International Journal of Parallel, Emergent and Distributed Systems
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713729127

Data replication approaches for ad hoc wireless networks satisfying time constraints

To cite this Article: , 'Data replication approaches for ad hoc wireless networks satisfying time constraints', International Journal of Parallel, Emergent and Distributed Systems, 22:3, 149 - 161
To link to this article: DOI: 10.1080/17445760601122225
URL: http://dx.doi.org/10.1080/17445760601122225

The International Journal of Parallel, Emergent and Distributed Systems,
Vol. 22, No. 3, June 2007, 149–161

Taylor & Francis
Taylor & Francis Group

Data replication approaches for ad hoc wireless networks satisfying time constraints

XIAO CHEN*

Department of Computer Science, Texas State University, San Marcos, TX 78666, USA

(Received July 2006; revised September 2006; in final form December 2006)

Recent advances in technology have enabled the development of *ad hoc wireless networks* in which a collection of wireless mobile hosts (nodes) may move freely and self-organize into arbitrary and temporary network topologies in areas where communication infrastructures do not exist. In such networks, hosts frequently need data items to finish a certain task. The hosts chosen to store data items are called *data servers*. If the data items are stored too far away from the requiring host, multiple intermediate hosts are needed to relay the data. This creates a certain amount of delay which is not insignificant in time-critical applications. In this paper, we propose replica allocation schemes to let each host obtain data items in at most $k (k \geq 1)$ hops using minimal number of data servers in the network. Simulation results show that our schemes can generate a small number of data servers satisfying the k-hop time constraints.

Keywords: Ad hoc wireless network; Data server; Dominating set; Wireless transmission range

1. Introduction

Recent advances in technology have enabled the development of ad hoc wireless networks. An *ad hoc wireless network* is a special type of wireless network in which a collection of mobile hosts (nodes) with wireless network interfaces may move freely and self-organize into arbitrary and temporary network topologies, allowing people and wireless devices to communicate with each other in areas where communication infrastructures do not exist.

Since no special infrastructure is required, possible applications of ad hoc wireless network [12] include: Soldiers relaying information for situational awareness on the battlefield, business associates sharing information during a meeting; attendees using laptop computers to participate in an interactive conference; and emergency disaster relief personnel coordinating efforts after a fire, hurricane, or an earthquake. The other possible applications [9] include personal area and home networking, location-based services and sensor networks.

In ad hoc wireless networks, two mobile hosts can communicate with each other directly only if they are located closely together within each other's *wireless transmission range*. If two hosts that want to communicate are outside their wireless transmission ranges, they

*Email: xc10@txstate.edu

The International Journal of Parallel, Emergent and Distributed Systems
ISSN 1744-5760 print/ISSN 1744-5779 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/17445760601122225

could communicate only if other hosts between them can forward packets for them [9]. For example, mobile hosts A and C are outside each other's transmission range. If A and C wish to exchange packets, they may use host B to forward packets for them, if B is within the transmission ranges of both A and C.

In ad hoc wireless networks, hosts frequently need some data items to finish a certain task. A host that stores data items is called a *data server*. If the data item is stored too far away from the requiring host, multiple intermediate hosts are needed to relay the data, which causes a non-neglectable delay for some time-critical applications such as the ones in the military. A possible solution is to replicate data items at multiple data servers in the network. *Data replication* has been discussed by a number of papers [3–8,10] to serve the purpose of improving data accessibility, that is, since mobile hosts move frequently, the network often disconnects, which leads to low data accessibility. In this paper, we use data replication to address the issue of data access delay. We use the number of hops to represent the access delay (time constraint). In a time-critical application such as a military mission, hosts must get the data within a time constraint. Depending on how critical the application is, it is ideal to bound each data access by $k (k \geq 1)$ hops. On the other hand, mobile hosts have limited storage capacity and it is not possible to store data replicas in all of them. So we should reduce the number of data servers in the network.

In this paper, our goal is to find data allocation schemes to let each mobile host obtain data items in at most k hops using minimal number of data servers in the network. We assume the network is connected at all times, and a mobile host can access data many times at any time and the movement of the hosts are irregular. Since there is a special scheme for $k = 1$, we discuss it independently in one section. Then we discuss three schemes for $k \geq 1$.

The rest of the paper is organized as follows: Section 2 discusses the scheme for $k = 1$, Section 3 presents three schemes for $k \geq 1$, and Section 4 is the conclusion.

2. A scheme for $k = 1$

2.1 Preliminaries

In this section, we discuss the case when $k = 1$. We define an ad hoc wireless network as a graph $G = (V, E)$, where V is the set of all mobile hosts and E is the set of all edges between pairs of hosts. If two hosts are within each other's transmission range, there is an edge between them in G. Also we assume the network graph is not a complete graph. If a network graph is complete, then any host can be the data server because every host can get data items from any host in one hop.

From the above description, our objective is to guarantee that each host get the data in at most one hop with the minimum number of data servers in the network. This problem is equivalent to finding the minimal connected dominating set in the network. A *dominating set* is a set of nodes such that any node in the network is a neighbor of some element in the set. That is, each host is either in the dominating set or a 1-hop neighbor of some host in the dominating set. So each host can get the data in at most one hop. Thus, the replica can be stored in the nodes in the dominating set. If the dominating set is minimal, the number of data servers in the network is minimal. However, computing a minimal size connected dominating set is NP hard. We have put forward some heuristic algorithms in Ref. [2]. In this

paper, we will adopt the best heuristic algorithm in Ref. [2] to find the hosts that we can store data replicas. In the following, we will explain in detail of our approach.

In the heuristic algorithm we propose in Ref. [2], we use 2-*hop neighborhood information*, that is, a host knows the information of its neighbors and the neighbors of its neighbors. This information can be obtained by regularly exchanging Hello messages containing lists of neighbors between hosts. The algorithms that use this kind of information are very attractive for ad hoc wireless networks since they need only local updates at each detected topology change and no global knowledge of the network topology is necessary. In our approach, it requires the knowledge of a total order of hosts. One can possibly use IP address of a host as an ID. So, a host is smaller than another host if it has a smaller ID. Actually, any total order on which all hosts agree can be used.

2.2 Our scheme

To find those hosts to store replicas in the network, our method only requires that each host compute a *multipoint relay set* (or MPR set for short) in its 2-hop neighborhood. An element in the MPR set is called a *multipoint relay* of that host. The dominating set of the whole network is the set of all MPR hosts. However, the network dominating set formed in this way has too many hosts. Then we use two rules to get rid of some hosts. The resulting dominating set is the minimal dominating set that our heuristic algorithm can generate so far, which has been shown by simulations in our paper [2].

In our scheme, each host v computes its MPR set $M(v)$ using the following Greedy Algorithm. It selects hosts out of its 1-hop neighbors into the set based on its 2-hop neighborhood information. If the 1-hop neighbor u is selected by v into v's MPR set, v is called the selector of u. Let $N(v)$ be the set of host v and its neighbors. Let $N_1(v) = N(v) - v$ denote the hosts one hop away from host v and $N_2(v) = N(N(v)) - N(v)$ denote the hosts two hops away from v. v *covers* u if u is a neighbor of v.

2.2.1 Greedy algorithm to compute $M(v)$. Host u is a *free neighbor* of v if v is not the largest degree neighbor of u.

- Step 1: add all free neighbors to $M(v)$.
- Step 2: add $u \in N_1(v)$ to $M(v)$, if there is an uncovered node in $N_2(v)$ covered only by u.
- Step 3: add $u \in N_1(v)$ to $M(v)$, if u covers the largest number of uncovered hosts in $N_2(v)$ that have not been covered by the current $M(v)$. Use node ID to break the tie when two hosts cover the same number of uncovered hosts. This step will repeat until there are no more uncovered hosts in $N_2(v)$.

After each host has calculated its MPR set, the dominating set of the whole network is the union of these MPR sets. To reduce the dominating set size, two restrictions are added to let a host decide whether it should be in the dominating set or not.

- Rule 1: the host has the largest degree than all its neighbors' and it has two neighbors that are not connected to each other.
- Rule 2: or the host is a host selected by its neighbor with the largest degree.

Now the network dominating set is composed of hosts which can satisfy either of the two rules. To implement, each host can decide independently whether it should be in the

dominating set by checking the two Rules. Rule 1 is self-explanatory. In Rule 2, the Greedy Algorithm is invoked to let the host know if it is chosen by its neighbor with the largest degree. The explanation and correctness of our algorithm are shown in the next section.

Now we use the network in figure 1 as an example to explain our method. Each host checks Rule 1 and Rule 2 to decide whether it should be in the network dominating set or not. By checking Rule 1, only *s* satisfies the condition and elects itself into the network dominating set. Next each host except *s* checks Rule 2. In Rule 2, each host has to know if it is selected by its neighbor with the largest degree. To know that, each host should run the Greedy Algorithm to compute its MPR set, and send a message containing its id and degree to all those selected. When a host receives such a message from some host, it can check whether this is from the largest degree host among its neighbors or not and make a decision. Take host *s* for an example, it runs the Greedy Algorithm to compute its MPR set. According to the Greedy Algorithm, it does not have any free neighbors, so no free neighbors will be added into its MPR set. Host *s*'s 2-hop neighbor *m* is only covered by its 1-hop neighbor *u*, according to Step 2 in the Greedy Algorithm, *u* should be selected into the MPR set. Another *s*'s 2-hop neighbor *n* is also covered by *u*. Of all the remaining 2-hop neighbors, apply Step 3, 1-hop neighbor *v* covers most of them, namely, *x*, *z* and *k*, while 1-hop neighbor *w* only covers *z* and *k*. So *v* is selected by *s* into its MPR set. Now all the 2-hop neighbors *m*, *n*, *x*, *z* and *k* are covered. The Greedy Algorithm terminates. Then *s* sends a message to hosts *u* and *v*, respectively to let them know that they are selected. When *u* receives the message from *s*, it finds out that this is from the largest degree host among its neighbors. So *u* elects itself into the network dominating set. When *v* receives the message from *s*, it does the same thing and elects itself into the network dominating set too. After each host except *s* is done with Rule 2, the final network dominating set contains hosts *s*, *u* and *v*.

2.3 Correctness of our approach

In this section, we are going to show that our approach can satisfy the time requirement (1-hop constraint) and uses the minimal number of data servers in the network so far.

In our approach, we store the replicas in the hosts in the network dominating set. Since all the hosts in the network are either in the dominating set or neighbors of the hosts in the dominating set. If any host wants some data, it can get it either locally or in one hop.

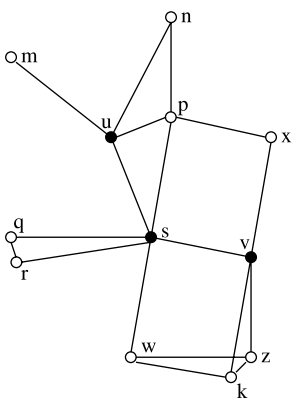

Figure 1. An example of data server selection in an ad hoc wireless network.

Therefore storing the replicas in the hosts in the dominating set satisfies the 1-hop time constraint. The simulations in Ref. [2] have shown that our heuristic algorithm can generate the minimal dominating set so far. That the set generated by our approach is a *connected* dominating set has been proved in Ref. [2]. Our proof, which is derived from the results in papers [1] and [11], is explained as follows.

Actually if Step 1 (add all free neighbors) in the Greedy Algorithm and the second condition (has two neighbors that are not connected to each other) in Rule 1 are removed, the generated set is already a connected dominating set of the network. We will prove that first. Next we will explain how Step 1 and the second condition can help further reduce the network dominating set size.

Let us call D the set of all hosts that have decided to be in the connected dominating set. The largest degree host of the network is clearly in D by Rule 1. Let C be the connected component of the largest degree host in the subgraph induced by D. We are going to show that C is a dominating set for the network (assuming the network is connected of course). This will prove in particular that any host in D has a neighbor in C, implying that C indeed equals D. This will thus prove that D is connected on the one hand and that D is a dominating set on the other hand.

Assume by contradiction that C is not a dominating set. Let $\mathcal{N}(C)$ denote the set of all hosts that are in a given set C or have a neighbor in C. There must exist some hosts that are not in $\mathcal{N}(C)$. Consider the set V of hosts connecting some host in C to some host in $\overline{\mathcal{N}(C)}$, the complementary of $\mathcal{N}(C)$. V is the set of hosts which have at least one neighbor in C and at least one neighbor in $\overline{\mathcal{N}(C)}$. As the network is connected, our assumption implies that V is not empty. Notice that $V \cap C = \emptyset$ by construction. We now consider the largest degree host m in $\mathcal{N}(V)$.

- Either m is in $\overline{\mathcal{N}(C)}$. As $m \in \mathcal{N}(V)$, there exists a neighbor v of m in V. Let c be a neighbor of v in C. Consider all the 1-hop neighbors selected by m: as c is a 2-hop neighbor of m, there must exist some host r selected by m which is a neighbor of c. Notice that r must be in V. As the largest degree neighbor of r is m, r should have elected itself in D by Rule 2, contradicting $V \cap C = \emptyset$.
- Either m is in $\mathcal{N}(C)$ which can be partitioned in V and $\mathcal{N}(C) - V$:
  - If m is in V, m should have elected itself as being in D since the degrees of all its neighbors are smaller. This is again a contradiction with $V \cap C = \emptyset$.
  - On the other hand if m is not in V, it cannot have any neighbor in $\overline{\mathcal{N}(C)}$. Let v be a neighbor of m in V, and let x be a neighbor of v in $\overline{\mathcal{N}(C)}$. As m has no neighbor in $\overline{\mathcal{N}(C)}$, x is not a neighbor of m and some host r selected by m must be connected to x. As m cannot have any neighbor in $\overline{\mathcal{N}(C)}$, r must be in $\mathcal{N}(C)$. As r has a neighbor in $\overline{\mathcal{N}(C)}$, it is in V. The largest degree neighbor of r is thus m implying that r should be in D by Rule 2, contradicting again $V \cap C = \emptyset$.

In all cases we get a contradiction. C thus have to be a dominating set. As mentioned before, this implies that $D = C$ is a connected dominating set.

Next, we explain why in the Greedy Algorithm, Step 1 is added and put before Step 2 and Step 3. When computing a MPR set for a host v, the free neighbors should be considered first because they will be eliminated from the dominating set of the network by Rule 2 since v is

not their largest degree neighbor. They come for "free" for host v. That means, the inclusion of them will not increase the network dominating set size.

Take the network in figure 2 as an example. Suppose Step 1 is removed from the Greedy Algorithm, in the figure, when host v calculates its MPR set, it can select its 1-hop neighbor x to cover w. Since x is selected by its neighbor with the largest degree, according to Rule 2, x will elect itself into the network dominating set. If Step 1 is added into the Greedy Algorithm, then v will select y instead of x into its MPR set because y is a free node of v and can cover w. But y will not elect itself into the network dominating set because it does not satisfy Rule 2. So with Step 1 added, we can prevent hosts from entering the network dominating set as much as possible.

Next, we show why in Rule 1, the condition that a host should have two neighbors that are not connected to each other is added. This is because if all the hosts that do not satisfy this condition are excluded from the dominating set, the resulting hosts can still form a connected dominating set.

We assume the network graph is not a complete graph. Suppose all of a host $v's$ neighbors are pair-wise connected, then there must exist a host that is not a neighbor of v. Let w be such a host with the largest degree. Since v has the largest degree in its 1-hop neighborhood, either v or w has the largest degree in the 1-hop neighborhood of any neighbor of v. When one neighbor of v, say u, is selected by its largest degree neighbor $v(w)$, u should elect itself into the network dominating set. Since u covers v and all neighbors of v, v can be removed and the rest of the hosts in the network dominating set can still form a connected dominating set.

2.4 Simulation results

In this section, we use simulation results to display the number of data servers needed in a dense and a sparse ad hoc wireless networks. It has already been shown by simulations in Ref. [2] that our method can generate the minimum number of data servers so far. So here we just present the results of our approach.

In the simulations, random hosts are distributed in a 100×100 space. Transmission ranges of 6 (sparse) and 40 (dense) are considered. The number of hosts ranges from 20 to 100. Each set of data has been run 100,000 times and an average number of data servers in each network topology is calculated (tables 1 and 2).

From the two tables, we can see that in a dense network, the number of data servers is smaller while in a sparse network, the number of data servers is larger. This is because in a dense network, there are more connections among the hosts and a host is more likely to be

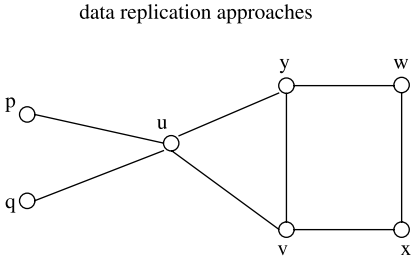

Figure 2. An example to explain why Step 1 in the Greedy algorithm is added.

Table 1. Number of data servers in a dense network.

Number of hosts	20	40	60	80	100
Number of data servers	5	6	7	7	7

Table 2. Number of data servers in a sparse network.

Number of hosts	20	40	60	80	100
Number of data servers	7	15	22	29	37

covered by other hosts. We can also see that in a dense network, after a certain point, the number of data servers will not change anymore. This is because all the hosts are covered.

3. Three schemes for $k \geq 1$

3.1 Preliminaries

In this section, we discuss when $k \geq 1$. We propose three data replication schemes so that each host can obtain data in at most k hops with minimal number of data servers. These three approaches are: a centralized approach (CEN), a distributed approach with no status information (DNSI) and a distributed approach with status information (DWSI).

Before we introduce our three approaches, first we present a scheme that will be used by all our approaches. It is called *controlled flooding*. Unlike the regular flooding where no limit is set, in the controlled flooding, the source sets a limit L and floods the message to neighbors at most L hops away. The detail is as follows.

3.1.1 Controlled flooding algorithm

- The source sets a limit of L and broadcasts a message to all its 1-hop neighbors.
- If a host in the network receives this kind of message for the first time, it decrements the limit by one and if the limit is not zero, it will relay the message to all its 1-hop neighbors. If a host has received the message before, it simply ignores it.

Next, we introduce our first approach: the centralized one.

3.2 A centralized approach (CEN)

In the CEN, the following algorithm is run by a coordinator to identify the data servers. A coordinator can be any host. A CEN is not suitable for an ad hoc environment where hosts move frequently. However, it can be used as a benchmark to test how effective the distributed ones are. In the following, if a host i can get the data items it needs from a host j in k hops, we say that host j *covers* host i in k hops. Given a network, finding the minimum number of hosts to cover all the hosts in k hops is NP hard. Thus, we provide a heuristic greedy algorithm as follows:

(i) The coordinator counts the number of hosts each host can cover in k hops using controlled flooding. The limit L of the controlled flooding is set to k.

(ii) For any host i in the network, if it is only covered by a host j in k hops, host j will become a data server. After a host becomes a data server, itself and all the hosts covered by it in k hops will not be considered again in the algorithm.
(iii) The rest of the hosts are ordered in non-decreasing order by the number of hosts they can cover.
(iv) The host that can cover the most hosts becomes a data server.
(v) If the number of remaining hosts is not 0, go back to Step (i); otherwise, the algorithm terminates.

When the greedy algorithm terminates, all the data servers are identified to cover all the hosts in at most k hops.

3.3 A distributed approach with no status information (DNSI)

The CEN discussed above is not suited to a changing network topology. In this section, we are going to propose a distributed one. In this approach, the hosts store very few information about each other: each regular host only stores the information (id) of the first data server that can reach it while each data server does not store any information about the regular hosts covered by it in k hops. Therefore this approach is called a DNSI. The approach has two phases: an initialization phase and an updating phase. In the initialization phase, the data servers are identified. In the updating phase, the data servers may change due to topology change.

1) Initialization phase: in this strategy, each host i will decide whether it should store data items locally by generating a random number h_i ($0 \leq h_i \leq 100$). If h_i is less than threshold $d(0 \leq d \leq 100)$, then host i will be a data server; otherwise, it will still remain as a regular host. If $|H|$ is the total number of hosts in the network, then $d^*|H|$ hosts will become data servers. The advantage of this local decision is that it avoids the initial communication overhead. All these identified data servers will first get the data from the source, which may take longer than k hops.

Next each data server sets limit to k and uses controlled flooding to broadcast the message that it has become a data server to its neighbors within k hops. The message contains data server's id i. When a regular host receives the message, it knows it can get the data from server i in at most k hops. If a regular host receives more than one such messages, it accepts the first one and ignores the rest. When another data server receives the message, it simply ignores it.

If a regular host does not receive such a message after a predefined time constraint, it should become a data server itself. And then similarly it will broadcast the message along with its own id to its neighbors within k hops using controlled flooding with the limit set to k. This process is repeated until each host is covered by a data server within k hops.

2) Updating phase: in an ad hoc network, hosts can move around, switch on and off at will and possibly cease to function. All of these can make the network topology dynamic. Therefore the data servers need to be updated if necessary. If a host needs data but cannot get it in k hops, it should first get the data from some data server (more than k hops away) and then mark itself as a data server. Next as described above, it should broadcast this news to all its neighbors within k hops. This process is repeated until the network reaches a stable state again: each host is covered by a data server within k hops.

3.4 A distributed approach with status information (DWSI)

In DNSI, the data servers store no information about regular hosts. Memory space in each server is saved in this way but it can also result in redundant data servers: for example, if a data server covers hosts in the k-hop neighborhood that are also covered by other hosts, this data server is redundant and it can be put back to a regular host. In order to identify these redundant data servers, the data servers and regular hosts should store information about each other. Therefore in the following, we put forward a DWSI to reduce the number of data servers generated. It also has two phases: an initialization phase and an updating phase.

1) Initialization phase: as in the DNSI approach, each host will decide whether it should be a data server locally by generating a random number h_i, ($0 \leq h_i \leq 100$). If h_i is less than threshold d($0 \leq d \leq 100$), then host i will be a data server.

Next similar to the DNSI approach, each data server i sets the limit to k and uses controlled flooding to broadcast the message to its neighbors within k hops. The message contains the data server's id i and a count c initialized to 0. The count represents the number of hops from the data server to the host it reaches. It will be incremented by one each time the message is sent to the next host. When a regular host receives the message, it knows it can get the data from server i in c hops ($c \leq k$). The difference in this approach is that if a regular host receives more than one such messages, it will store the ids of all these data servers and the hops from them in its memory. Whenever it needs data, it will get it from the closest data server. The host reached will also send back a reply to server i along with its id j. When server i receives the reply, it will store the host's id j in its memory. Thus data server i knows which hosts are covered by it and these hosts are called the *members* of data server i. If another data server receives the message from server i, it records the id and hops from i in its memory and sends back a reply with its own id to the sender so that data servers within k-hop distance can know each other.

If a regular host does not receive such a message from a data server after a predefined time constraint, it should become a data server itself. It will follow the above procedure to broadcast the message within its k-hop neighborhood and the related information will be recorded. This process is repeated until each host is covered by a data server within k hops.

To get rid of redundant data servers, data servers can exchange their members' information after a certain period of time. If the members of a data server are covered by other hosts, it can be put back to a regular host.

2) Updating phase: when the network topology changes, if a host needs data but cannot get it in k hops, it should first get it from some data server (more than k hops away) and then mark itself as a data server. Next as described above, it should broadcast this news to its neighbors within k hops with related information recorded. This process is repeated until the network reaches a stable state again: each host is covered by at least one data server within k hops.

3.5 Simulations

In this section, we conduct simulations to compare these three methods: CEN, DNSI and DWSI. Random hosts are generated in a 100×100 space. We input the number of hosts, transmission range r of each host, the value of k, and the threshold d as parameters. The host numbers range from 60 to 160. The threshold is set to 10, then 5, and then 2. We consider

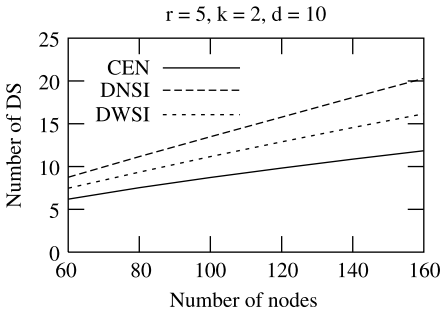

Figure 3. Number of data servers with transmission range 5, access delay 2 hops, and threshold 10.

transmission ranges of 5, 30 and 60 to represent sparse and dense networks. We pick the values of k to be 2 and 3. Each sample runs 10,000 times.

The graphs for simulation results are shown here (figures 3–8). In the figure labels, DS is used instead of Data Servers for space's sake. In figures 6–8, the results of DWSI are so close to CEN that its line is overlapped with CEN's. In all the cases, we can see that the CEN generates the least number of data servers. DWSI generates fewer data servers than DNSI. This means if the data servers and the regular hosts can store some information about each other, the number of data servers can be reduced.

Also, the denser the network, for example, when r is 30 and 60, the closer the DWSI is to the CEN. And the number of data servers is reduced. That means, there are more connections and one data server can cover more regular hosts.

Comparing DNSI with DWSI and CEN, the number of data servers generated by DNSI is large when $d = 10$. This is because we set the threshold too high. If we lower the threshold d to 5 and then to 2, the number of data servers generated by DNSI each time will be greatly reduced and is close to CEN and DWSI (figures 6–8). But using a smaller threshold will incur more delay to generate all the data servers to cover the whole network. This is because when the threshold is small, in the initialization phase, fewer hosts will become data servers. It will take some hosts some time (at least after the predefined time expires) to realize that they should be data servers. The more hosts to realize this, the more the delay. In figure 9, when the threshold is 10, very few hosts will become data servers later. When the threshold is 5, there are a few. And when the threshold is 2, the number of hosts that later become data servers increases a lot. That means when we reduce the threshold, we should also consider the delay to generate the data servers.

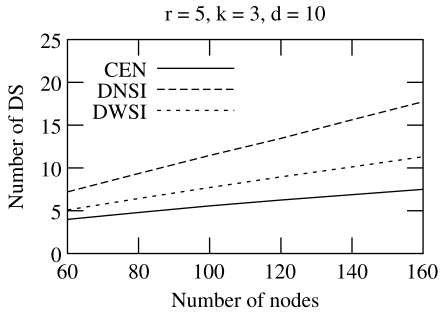

Figure 4. Number of data servers with transmission range 5, access delay 3 hops, and threshold 10 within k hops.

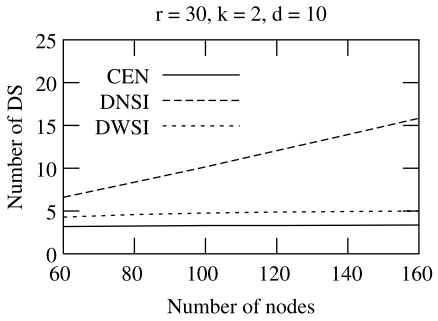

Figure 5. Number of data servers with transmission range 30, access delay 2, and threshold 10.

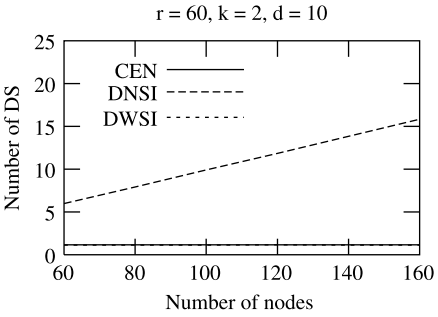

Figure 6. Number of data servers with transmission range 60, access delay 2, and threshold 10.

4. Conclusion

In this paper, we have proposed replica allocation schemes to let each host obtain data items in at most $k (k \geq 1)$ hops using minimal number of data servers in the network. In our schemes, no access frequencies by mobile hosts are assumed. Since when $k = 1$, the problem is equivalent to finding the smallest size dominating set of the network, we have put forward a special scheme. In our scheme, each host decides whether it should become a data server based on 2-hop neighborhood information. Our simulations have shown that our scheme can generate the minimal number of data servers in the network so far. When $k \geq 1$, three replica allocation schemes have been proposed, a centralized scheme (CEN) (used as a benchmark to compare

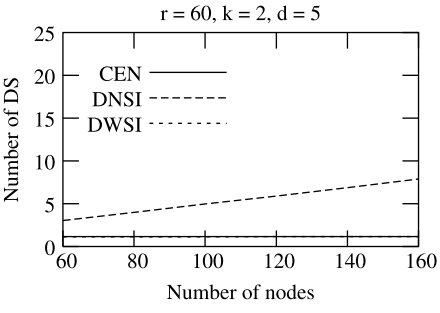

Figure 7. Number of data servers with transmission range 60, access delay 2, and threshold 5.

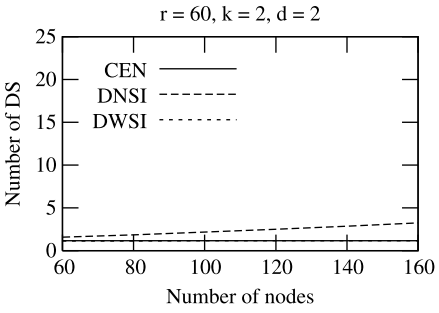

Figure 8. Number of data servers with transmission range 60, access delay 2, and threshold 2.

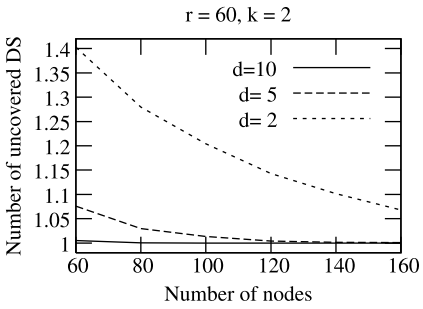

Figure 9. Number of hosts that later become data servers (uncovered DS) with different thresholds.

with the distributed ones), a distributed scheme with no status information stored at the data servers (DNSI), and a distributed scheme with status information stored at the data servers (DWSI). Simulation results have shown that CEN generates the least number of data servers (because it is a centralized algorithm). DWSI is close to CEN, particularly in dense networks. DNSI can be close to CEN and DWSI when the threshold is set not too high with the proper delay considered. This means that both DNSI and DWSI are practical algorithms to find data servers in ad hoc wireless networks to satisfy the *k*-hop time constraints. In our future work, we will continue our efforts to find more and better heuristic distributed algorithms.

Acknowledgements

The author wants to thank all the reviewers of their valuable comments, and Prof. Jian Shen in the Mathematics Department at Texas State University for his excellent suggestions on the paper.

References

[1] Adjih, C., Jacquet, P. and Viennot, L., 2002, Computing connected dominated sets with multipoint relays, Technical Report, INRI, October.
[2] Chen, X. and Shen, J., 2005, Improved schemes for power-efficient broadcast in ad hoc networks, *International Journal of High Performance Computing and Networking*, **4**(2).
[3] Hara, T., 2001, Efficient replica allocation in ad hoc networks for improving data accessibility, Proceedings of IEEE Infocom 2001 , pp. 1568–1576.

[4] Hara, T., 2003, Replica allocation in ad hoc networks with periodic data update, *Mobile Networks and Applications*, Springer Netherlands **8**(4), 343–354.
[5] Hara, T. and Madria, S.K., 2004, Dynamic data replication schemes for mobile ad-hoc network based on aperiodic updates. *Proceedings of International Conference on Database Systems for Advanced Applications*, pp. 869–881.
[6] Hara, T., 2006, Data replication for improving data accessibility in ad hoc networks, *IEEE Transaction on Mobile Computing*, **5**(11), 1515–1532.
[7] Pei, G., Gerla, M., Hong, X. and Chiang, C.-C., 1999, A wireless hierarchical routing protocol with group mobility, Proceedings of the IEEE Wireless Communications and Networking Conference.
[8] Pei, G., Gerla, M. and Hong, X., 2000, LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility. *Proceedings of the 1st ACM Annual Workshop on Mobile Ad Hoc Networking and Computing*.
[9] Perkins, C., 2001, *Ad hoc Networks* (Addison-Wesley).
[10] Wang, K.H. and Li, B., 2002, Group mobility and partition prediction on wireless ad-hoc networks. *Proceedings of IEEE ICC Conference*, pp. 1017–1021.
[11] Wu, J., 2003, An enhanced approach to determine a small forward node set based on multipoint relays. *Proceedings of 2003 IEEE Semiannual Vehicular Technology Conference, October*.
[12] Wu, B., Chen, J., Wu, J. and Cardei, M., 2006, A survey on attacks and countermeasures in mobile ad hoc networks. In: Y. Xiao, X. Shen and D.-Z. Du (Eds.) *Wireless/Mobile Network Security* (Springer).