P$, where P is the sum of P_1, P_2, \dots, P_n ;

Sort d_j in set d in decreasing order of the request frequency f_1, f_2, \dots, f_j , which are associated with d_1, d_2, \dots, d_N respectively;

for $i = 1, \dots, n$

*Assign the first $R_i * N$ amount of data items in set d to server i ;*

}

else

Send an appointment message to the server with the highest power level to notify it to start this algorithm for the next broadcast cycle;

Initially, each server will broadcast its power level and location to start the communication. The server with the highest power level becomes the leader and runs this algorithm every C broadcast cycles. The leader may change since after some time, the leader may have lower energy than other servers because of its heavier duty of

broadcasting. To restart the algorithm, a routing algorithm in the mobile ad-hoc network is also assumed to be available [10].

4.2. Part 2: All Server Protocol

All the servers in the network will change their broadcast schedules after receiving a message from the leader. According to our scheme, the broadcast content of a server is only a portion of the more frequently requested data so the number of requests for data items not in the broadcast content from the clients may increase. We have to dynamically change a server's broadcast content so that broadcast data can satisfy as many requests as possible. Consequently, the server will save its energy by not serving the data on demand for many identical requests. A client will also save energy by going into doze mode to wait for the broadcast and directly retrieving data from the "air". The client can follow the provided index to find the data items it wants if the broadcast can satisfy its deadline. Otherwise, the client has to submit a request to the nearest server available.

We assume that the server always keeps track of the request frequency (RF) for each data item. The Exponential Weighted Moving Average (EWMA) method [11] given below will be used for calculating request frequencies.

$$f_{i(n)} = (f_{i(n)} + \alpha * f_{i(n-1)} + \alpha^2 * f_{i(n-2)} + \dots + \alpha^{n-1} * f_{i(1)}) / S$$

Here $S = (1 + \alpha + \alpha^2 + \dots + \alpha^{n-1})$; f_i is the RF of data item i , $f_{i(n)}$ is the RF of data item i in the current broadcast cycle n . α amounts to an exponential reduction in the weight

If the RF for a particular hot data item is higher than that of a data item in the current broadcast, this hot data item should be included in the next broadcast. There are two alternatives to accomplish this. In alternative 1, this data item can replace the data item with the least RF in the current content of the periodic broadcast. After some time, the content of the periodic broadcast is getting more and more matching with the clients' access patterns through this self-learning process.

Adaptive Broadcast Scheduling Algorithm (Part 2, Alternative 1): for all servers

Any server in the assumed mobile ad-hoc environment, S_i , will run this algorithm.

For each broadcast cycle do the following:

if S_i receives a broadcast assignment message from another server which is running our proposed adaptive broadcast scheduling algorithm (part 1) {
if the number of data items in the assignment is less

than the number of data items in the current broadcast content

cut down the number of data items in broadcast to the number of data items in assignment by "drop-tail";

else

add data items from the more frequently requested data group to the broadcast, up to the number of data items in the assignment;

}

if S_i receives a data request from a client {

if S_i can satisfy the request within the deadline serve the data on demand based on "earliest deadline first" rule;

else

reject the request;

}

for each data item that was requested in this broadcast cycle {

calculate the request frequency for this data item;

if the request frequency of this data item i is greater than the least request frequency of a data item in the current broadcast content and this data item is in more frequently requested data group {

replace the data item with the least request frequency in the current broadcast with the newly requested data item i ;

sort the data items in broadcast content

according to the decreasing order of request frequency;

reconstruct the index;

}

}

In alternative 2, when the RF of a hot data item gets higher than the RF of a data item in the current broadcast it is simply added to the broadcast.

The adaptive broadcast scheduling algorithm, which consists of part 1 and part 2 with alternative 1, is called the *adaptive replacement broadcast scheduling algorithm* (ARBSA). The adaptive broadcast scheduling algorithm which consists of part 1 and part 2 with alternative 2 is called the *adaptive addition broadcast scheduling algorithm* (AABSA).

5. The Proposed Popularity Based Adaptive Broadcast Scheduling Algorithms

The EWMA based algorithms presented above require a global update when a request arrives. This

requires a lot of communication among servers resulting in excessive energy consumption. Also data items to be broadcast are selected according to the global RF ignoring requests submitted to a server locally. Request deadlines and client movements are also not considered in calculating request frequencies. If the leader server leaves the network or fails, it is not clear who will run Part 1 of the algorithms. Servers that do not have any clients may waste energy since they are not required to switch to doze mode. Also, EWMA is used to calculate the RF of a data item. EWMA is a good forecasting technique where more recent values have greater influence than older data values. Therefore, if request frequencies are calculated using EWMA, data items requested in the current broadcast cycle will have more influence on the calculated request frequencies. Therefore, some of the less popular requests that were requested earlier may starve or the clients who requested the data item may leave the area.

To address the above-mentioned drawbacks, we propose two algorithms based on popularity factor [8]. They make the same assumptions about the network and the database as the EWMA based algorithms.

5.1. Broadcast Assignment

When a broadcast cycle starts, the server with the highest power becomes the leader and assigns the amount of data to be broadcast by each server according to the server's power level like the EWMA based algorithms. A server with less power will broadcast fewer data but more frequent data. The main difference is that the leader server informs the other servers only about the amount of the data they should broadcast, and the data items to be broadcast are taken by these servers from their local databases. The other difference is that the leader checks if a server will be able to broadcast the assigned data or not. If a server cannot broadcast the assigned data then it will broadcast the data and index from the previous broadcast so that it does not have to sort data and reconstruct index. A broadcast cycle in terms of time will consist of time to broadcast data and index and time to serve data on demand. A server goes into doze mode after broadcasting data and index and serving data on demand until its time to start the next broadcast cycle. The server is allowed to receive clients' requests during this period, but the number of such requests is expected to be low since data that has high access frequency is broadcast by the server. Similar to the EWMA based algorithms, the popularity based adaptive broadcast scheduling algorithms consist of two parts: leader protocol and all server protocol.

5.2. Part 1: Leader Protocol

Let $P_1, P_2, P_3, \dots, P_j, \dots, P_n$ be the power levels of

the servers $S_1, S_2, S_3, \dots, S_j, \dots, S_n$ currently in the network, 'n' is the number of servers currently in the network.

Let 'j' be amount of energy required to broadcast one data item and 'd' be the total number of hot data items.

Sort the power levels in ascending order such that $P_1 < P_2$ where P_1 and P_2 are power levels of S_1 and S_2 and the power level of S_1 is less than S_2

If the server running this algorithm has the highest power level among all the servers {

calculate the sum of power levels of all the servers 'P';

for $i = 1, \dots, n$ {

calculate the amount of data to be broadcast by server S_i as $X_i = (P_i / P) * d$;

if $J * X_i$ is greater than P_i then

send a message to P_i , informing that it has to broadcast the previous broadcast set;

else

assign the 'X_i' amount of data to be broadcasted by server S_i ;

}

appoint a backup leader server, which has the 2nd highest power level;

}

else

send an appointment message to the server with the highest power level to start this algorithm

The leader server will run the above algorithm whenever a change occurs in the sorted list of power levels maintained by the leader. The back up leader is assigned so that if the leader moves out of the network or fails, it will act as the new leader.

5.3. Part 2: All Server Protocol

The servers, which accept the instructions from the leader, change their broadcast schedules accordingly. These servers will sort the hot data items in descending order of their request frequencies kept in their local database. After sorting, the servers will take the amount of data assigned by the leader from their local database and start broadcasting it. Similar to the argument we have made for ABRSA's and AARSA's part 2, the servers must dynamically change their broadcast content so that broadcasting can satisfy most of the requests. In these algorithms, the servers keep track of the number of requests for all the hot data items and use this information to calculate the popularity factor (PF) of data items as defined below [8].

The popularity factor PFA of a data item A is defined as the number of clients interested in A at a particular time T [8]. When a client requests for A, the popularity factor is incremented by 1 and the system records the

corresponding time TSA . A corresponding decrement of 1 will be performed at $TSA + RL$ to reflect the departure of the client who requested A , where *residence latency* (RL) is defined as the average time a mobile unit spends in a cell. RL will be computed *a priori* based on the advance knowledge of user movement patterns and cell geography [8]. Requests have deadlines, so whenever the deadline of a request for A expires, a decrement of 1 is performed to the popularity factor. The idea is to include only those data items in the broadcast that can satisfy the requests of clients currently in the cell within the associated deadlines. This way the server has to keep track of only those requests, which were received during the interval $current_time - RL$ [8].

If the popularity factor of a *hot* item becomes higher than that of a data item in the current broadcast then it is added to the broadcast using the two alternatives described in Section 4.2. Below is the algorithm's part 2 using alternative 1.

Real Time Popularity Based Adaptive Replacement Broadcast Scheduling Algorithm (Part2, Alternative 1):

Every server S_i will run this algorithm.

For each broadcast cycle do the following:

If S_i receives a message to broadcast the previous broadcast set

broadcast the previous broadcast set and index

else {

sort the hot data items in descending order according to popularity factor;

take the first R_i amount of data from the local database;

while a new assignment message is not received {

broadcast the data using (1, m) indexing technique;

if S_i has received a data request for a cold item

serve data on demand based on the basis of earliest deadline first;

for each data item requested in (current_time - RL) period {

calculate the popularity factor;

if the popularity of this data item becomes greater than the popularity of any of the data items in the current broadcast content {

replace the data item with the least popularity factor in the current broadcast with this data item;

reconstruct index;

}

}

if S_i did not receive any requests in the previous C broadcasts

broadcast a greeting message to clients and wait for a reply;

if no reply received

switch to doze mode until a request is received;

if this is the backup leader

if the power level of the leader was received continue;

else

assume control and run the algorithm part 1;

}

}

The proposed popularity based algorithm that uses alternative 1 for part 2 is called the real-time Popularity Based Adaptive Replacement Broadcast Scheduling Algorithm (PBARBSA), while the one that uses alternative 2 for part 2 is called the real-time Popularity Based Adaptive Addition Broadcasting Scheduling Algorithm (PBAABSA).

6. Simulation Model

6.1. Simulation Model Description

In order to measure the performance of the proposed algorithms, simulation experiments are conducted. The simulation model is implemented using Visual Slam as the simulation language and Awesim as the simulation tool [13]. Global transactions are created with a fixed inter-arrival time using an exponential distribution. Each transaction has a creation time, ID, data item requested, and ID of the mobile unit requesting the data. The data item requested is generated using the Gaussian random distribution and is assigned randomly to a mobile unit. Then the data item generated is checked against the broadcast of the servers within the area of influence of the mobile unit. If the data item is found it is downloaded, otherwise a request is issued and the mobile unit goes into doze mode. And it tunes into subsequent broadcasts to find out if the requested data item is included in the broadcast or not. If the request was made for a cold data item then the client tunes into the on demand portion of the data to find out whether the desired data item is being served on demand. The mobile servers are defined as resources in the simulation model.

The attributes associated with each server are ID, X and Y coordinates, energy level, radius of influence, time in active mode, and time in doze mode.

6.2. Performance Metrics

Below are the four performance metrics we have used to evaluate the performance of the proposed algorithms:

Energy Consumed by Mobile Units

*(Time Spend by Mobile Unit in Active Mode * Energy Consumed per unit Time in Active Mode) + (Time Spend by Mobile Unit in Doze Mode * Energy Consumed per Unit Time in Doze Mode)*

Energy Consumed by Server

*(Time Spend by Server in Active Mode * Energy Consumed per Unit Time in Active Mode) + (Time Spend by Server in Doze Mode * Energy Consumed per unit Time in Doze Mode)*

Access Time

The time interval between the submission of a request to the server and the moment the client receives a reply.

Broadcast Hit Ratio

Broadcast Hit Ratio =

$\frac{\text{The Total Number of Requests Satisfied by Broadcast}}{\text{The Total Number of Requests Generated by Clients}}$

6.3. Simulation Parameters

Table 1. Static parameters of simulation model

Parameter	Meaning	Default
Bandwidth LMH	Bandwidth of wireless medium [16]	1 Mbps
Bandwidth SMH	Bandwidth of wireless medium [10]	100 kbps
CPU_power_LMH	CPU Power of LMH [5]	140 MIPS
CPU_power_SMH	CPU Power of SMH [9]	4 MIPS
LMH_power	LMH Power Dissipation Rate [12]	170W per hour
Mem_access_time	Main Memory [5] access time per word	0.00018 ms
Data_item_size	Data Item Size	25 KB
Req_size	Size of request	1 KB
SMH_power	SMH Power Dissipation Rate [9]	7W per hour
Word_size	Number of bytes per word [5]	8
Radius_SMH	Radius of influence of SMH	100 unit
Radius_LMH	Radius of influence of LMH	200 unit
Inter-arrival Time	Time between requests	0.02
D_SIZE	Database Size	1800
Hot_data	Number of hot data Items	20% of database

Table 2. Dynamic parameters of simulation model

Parameter	Meaning	Default	Range
Num_LMH	Number of LMH	3	3-5
Ratio_cold_hot	Ratio of cold to hot requests	5 % of database	5% to 35%

The simulation parameters are summarized in Tables 1 and 2. The number of SMHs in the system is 1000. The positions of LMH and SMH are assumed to be inside a 500unit X 500unit square region. The initial locations (i.e. X and Y coordinates) of all the MHs are obtained using a random distribution within a 500 X 500 square units region. The Gaussian distribution [15] is used to generate the workload. Sample runs were performed initially to get the hot data items.

6.4. Simulation Results

The experiments were conducted using 3 servers and 4 servers. In Fig. 2-5, ARBSA3 and ARBSA4 mean Adaptive Replacement Broadcast Scheduling Algorithm tested using 3 servers and 4 servers, respectively. Similarly for The Adaptive Addition Broadcast Scheduling Algorithm (AABSA), the real time Popularity Based Adaptive Replacement Broadcast Scheduling Algorithm (PBARBSA), and the real time Popularity Based Adaptive Addition Broadcast Scheduling Algorithm (PBAARBSA).

Broadcast Hit Ratio

Fig. 2 shows the effect of number of requests for cold data on the broadcast hit ratio. Broadcast hit ratio is obtained by dividing the number of *hot* data requests satisfied by the periodic broadcast by the total number of requests. The broadcast hit ratio for addition algorithms is higher than that for the replacement algorithms. In the addition algorithms, a data item is added to the broadcast if its RF becomes higher than that of any of the data items in the current broadcast. Therefore, the periodic broadcast contains more of the hot data items and most of the requests are satisfied by the periodic broadcast. In the replacement algorithms, the periodic broadcast size is fixed, so in order to include a data item in the broadcast, one of the data items in the current broadcast has to be dropped. It is possible that the dropped data item is more popular than the added data item; so the number of broadcast hits decreases. Fig. 2 also shows that the popularity-based algorithms provide a better broadcast hit ratio than the EWMA based algorithms. Since the popularity based algorithms consider local RF in deciding

which data items to include in the broadcast. So it is possible to include the requested data items in the broadcast more quickly than in the case of the EWMA based algorithms, in which the RF of a requested data item has to compete against the global request frequencies of the data items currently in the broadcast. This makes it difficult to include requested data items in the broadcast.

Another important observation from Fig. 2 is that, as the number of servers increases, the broadcast hit ratio increases. This is possible because clients can receive broadcasts from multiple servers and the likelihood that one of the servers is broadcasting the desired data item also increases. The figure also reveals that as the number of requests for hot data increases, the hit ratio drops in the addition algorithms simply because fewer requests for hot data items are being generated. But in the case of the replacement algorithms, the ratio increases when the number of requests for hot data items decreases up to a certain point and then it starts decreasing again. This happens because initially when there are too many requests, the servers cannot schedule all of them since the broadcast size is fixed, but as the rate drops, the servers are able to schedule more data items in the broadcast. Later on, the rate drops simply because there are not enough requests coming to increase the RF of less popular hot data items to a level that they can be included in the broadcast.

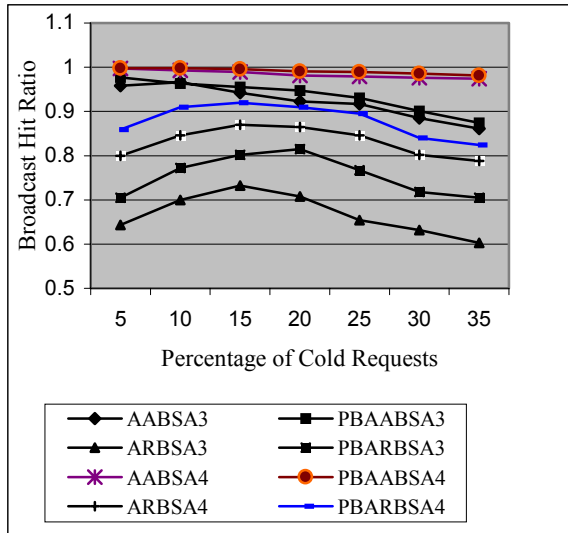


Figure 2. Broadcast hit ratio vs. percentage of cold requests

Power Consumed by Clients

Fig 3 shows that the energy consumed by clients increases as the number of requests for cold data items increases. A client can issue a request only after finding out that the desired data item is not in the broadcast of all

the servers within its area of influence. In the proposed algorithms, only hot data items are served through periodic broadcast; so for cold data items, clients must issue a request and then monitor the periodic broadcast and on demand broadcast until the desired data item arrives. So, clients have to spend more time in active mode if the number of requests issued for cold data items is high. The figure also shows that the addition algorithms cause clients to consume more energy than the replacement algorithms. The reason is that, as more and more data items are added to the periodic broadcast, the index size becomes larger, and thus the clients consume more energy as they have to search through a larger index. At the same time, as the number of data items in the periodic broadcast increases, more requests are satisfied by periodic broadcast, and thus energy consumed by clients decreases. Higher energy consumption by clients occurs when the addition algorithms are used suggests that searching a larger index offsets the energy savings achieved from including more data items in the broadcast.

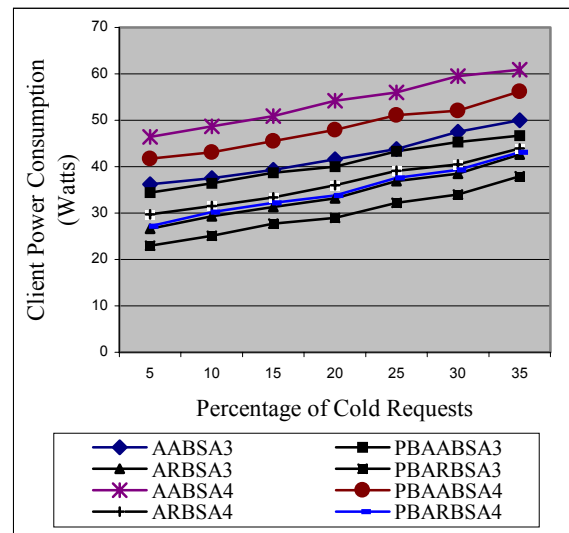


Figure 3. Total power consumed by clients vs. percentage of cold requests

Also obvious from Fig. 3 is that if there are more servers in the area of influence of clients, then energy consumption by clients increases. This occurs because a client has to check the broadcast of multiple servers to get the desired data item, so it has to spend more time in active mode, which results in higher energy consumption. Finally, the energy consumed by clients is less when the popularity based algorithms are used. These algorithms consider the local request frequencies to schedule data items for periodic broadcast. Therefore, it is easier to include a data item in the broadcast and clients have to tune into fewer subsequent broadcasts to obtain the desired data items. The difference in energy consumption

between the popularity based algorithms and the EWMA based algorithms increases when the replacement algorithms are used. In the replacement algorithms, the method used to calculate RF has more influence on performance since the broadcast size is fixed, and a scheduling algorithm that can fill the periodic broadcast with data items that are more desired by clients will perform better.

Access Time

Fig 4 shows that when the number of requests for the cold data items increases, the average access time for hot data decreases. When the number of requests for cold data items increases, there is a corresponding decrease in the number of requests for hot data items. Therefore, it becomes faster for a server to include data items in the periodic broadcast as fewer data items are waiting to be scheduled in the broadcast. This figure also depicts that the average access time is lower when there are more servers in the network as this increases the likelihood that one of the servers is broadcasting the desired data item. Therefore, the clients can get data faster and the average access time is lower. It is also obvious from Fig. 4 that the addition algorithms provide a lower average access time since they keep on adding data items to the broadcast, which results in the requests being satisfied faster. Also the popularity based algorithms provide better access time than the EWMA based algorithms since with the former algorithms, it is faster to include a requested data item in the broadcast, which results in the clients requests being satisfied faster.

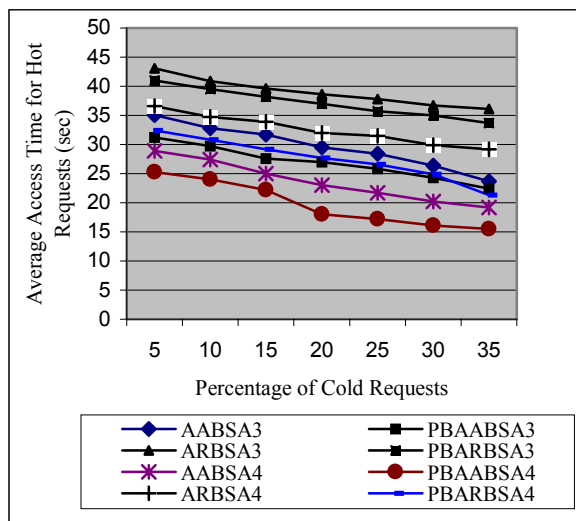


Fig. 4. Average access time for hot requests vs. percentage of cold requests

Power Consumed By Servers

Fig. 5 shows the average power consumed by a server in the network. The behavior of servers is very complex. The energy consumed by servers depends upon the following three factors: energy required to broadcast data items, energy required to handle requests, and energy required to perform computations in doze mode. The average energy consumption by a server is relatively stable although the mix of hot and cold requests is being changed. The reason for getting the stable results is that when the number of cold requests is increased, there is a corresponding decrease in the number of hot requests, so energy saved by not receiving hot requests is balanced by receiving cold requests. The servers have to perform fewer calculations when changing the broadcast content since there are fewer hot requests in the queue. But because there are more requests in the on demand queue, the energy saved by performing fewer calculations on hot requests is balanced with that saved by performing more calculations on the cold requests.

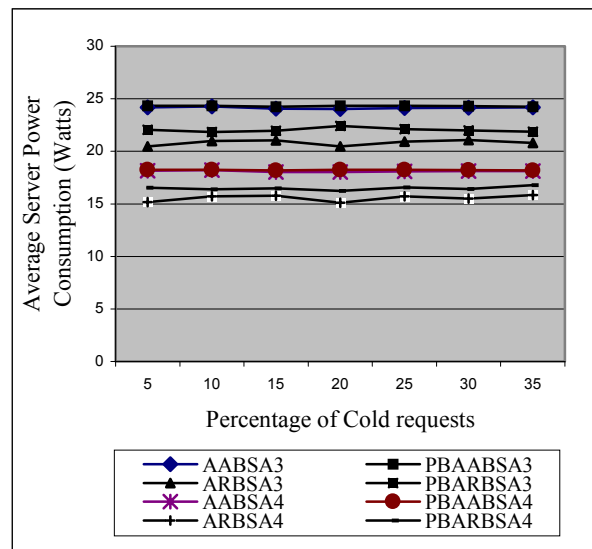


Figure 5. Average power consumed by a server vs percentage of cold requests

In performing calculations for energy consumption for servers, the cost involved in sending messages to other server has been ignored since a routing table must be available to perform these calculations. So when an update in the global table is performed, the cost involved in exchanging messages is ignored, but it is expected to be significant. This is one of the main reasons that the popularity based techniques perform poorly in terms of energy consumption. The other reason is that when its time to update the broadcast, in the EWMA based techniques, the server considers only those requests that

were received in the current broadcast cycle. But in the case of popularity based techniques, the server has to consider requests received during the interval *current time-RL*, which is larger than the broadcast cycle. Therefore, more requests have to be considered when scheduling broadcast in the popularity-based techniques than in the EWMA based techniques. Fig. 5 also shows that the average energy consumed by servers is higher when the addition algorithms are used. When a data item is added to the broadcast the broadcast size increases, so does the energy consumed by servers. Also, if there are more servers present in the network then the average energy consumed by a server is less since each server has to broadcast fewer data items as well as it has to serve fewer data items on demand.

7. Conclusions and Future Work

This paper has proposed data broadcasting algorithms for mobile ad-hoc networks (MANET). These algorithms consider three issues in scheduling broadcasts: energy consumed by mobile clients, energy consumed by mobile servers, and real-time constraints on client requests. The algorithms differ from each other based on how much data should be assigned to individual servers to broadcast, how request frequencies are computed, how the broadcast contents is dynamically changed, and how mobility of client and servers is considered. Simulation experiments conducted to study the performance of the proposed algorithms show that the popularity-based algorithms provide better broadcast hit ratio, access time, and client energy consumption, but more energy consumption than the EWMA based algorithms. The addition algorithms provide better broadcast hit ratio and access time but they are not energy-efficient for servers and clients. The replacement algorithms provide better energy consumption for clients and servers but do not give good broadcast hit ratio and access time. If there are more servers in a MANET, then the broadcast hit ratio, access time, and energy consumed by servers improve but the energy consumed by clients becomes worse.

The choice of an appropriate strategy will depend on the requirements of the environment. If access time and broadcast hit ratio are of prime importance, then the addition algorithms can be used. But if the main aim is to conserve energy consumed by clients and servers, then the replacement algorithms should be employed.

For future work, simulation experiments studying the effects of client and server mobility will be conducted.

8. References

- [1] Askoy, D. and Franklin, M., "Scheduling for Large Scale On-Demand Data Broadcasting", Proceedings of the 12th International Conference on Information Networking, pp. 656-659, Jan 1998.
- [2] Bandyopadhyay, S., and K. Paul, "Evaluating the Performance of Mobile Agent-Based Communication among Mobile Hosts in Large Ad-Hoc Wireless Network", MSWIM 1999.
- [3] Barbara D., T. Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments", ACM SIGMOD, May 1994.
- [4] Datta, A., VanderMeer, D. E., Kim, J., Celik, A., and Kumar, V., "Adaptive Broadcast Protocols to Support Efficient and Energy Conserving Retrieval from Databases in Mobile Computing Environments", A TimeCenter Technical Report, University of Arizona, April, 1997.
- [5] DECdirect Workgroup Solutions Catalog, winter 1993.
- [6] Hong X., et al, "A Group Mobility Model for Ad Hoc Wireless Networks", MSWIM 1999.
- [7] Imielinski, T. and Viswanathan, S., "Adaptive Wireless Information Systems", in Proceedings of SIGDBS (Special Interest Group in Database Systems) Conference, October 1994.
- [8] Imielinski, T., Viswanathan, S., and Badrinath, B. R., "Power Efficient Filtering of Data on Air", Proc of 4th Intl Conference on Extending Database Technology, Cambridge, March 1994, pp. 245-258.
- [9] Imielinski, T., Viswanathan, S., and Badrinath, B. R., "Data on Air: Organization and Access", IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 3, May 1997, pp. 353-372.
- [10] Ko, Y., Vaidya, N., "Location-Aided Routing (LAR) in Mobile Ad-hoc Networks", MOBICOM 1998.
- [11] Lam, K., Chan, E. and Yuen, C. H., "Data Broadcast for Time-Constrained Read-Only Transactions in Mobile Computing Systems", Proceedings of the First International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems, April 1999.
- [12] Michael, C. et. al., "Aspects of Energy Conservation on the St. George University of Toronto Campus", A Report by Division of Environment at the University of Toronto, 1996/97.
- [13] Pritsker A. Alan B, O'Reilly Jean J., "Simulation with Visual SLAM and Awesim", Systems Publishing Corp., 1999.
- [14] Ramamritham K., "Real-Time Databases", Distributed and Parallel Databases, Vol. 1, No. 2, April 1993, pp 199-226.
- [15] Stathatos, K., Roussopoulos, N. and Baras, J. S., "Adaptive Data Broadcast in Hybrid Networks", 23rd VLDB 1997.
- [16] Xuan, P., Sen, S., Gonzalez, O., Fernandez, J., and Ramamritham K., "Broadcast on Demand: Efficient and Timely Dissemination of Data in Mobile Environments", 3rd IEEE Real-Time Technology Application Symposium, 1997.