Computer Networks 57 (2013) 3654–3672

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

SCALAR: Scalable data lookup and replication protocol for mobile ad hoc networks

Emre Atsan [a,b], Öznur Özkasap [b,*]

[a] School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland
[b] Department of Computer Engineering, Koc University, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 6 April 2012
Received in revised form 8 August 2013
Accepted 12 August 2013
Available online 28 August 2013

Keywords:
Data replication
Data lookup
Scalability
Virtual backbone
Connected dominating set
Mobile ad hoc networks

ABSTRACT

Data replication, as an essential service for MANETs, is used to increase data availability by creating local or nearly located copies of frequently used items, reduce communication overhead, achieve fault-tolerance and load balancing. Data replication protocols proposed for MANETs are often prone to scalability problems due to their definitions or underlying routing protocols they are based on. In particular, they exhibit poor performance when the network size is scaled up. However, scalability is an important criterion for several MANET applications. We propose a scalable and reactive data replication approach, named SCALAR, combined with a low-cost data lookup protocol. SCALAR is a virtual backbone based solution, in which the network nodes construct a connected dominating set based on network topology graph. To the best of our knowledge, SCALAR is the first work applying virtual backbone structure to operate a data lookup and replication process in MANETs. Theoretical message-complexity analysis of the proposed protocols is given. Extensive simulations are performed to analyze and compare the behavior of SCALAR, and it is shown to outperform the other solutions in terms of data accessibility, message overhead and query deepness. It is also demonstrated as an efficient solution for high-density, high-load, large-scale mobile ad hoc networks.

1. Introduction

A mobile ad hoc network (MANET) is a self-organizing, infrastructureless, dynamic wireless network of autonomous mobile nodes. The network is ad hoc since there is no fixed and known network structure that every other node forwards data. Each node is not only an end system, but also a router in the network's multi-hop communication structure. MANETs are adaptive networks, which are reconstructed in the case of network changes due to mobility. Possible application areas are military communication systems, personal area networks, wireless peer-to-peer networks, and vehicular ad hoc networks.

In MANET research, most of the effort has focused on the design of routing protocols that aim to find multi-hop paths between two nodes. Besides routing, accessing to a remote data is equally important since the goal of an ad hoc network structure may be to provide the necessary data items to requester nodes. Different than static, infrastructure-based conventional networks, locating and accessing the remote data (*data lookup*) is a challenging problem in ad hoc environments. In this case, mobile users need to learn the availability of data items in an ad hoc manner without the help of any central server. Moreover, due to the unpredictable mobility behaviors of nodes, a rapid change in the MANET topology is presumable. This change can result in partitioning, dividing the network into isolated sub-networks. Thus, data availability in MANETs is lower than static networks.

One possible solution to this problem can be *replication* of popular data items in the network. In conventional distributed systems, replication not only increases availabil-

* Corresponding author. Tel.: +90 212 338 1584; fax: +90 212 338 1548.
E-mail address: oozkasap@ku.edu.tr (Ö. Özkasap).

1389-1286/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2013.08.012

ity, but also helps for load balancing and fault-tolerance. Also, in geographically widely dispersed systems, having a copy at a nearby location can solve most of the communication latency related problems. On the other hand, it should be taken into consideration that it may not be possible to replicate every data item at each network node due to limited resources or power considerations of the nodes. Therefore, it is important to define good replication criteria and rules that can select the most appropriate data items and best hosts for replication. Another possible solution to the problem of low data availability, or accessibility in mobile networks is *cooperative caching* techniques [1–4]. Cooperative caching for MANETs is the coordination of several nodes to share a cached data in an efficient way for all. Caching-based systems let nodes to cache data or path to a data item in order to increase data availability and achieve lower query delays. Caching reduces communication cost of the system, which consequently results in the reduction of bandwidth and energy usage. In general, caching approaches aim to achieve that neighboring nodes cache different data items. This will increase the data diversity in a region and as a result, more data will be accessible in the network [1]. Furthermore, caching and replication schemes help to establish load balancing among nodes.

Accessing a specific data item in MANETs is an important problem and solutions to this problem can give better results when they are supported with a good data replication or caching approach. On the other hand, possible solutions to this problem can be even more complex if the size of the network increases; which means a possible solution for a meso-scale network can be inefficient for large scale scenarios. Existing protocols for data lookup and replication in MANETs do not consider scalability. In [5], three replica allocation schemes are proposed in which the main parameter for replication decision is the *access frequencies* of data items. However, that study does not consider large-scale networks. Other replication schemes, such as [6–9] also do not address the scalability problem by definition. These solutions can give unexpected results as the system size scales up, and need to utilize additional mechanisms for improvement. We observe that, most of the time, this is because of the dependency to underlying routing protocols which are shown to have scalability problems [10]. In some cases, this performance loss in large scale networks is due to proposed solution's large control message overhead. In small or meso-scale networks the message overhead may not be detrimental, but it affects the performance of the solution in large-scale scenarios.

In this study, we propose a SCAlable data Lookup And Replication framework, called *SCALAR*, which does not depend on the existence of an underlying MANET routing protocol. Our motivation is that a scalable and efficient solution for data lookup and replication in MANETS is essential for several applications. SCALAR consists of three parts: virtual backbone construction, data lookup and reactive replication. It dynamically constructs a virtual backbone structure among the mobile nodes, which is based on an approximation of minimum connected dominating set (CDS) problem in graph theory. CDS-based approach is chosen to minimize the number of nodes in the network involved in data lookup and replication process. Then, our data lookup protocol takes advantage of the dominating set behavior of constructed virtual backbone for low-cost data requests. Lastly, we extend the data lookup protocol with a novel data replication approach, which aims to replicate frequently accessed data items from far away places.

Contributions SCALAR is proposed as a fully distributed algorithm, which operates in a peer-to-peer (P2P) fashion. Different from other data lookup and replication approaches existing in the literature, SCALAR is a complete solution, in which disseminating request packets does not require an underlying routing protocol. Furthermore, SCALAR specifically aims at operating efficiently in large scale ad hoc networks different from other data replication or caching solutions developed for MANETs. To the best of our knowledge, using a virtual backbone structure in order to operate a data lookup and replication process has not been investigated in the literature before. Performance results show that our proposed system outperforms the straightforward solutions for data lookup in MANETs when the scale of the network is large. It is demonstrated that for increasing number of nodes in the network, message complexity of our solution has an acceptable bound. Besides, simulation results present that SCALAR is an efficient solution for high density networks, as well as large scale ones.

The rest of this paper is organized as follows: We review the related work in Section 2. Section 3 presents our SCALAR protocol. Experimental preliminaries are given in Section 4. Performance results and analysis of SCALAR are discussed in Section 5. Comparative results for application scenarios are presented in Section 6. Finally, Section 7 concludes and gives future directions.

2. Related work

In this section, we first review prior studies on data replication and cooperative caching techniques for MANETs, and then demonstrate a feature comparison of these techniques. We also give an overview of controlled-flooding techniques for MANETs.

2.1. Data replication techniques

Data replication is used to avoid data losses in case of unpredictable disconnections of mobile nodes by increasing system wide data availability [11]. Also, replication improves the efficiency by decreasing the number of hops that a data item is transmitted from source to destination. Recent surveys on MANET replication techniques classify protocols according to issues of energy consumption, scalability, and network partitioning [12,13].

One of the well-known schemes for data replication in MANETs is proposed by Hara and Madria in [5] in which the main parameter for replication decision is the *access frequencies* of data items, and each method proposed for replica allocation has been studied for both update-less and randomly updated systems. The following three replica allocation methods are offered for network environments without any update mechanism for data items:

SAF (Static Access Frequency) method: In SAF, each mobile host gives the replication decision based on only its own access frequencies to the data items. This means, each host replicates the data items for its own personal needs, without considering any of the data replications in neighboring nodes and their access frequencies. SAF creates low message traffic and low processing overhead. On the other side, this method gives low data accessibility because of the fact that many mobile hosts probably replicate the same data items if they have similar access characteristics.

DAFN (Dynamic Access Frequency and Neighborhood) method: In a mobile environment, where mobile nodes frequently access the same small set of data items, if SAF is used for replication, then neighboring nodes possibly replicate the same data items. This duplicate replication between connected nodes is most of the time unnecessary and is a waste of memory space. In order to eliminate this replica duplication, DAFN method suggests that if there is duplication of replicas between neighboring nodes, then the neighbor with the lowest access frequency changes its replica to another data item. DAFN increases the data availability by increasing the number of distinct replicas in the network.

DCG (Dynamic Connectivity based Grouping) method: Different than DAFN method, DCG method shares replicas among larger and stable groups of nodes. For this purpose, it creates biconnected components of a network where each component is considered as a group. By definition, biconnected groups are not partitioned even if one mobile host disappears from the network (or group). This provides high stability to the group. DCG offers better data availability. On the other side, both processing overhead and message traffic are higher than SAF and DAFN. This is due to the exchange of more information in each allocation period by mobile nodes.

Another data replication method for MANETs is DREAM [8], which focuses on the balancing of energy consumption among the mobile nodes and aims to increase the data availability in the network. Basically, DREAM gives high priorities to the data items that are accessed frequently by the transactions that have time limitations or deadlines. It is similar to [5] in means of replication decision based on access frequency of data items. However, this technique also extends the replication decision with new decision parameters: link stability, transaction types, data types and remaining power.

CLEAR (Context and Location-based Efficient Allocation of Replicas) is dynamic replica allocation scheme that aims at enhancing data availability in mobile ad hoc peer-to-peer networks [7]. It constructs a super-peer structure for replica allocation and utilizes mobility patterns. CLEAR considers node loads, data sizes, consistencies of replicas and user schedules during replication decision, and it assumes that the cost of maintaining a desired level of consistency for the replicas of a data item can be estimated from the percentage of change in the value of an attribute of the data item. Data size and access frequency of an item play role in the replication decision. Load, access frequencies, memory and energy constraints of a mobile host are used for the selection of replica node. This technique calculates the cost-effectiveness of the replication for a data item in a mobile host to give the final replication decision. However, CLEAR does not address scalability as super-peer election can be a problem in large-scale systems and their performance results are limited to 50 nodes in a region. Another study offers a classification of consistency levels for data replication based on application requirements, and protocols to achieve the consistency levels [9].

CADRE (Collaborative Allocation and Deallocation of Replicas with Efficiency) aims at achieving fair replica allocation among mobile nodes and makes the deallocation decisions collaborative [6]. Unlike CLEAR, CADRE addresses replica deallocation, fairness in replication and avoidance from thrashing. By making allocation and deallocation decisions jointly among nodes in the network, CADRE can protect the network from multiple reallocations of the same data item, which can lead to a thrashing condition that mobile nodes spend more resources on re-allocation of data items than replying other nodes' requests. CADRE employs super-peer architecture with gateway nodes as peers with high processing capacity, bandwidth and energy that they aim at improving efficiency of search and replication. However, like CLEAR, CADRE does not address scalability and their performance results are limited to 50 nodes in a region.

An integrated data lookup and replication approach for distributed data accessability for a cooperative group of users, sharing information among themselves in MANET, is proposed in [14]. The approach is based on a cross layer design supported by predictive location-based QoS routing. It assumes that node movement behavior is predictable and with the help of location information and velocity of a node taken from the routing layer, the system can predict the upcoming partitioning and makes replication decisions using these information. Their performance results indicate that using cross-layer approach, the routing protocol supports data advertising, lookup and replication that results in high data access success rate in dynamic ad hoc networks. However, the performance results are reported up to 40 nodes and do not address scalability.

DHTR (Distributed Hash Table Replication) is an optimistic replication method proposed for efficient consistency maintenance in large scale MANETs [15]. It is based on clustering of nodes into hierarchical groups and it uses a distributed hash table for efficient data lookups. DHTR exploits the communication simplicity of a cluster-based hierarchical network structure and claims to decrease the communication overhead of the network. Communication complexity is an important scalability constraint for large scale ad hoc networks. DHTR creates a network architecture that consists of non-overlapping cluster groups that each of these groups is managed by a cluster head. However, it does not provide a replica management protocol or any details of which data to replicate where. Generally, DHTR targets a scalable query and update propagation technique that can be used in a cluster-based replication algorithm. It also depends on an underlying routing protocol and AODV routing protocol is used for this purpose. However, it could cause scalability problems in large networks, and their performance results focusing on query/update propagation are reported for simulations including up to 250 nodes.

A recent study addresses the joint optimization problem for content replication mobile networks, and considers both the number of replicas and finding their most suitable location on the network [16]. Their distributed lightweight replication algorithm and preliminary results for replicating single data item are shown to approximate well the accuracy of centralized solutions. A user-centric replication model named CReaM has been recently proposed in [17]. It allows users to control the amount of resources that they would share for replication, and then these resources are used to improve data availability. Based on user needs, the data availability is aimed to be increased by replication. CReaM is designed for highly mobile networks subject to frequent topology changes. The nodes have autonomic behavior that triggers replication requests based on user settings and resource information.

2.2. Cooperative caching techniques

Data caching techniques namely, CachePath, CacheData and HybridCache, that can be embedded as a middleware support between routing and application layer protocols are proposed in [1]. In CachePath, transporter nodes cache the path to the nearest cache of a data item. A node need not cache every path information of all passing data. Instead it only saves the data path when it is closer to the caching node than the data source. In CacheData, transporter nodes cache the data instead of the path. A node decides to cache the passing-by data if it is frequently accessed by neighboring nodes. HybridCache is a combination of CachePath and CacheData. HybridCache decides when to use which scheme based on a criteria (data size and time-to-live value). Basically, it caches the data item if it is small sized enough, or caches the path otherwise. When the cache is full, deciding what to cache is an important part of the solution in caching techniques. In these techniques, mobile nodes consider two factors in selecting the cached item to be replaced: distance and access frequency.

Similar to [1], COOP caching [4] sits on the middleware level between application and routing layer protocols. Requested items are found by using the so-called cocktail scheme cache resolution strategy. In cache resolution, a node that receives a data request checks local caches and recent requests for requested item. If not found, it starts an adaptive flooding, which limits the number of hops that a broadcast packet can travel. Adaptive flooding aims to find a cache of the requested item in the neighborhood. If it cannot find, it directly sends the request to the server (or source of the requested item) using the underlying routing protocol. While the request is carried to the server, if an intermediate node finds the requested item in its cache, it stops forwarding and returns the data to the requester. COOP also studies how to decide which data item to keep in limited cache, and similar to [1], it uses TTL-based consistency control mechanism for cached data items.

In benefit-based data caching [3], the optimization problem of minimizing total access cost of an ad hoc network with limited memory space and multiple data items is considered. It is stated that this problem is known to be NP-hard, so they propose a polynomial-time approximation algorithm, which achieves a reduction in the total access cost at least one-fourth of the optimal solution. They also give a distributed version of their approximation algorithm, which performs close to the central one. Proposed approximation algorithm is a greedy approach, which caches data items in the memory maximizing the reduction in total access cost in a greedy manner.

A recent study proposes a cooperative caching approach based on P2P data exchange that aims to create content diversity among the nodes' neighborhood for providing efficient data access [18]. The cross-layer probabilistic estimate for locating the cached data in a node's proximity is the novel feature of the approach. Another recent work proposes adaptive caching methods that consider resource constraints such as storage space, battery life, and bandwidth in MANETs [19]. The cache replication method aims to balance the broadcast overhead among the nodes for reducing both energy consumption and bandwidth utilization. The cache distribution method aims to redistribute cached data items based on their demand in the network for reducing response time and hop count.

2.3. Feature comparison of data replication and caching

Replication aims to form local or close copies of data items in order to reduce communication cost of data access and to improve data availability. On the other hand, caching is the action of saving data items or their paths when data items are received by nodes. The main difference between caching and replication can be stated as the caching occurs after the retrieval of data item, while replication of a data item can occur before a request received for that item. Replication is a decision of larger groups, in the sense that, the nodes decide to make copies based on a global decision, which involves a larger set of nodes. In general, replication is generally based on the access statistics of nodes or other additional network information.

In Table 1, we compare the data replication and caching techniques that we reviewed using common features critical for a replication/caching technique. Compared features are as follows:

- *Cluster-based architecture*: Does the technique require or construct a cluster-based architecture?
- *Consistency mechanism*: Does the technique have a consistency control mechanism with/without update propagation in the network?
- *Limited memory*: Does the technique have a memory constraint for replicated/cached data items?
- *Routing Protocol Dependency*: Does the technique assume the existence of an underlying routing protocol?
- *Fairness*: Does the technique aim at load balancing/fairness among mobile nodes?
- *Scalability*: Is the technique developed for scalable networks?

2.4. Controlled flooding

MANETs assume wireless communication among mobile nodes without any physical infrastructure support. However, this infrastructureless communication causes in-

Table 1
Feature comparison table.

	Hara et al. [5]	DREAM [8]	CADRE [6]	CLEAR [7]	Chen et al. [14]	DHTR [15]	Yin et al. [1]	COOP [4]	Benefit-based [3]
Cluster-based	–	–	+	+	+	+	–	–	–
Consist. mech.	+	+	–	+	–	+	+	+	–
Lim. memory	+	+	+	+	+	–	+	+	+
Routing dep.	–	+	–	–	+	+	+	+	+
Fairness	–	–	+	+	+	–	–	–	–
Scalability	–	–	–	–	–	+	–	–	–

creased communication costs. A common message overhead source in a MANET environment is blind flooding/broadcasting. Flooding is used in the route discovery phases of several routing protocols developed for MANETs [20]. This flooding/broadcasting packets in the network cause the creation of excessive redundant packets (*broadcast storm problem* [21]) and collision/contention problem in wireless channel. In large scale MANETs, problems become more pronounced [10].

Broadcasting data to all nodes in a system is an essential service for several cooperative applications in ad hoc networks. Such a service should be designed to be efficient and reliable. In the simple broadcasting method of flooding, a source transmits a data message to all nodes in its wireless range. On receiving the data for the first time, each node forwards or rebroadcasts it to nodes in its range.

The works of [22,21] study broadcasting and related issues in a MANET. It is shown that if blind flooding does the broadcasting, then problems of redundancy, contention, and collision emerge. These problems with flooding are known collectively as the broadcast storm problem. In order to deal with this problem, two directions were extensively studied. One is to reduce the probability of redundant broadcasts, and the other is to distinguish the timing of rebroadcasts. Based on these, probabilistic, counter-based, distance-based, location-based, and cluster-based schemes were developed for efficient broadcasting in MANETs and analyzed via simulations.

In probabilistic flooding, upon receiving a message for the first time, a node computes a fixed probability and rebroadcasts the message with that probability. It has been shown that plain flooding achieves the highest possible delivery ratio, particularly for sparse networks, when compared to probabilistic flooding. Furthermore, high delivery ratio levels in data broadcasting with probabilistic flooding have been demonstrated to be possible when the rebroadcasting probability is set to a rather high value [23,24]. For example, low density network simulations reported in [23] show that success rate (that is, delivery ratio) varies linearly with the rebroadcasting probability and high success rates are observed with probabilities greater than 0.9.

In order to reduce the number of redundant broadcasts, additional approaches such as counter-based, distance-based, and location-based rebroadcasting use local information at a node to decide whether to rebroadcast a message. For instance, the counter-based algorithm uses a counter to control the number of times a broadcast message is received at a node during the interval before rebroadcasting is performed. When the counter's value reaches a threshold for the message, the rebroadcast is suppressed. This scheme has been demonstrated to eliminate many redundant broadcasts in dense networks [21]. However, a drawback of such schemes is that they increase message transmission delays.

To maximize the utilization of incapacitated node resources and to minimize drawbacks caused by flooding, virtual backbone (or spine) structures inspired by physical internet backbones were proposed. Virtual backbones are used in topology management and routing protocol design in MANETs. They help to avoid the excessive use of broadcast flooding in large scale ad hoc networks. There are several virtual backbone construction approaches available in the literature [25–27]. Besides, in [28] it is shown that routing protocols adapted over virtual backbones perform better than their original definitions. It is because of the fact that the backbone enables routing protocols to use only a subset of nodes in the network for route management. By this way, it helps to avoid the excessive use of broadcast flooding in large scale ad hoc networks.

Another cluster/virtual backbone based scalable MANET routing approach named Cluster Overlay Broadcast (COB) was proposed in [29], and its further performance analysis is described in [30]. This work employs Least Cluster Change (LCC) algorithm [27] to establish and maintain a clustered structure of the network, and it has similarities to our data lookup process. However, it does not extend the routing process with a scalable data replication mechanism. Besides, cluster creation algorithms employed by these works are also different. Authors analyze the route discovery complexity of their scheme and conclude that it is polynomially proportional to the minimum number of hops between the source node and the destination node.

3. SCALAR

In this section, we first describe the system model, and then give details of SCALAR composed of:

- a virtual backbone construction algorithm,
- a scalable data lookup protocol, and
- a reactive replication scheme.

Our solution performs well in means of data accessibility, and does not create too much message overhead to the network with the increasing number of nodes. These properties are achieved with the idea of constructing a virtual and dynamic backbone that minimizes the number of nodes in the network involved in searching a specific data item. Virtual backbone construction algorithm is based on an approximation of minimum connected dominating set construction problem in graph theory. Scalable data lookup protocol takes the advantage of using a backbone which dominates the set of connected network nodes. The distributed data replication scheme, constructed on top of the scalable data lookup protocol, increases data availability and provides lower message overhead to the system without putting any extra message cost. It runs in a passive mode, which means it does not use any dedicated replication-protocol-specific control packets. Thus, it can completely eliminate the control overhead caused by active replication protocols. This is a valuable virtue for any protocol aiming the scalability. Access frequency and distance (hop count) of an item to the requesting node are the replication decision parameters in this scheme. Basically, nodes are eager to replicate data items that are further from them with higher request frequencies.

3.1. System model

System environment is assumed to be a meso-scale to large-scale mobile ad hoc network. Each node in the system has a unique host identifier. The set of all nodes is denoted as $M = \{M_1, M_2, \ldots, M_N\}$, where N is the total number of nodes in the network. Initially, each node M_i is the owner of data item d_i, where the set of all data items are denoted as $D = \{d_1, d_2, \ldots, d_N\}$. Every M_i can save replicas of data items in set D, limited with its memory capacity. Nodes are assumed to be identical with equal memory capacity and cannot be in any kind of selfishness. Every node, M_i, is aware of existing data set, D, and can request any data item d_j at any time.

3.2. Virtual backbone construction

We are inspired by the connected dominating set (CDS) approach as the virtual backbone construction mechanism, and provide a distributed implementation for our problem. For the construction of a virtual backbone, a CDS of the unit-disk graph of network topology is used.

3.2.1. Connected dominating set

A dominating set (DS) of a graph $G = (V,E)$ is a subset of vertices $S \in V$ in the graph G, where every vertex $v \in V$ is either in the subset S, or adjacent to a vertex in the subset S. Fig. 1(a) highlights a dominating set in a graph. A connected dominating set (CDS) is a DS whose induced subgraph is connected as illustrated in Fig. 1(b).

A CDS of a unit disk graph of a MANET can be useful for data lookup in the system and dynamic routing of packets. Hence, the search space of data or route is limited to the CDS. On the other hand, the computation of the minimum-sized connected dominating set over a graph is an NP-complete problem. It is because approximations are used in practice [31].

3.2.2. CDS construction algorithm

A simple distributed algorithm in order to find a CDS in a unit disk graph $G = (V,E)$ of a MANET has been proposed in [32]. CDS construction phase of SCALAR is inspired by this earlier work. The algorithm we use for CDS construction has the following properties:

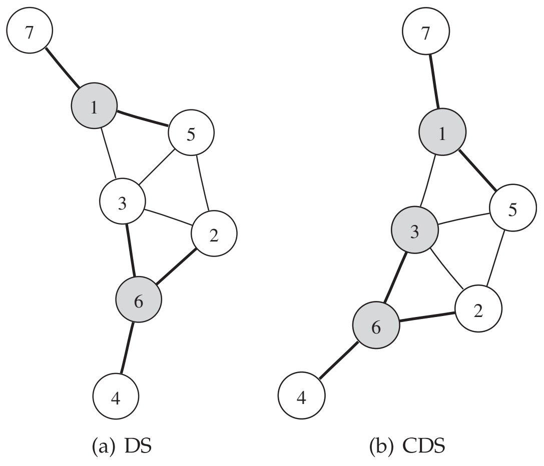


Fig. 1. DS and CDS illustrations of a given network.

- Every node requires only local or single hop information about the network topology. $N(u)$ is the (open) neighbor set of node u and $N[u] = N(u) \cup \{u\}$ is called closed neighbor set of u.
- Resulting set of nodes forms a dominating set; and every one of them is directly connected to at least one other dominator node.
- All other nodes, which do not exist in the CDS, are directly connected to at least one node in the resulting set.

Algorithm starts with a neighbor list exchange among nodes. In this step, every node u exchanges its neighbor list with all of its neighbors, $N(u)$. After sending and receiving all neighbor lists, every node moves to the *marking process*.

Marking Process: Initially all nodes start as marked false (i.e. not a dominator). A node marks itself as true (meaning that it is a dominator node) if it has at least two non-adjacent neighbors in its neighbor list. If G is fully connected, there will be no dominator in the network after the marking process, which is meaningful since every node is a neighbor of every other node in the network.

After the marking process, the initial CDS, V', is formed by the nodes that are marked as true. Then, nodes in V' go into *pruning* phases in order to form the final set of CDS nodes. There are two rules for each pruning phase given as follows:

Rule 1. Consider two vertices v and u that are marked as true. If $N[v] \subseteq N(u)$ in G and $id(v) < id(u)$, then change the mark of v to false (not a dominator).

Rule 1 indicates that if any neighbor of v is also a neighbor of u, and v is connected to u and has a lower *id* value, then any path including v can be replaced by u; in other words v covers u. The resulting CDS ($V' = V' - v$) after Rule 1 is still a connected dominating set of G.

Rule 2. Assume that, after applying Rule 1, u and $w \in V'$ and they are neighbors of $v \in V'$. If;

- $N(v) \subseteq N(u) \cup N(w)$ in G and,
- $id(v) = \min\{id(u), id(v), id(w)\}$

then v is marked false and deleted from the CDS V'.

In other words, Rule 2 indicates that if any two dominator neighbors of v covers v, then v can be eliminated from the CDS. Again resulting set $V' = V' - v$ after Rule 2 is still a CDS.

After applying two pruning rules to the initial CDS, the algorithm finalizes the construction of CDS. The entire construction is completed in two rounds (time complexity) and the message complexity is $O(n)$, where n is the number of nodes in the network.

3.2.3. Distributed implementation

We have developed and implemented a distributed CDS construction algorithm that works as follows: Each node first broadcasts its neighbor information to all of its one-hop neighbors. After receiving the same information from its neighbors, it declares itself as dominator (a member of CDS) if and only if it has two non-adjacent neighbors. Then it goes into pruning phase, and drops itself from the CDS, if the rules mentioned in the previous section apply. During the CDS construction, if a data from the neighbor list of a node cannot be received due to collision or mobility of nodes, the procedures can continue to work with lower performance values such as increased CDS size.

In order to maintain the CDS in mobility scenarios, we recalculate the CDS by periodically invoking the construction algorithm. Period of the CDS reconstruction depends on the mobility behavior of nodes. If nodes are moving fast, frequent calls of reconstruction may be necessary to support the connectedness of dominating set. However, it should be noted that, even with frequent reconstruction of CDS, it is not guaranteed that nodes determined as dominator form a CDS at any time instance. Thus, the algorithm proposes a best-effort solution for the construction of CDS in MANETs.

3.2.3.1. Cost analysis. Message complexity of our distributed implementation is $\Theta(n)$, where n is the number of nodes in the network. Distribution of packets sent is as follows:

- Hello Packets: $n \rightarrow$ Every node sends exactly one at the start.
- Neighbor List Packets: $n \rightarrow$ Every node sends one.
- Dominator-Announce Packets: $d \leqslant n \rightarrow$ Every node marked as dominator broadcasts one. d is the size of initial CDS constructed.
- Dominator-Cancel Packets: $m < d \leqslant n \rightarrow$ Every dominator node marked as non-dominator after applying Rules 1 and 2 sends one of this packet.

Number of total messages sent is $t = (2n + d + m)$, where $2n \leqslant t < 4n$.

3.3. Scalable data lookup protocol

Scalable data lookup protocol searches a data item in the system by sending the request to the virtual backbone members, to which every other node is directly connected. Requests are only forwarded among the backbone nodes, and as a result the number of nodes involved in the lookup process is kept limited. This also decreases the message cost, which is an important burden in large-scale mobile networks.

A straightforward solution to the data lookup problem in ad hoc networks is *flooding* the request to the network. Another solution is requesting the data item directly from a specific node, which every node in the system can match the identification of the requested data item with. This requires the execution of a routing protocol by all network nodes (i.e. AODV [33], DSR [34]). Both of these solutions are shown to result in serious contention and collision in large-scale wireless networks [21,10]. Our solution aims to meet the necessities of large scale MANETs while keeping the simplicity of straightforward ones. In the rest of this section, details of our scalable data lookup protocol will be given and its advantages over basic data lookup solutions will be discussed.

3.3.1. Node types

Our protocol assumes that every node in the system is either: (*a*) backbone (dominator) node, or (*b*) end system (dominatee). Nodes are assigned one of these types during the virtual backbone creation phase. The decision of being a backbone node purely depends on the nodes' connectivity information during the virtual backbone creation. End systems are only allowed to send their data requests to one of their neighbors in the backbone. We name this as *request injection*. Backbone nodes form the basis of the data lookup process. We divide this process into two parts: *search* and *data receive*.

3.3.2. Search

A *backbone node* participates in the search process via one of the following:

(1) *Sending* a request generated by itself to a set of backbone nodes in its neighborhood.
(2) *Forwarding* a received request to a set of backbone nodes in its neighborhood.
(3) Receiving a request *generated* from an end system or backbone node.
(4) Receiving a request *forwarded* from a backbone node.

In 1 and 2, a backbone node injects a data request into the backbone. A backbone node, M_i, decides to inject a new request for item d_j if none of the cases below are satisfied:

- d_j is owned by the backbone node M_i.
- d_j can be found in two-hop vicinity of M_i, where one-hop neighbors are backbone nodes.

Recall that every node already knows its neighbors and neighbors of its neighbors (i.e. two-hop vicinity) from the virtual backbone creation phase. If it is the first case, data is sent directly to the node that the request is received from. If it is the second case, the request is forwarded to the appropriate backbone neighbor who either owns the data or is on the path to the owner.

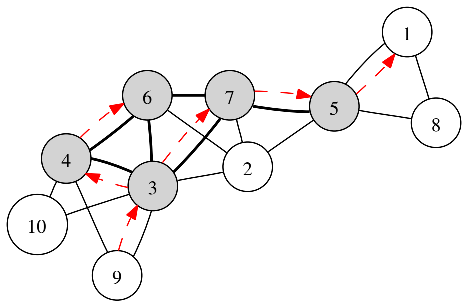


Fig. 2. Example scenario: Node 9 requests data item 1. Dashed lines represent the propagation of request packets.

End systems do not have active role and responsibility in scalable data lookup protocol. They do not forward any request or data packets. Basically, end systems inject their requests to the backbone in order to search a data item in the network by exploiting the dominating set property of the virtual backbone structure. After a request is injected into the backbone, it is the backbone's responsibility to search the requested item and transport the item to the requester. Similar to backbone nodes, if a requested data item exists in end system node's two hop vicinity, request is directly sent to the appropriate neighbor backbone member. Otherwise, request is injected via randomly selected backbone node.

During this request forwarding or new request injection, every node involved in the process saves a ⟨*id of requester node, hop distance to requester node*⟩ tuple of every forwarded or created request into a data structure (i.e. a hash table) with a unique key:

⟨`requested item`, `originator`, `packet id`⟩

The hash table is then used to route the received data packet to the destination, and is updated when the data is received on the way back. Note that, since each received request is kept in a hash table entry with a unique key, a node can identify the previously received requests and can ignore redundant requests.

3.3.3. Data receive

A *backbone node* participates in the data receive process by one of the following cases:

(1) If it receives a request for the data item, which it owns.
(2) If it receives a data packet from a backbone or end system node for which a data request is forwarded during the searching phase.
(3) If it is the originator of the request.

In case 1, requested data is packed into a network packet with a unique key and sent to the neighboring node from which the request is received. The key of the sent data packet is defined as a tuple of:

⟨`sent item id`, `destination node of the data packet`, `req.packet id`⟩

If case 2 holds, backbone node checks its requested items data structure (hash table) using the key of the received data packet (given above) in order to match the received data with a requester node id. If such a requester is found in the requested items data structure, then received packet is directly forwarded to it and this request is removed from the requested items list to avoid multiple transmissions of same data packet. If it is not found, received packet is ignored. Finally, if case 3 holds for a backbone node, it checks the key of the received packet in its requested items data structure. If requester is itself, it puts the data in its memory space and completes the process.

Data receive is simple in *end systems*: Every data packet received is the request of this node. Multiple data reception for the same request is not possible because this problem is handled in backbone nodes before coming to end systems. In SCALAR, obviously, backbone nodes consume more energy and bandwidth to support the requests and data reception of other nodes. On the other hand, as stated before, virtual backbone is dynamic and reconstructed periodically to support the location changes of nodes due to mobility. Thus, it is highly possible that some of the end systems become backbone nodes at some period or vice versa.

3.3.4. Example scenario

In Fig. 2, highlighted network nodes are backbone nodes and links connecting backbone nodes are bold. All other nodes are end systems. The scenario discusses the case in which node 9 requests data item 1 at time t. Between t and $t + \Delta t$, unit disk graph representation of MANET is shown in Fig. 2. We assume that Δt is a time period enough to complete the given steps below:

(1) Node 9 requests data 1 (owned by node 1), and checks its neighboring nodes list looking for a backbone node. It selects one backbone node randomly (in this case 3) and sends the request to that node.
(2) When node 3 receives a data request, if it owns the requested item, it sends it to 9. However in this case, data 1 is not owned by 3. Thus, it checks its one and two hop neighbors. If owner of data can be found in one of them, it forwards the request to the appropriate neighbor. In this case, node 3 cannot locate node 1 from 2-hop connectivity information. So it forwards the data to a set of random backbone nodes in its vicinity (set of 2, in this example). Note that number of nodes to forward in the backbone can be set based on the node density.
(3) When nodes 4 and 7 receive the request from 3, they perform the same checks as 3. During these checks, 7 finds that node 1 is in the vicinity of its backbone neighbor 5. Node 7 forwards the request to 5. On the other hand, node 4 has no idea of where node 1 can be. It forwards the data to a set of random backbone nodes as 3 did (in this case only 6).
(4) Node 6 forwards the request to a random set of backbone neighbors. In our illustration, we do not show the ignored requests due to duplication, for visual clarity. Actually, in this case nodes 3 and 7 ignore the request coming from 6.

(5) When node 5 receives the request from 7, it forwards the request to node 1. Node 1 packs the data and sends it to 5. After that, every node involved until request come to node 1 is used as part of the return path.

3.3.5. Message complexity

The idea behind using a virtual backbone structure for data lookup in an ad hoc network is to decrease the number of hosts that receive the request while keeping the availability of the data item at the same level with flooding-based solutions. The worst-case message complexity of our algorithm is directly related to the number of nodes in the backbone (V'). Since every backbone node can forward a unique data request for only once to k of its dominator neighbors, and only backbone nodes can forward data requests, number of messages sent in the network for a data request can be bounded to $O(k * V')$ where $G' = (V',E')$ represents the reduced graph constructed by the CDS of network topology graph G. This result shows that scalability of our protocol depends on the size of the connected dominating set. Fig. 3 shows the increase in the size of virtual backbone as the total number of nodes in the network increases. The graph is generated from our simulation results of distributed CDS implementation. Since finding the minimum CDS is NP-complete, better approximations used in virtual backbone construction phase of SCALAR increases the data lookup protocol scalability.

3.4. Reactive replication

Our Reactive Replication (RR) mechanism is built on top of the scalable data lookup protocol. It is *reactive* since replication decision for a data item is done when the data is received, and it does not require the exchange of explicit replication control packets. Thus, *RR* eliminates the control overhead caused by other active replication schemes, and it does not put extra cost on the scalable data lookup protocol. Furthermore, it increases the probability of finding a data item in a closer node on the virtual backbone, by this way it helps to decrease the overall messaging cost of the scalable data lookup protocol.

Both backbone and end system nodes can make a replication decision by applying the corresponding rules (see Section 3.4.2. In general, replication decision of a data item is based on *distance* to the data owner and its *request frequency*. RR aims to replicate the distant data items in order to decrease the number of requests propagated in the virtual backbone. Besides the distance (hop count), RR also considers the request frequency of an item during its replica allocation decision. Thus, if a data item is requested frequently in a specific time period (although its hop count is smaller than another received data), it may be selected for replication. *RR* regulates the local caches of the nodes so that costly requests are cached preferably. Cost of a request is calculated using a common function for all node types as described next.

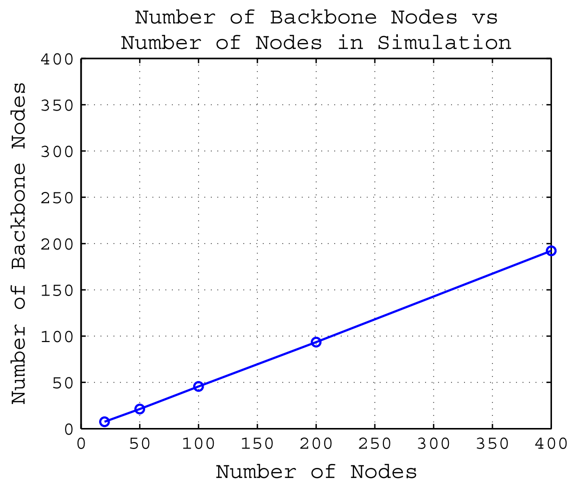


Fig. 3. Number of backbone nodes.

3.4.1. Cost function

Our cost function uses request frequency history and distance (number of hops) information to the data owner. RR aims to replicate frequently accessed distant data. At node M_j for a data item d_i, the cost function is defined as:

$$cost(\alpha_i, h_{ij}) = \frac{\alpha_i}{\sum_{k \in D} \alpha_k} * h_{ij} \tag{1}$$

where α_i is the local request frequency history of data item d_i and h_{ij} is the number of hops between node M_j and the node that data item d_i is received from. D is the set of all the data items available in the system. Basically, this function gives higher costs to the data items requested frequently in the past and that are distant to the requester node. By means of this cost function, replicating data items with larger costs will help to minimize the cost of the successive requests to the system. This is because, distance h_{ij} will be reduced in the cost calculation of next requested data item using a system-wide replication. Furthermore, an extra backbone node replica in the mid point of the data path will also help to improve data availability if the data path is very large. We believe that this behavior makes sense in large scale networks. With the reactive replication mechanism, during the search for the owner of a data item, if a replica is found in the backbone, then the search process stops and data receive process starts.

3.4.2. Replication decision

A backbone node gives a data replication decision if one of the following cases is true, as illustrated in Fig. 4.

- If received data is the backbone node's own request (not a forwarded request) and cost of the received data item is at least as large as the lowest-cost item replicated in a full cache, then the lowest-cost item is replaced with the newly received data item. If the cache has vacant space for a new item, then node does not need to make a cost comparison for keeping a replica of the data item. This is for maximum utilization of cache spaces.

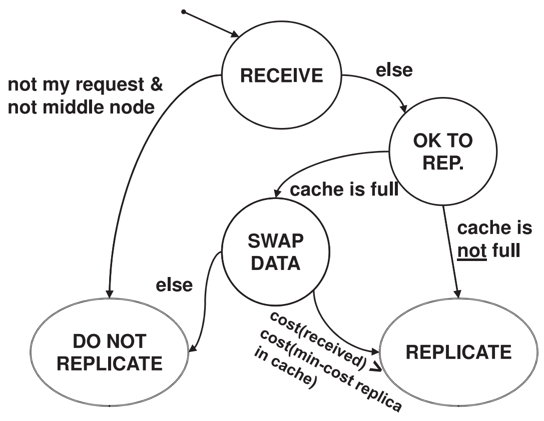


Fig. 4. Backbone – Replication decision tree.

- If the data item is received due to a forwarded request, then the backbone node checks its position on the path between receiver and sender of data (by comparing the hop counts of forwarded request and received data packet). If it is the middle node in the path, it may decide to replicate the data item. Decision of replication is based on comparing the costs of received data item and lowest-cost item in a full cache. However, if the cache has vacant space, then the node replicates received data directly.

An end system's replication decision is illustrated in Fig. 5. In contrast to backbone nodes, since end systems do not take place in data request forwarding, they do not give replication decisions about being on the middle point of data path. Therefore, an end system gives a data replication decision if and only if the following condition is true:

- If cost of the received data item is at least as large as the lowest-cost item replicated in a full cache, then the lowest-cost item is replaced with the newly received data item. However, if the cache has vacant space for a new item, then the node replicates the data item directly.

3.4.3. Example scenario

Fig. 6 represents a replication decision example for end system node 9(M_9) when it requests the data item 1, d_1. When M_9 receives the d_1, its memory is full with the data item d_6, which was replicated in a previous step. When d_1 is received, M_9 needs to make a decision about which data to replicate. At this point instructions are clear for M_9: If the cost of the received data item is as large as the minimum-cost item in the memory (d_6), then put the received data in place of it. In our case, assuming that the cost of d_1 (calculated with *cost function*) is larger than the cost of d_6, M_9 replicates d_1 instead of d_6.

4. Experimental preliminaries

In this section, we describe our simulation environment, settings and metrics used for performance evaluation.

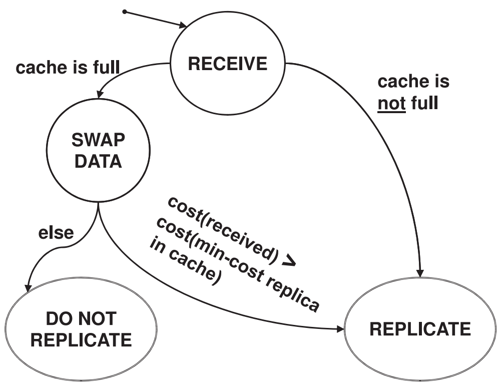


Fig. 5. End system – Replication decision tree.

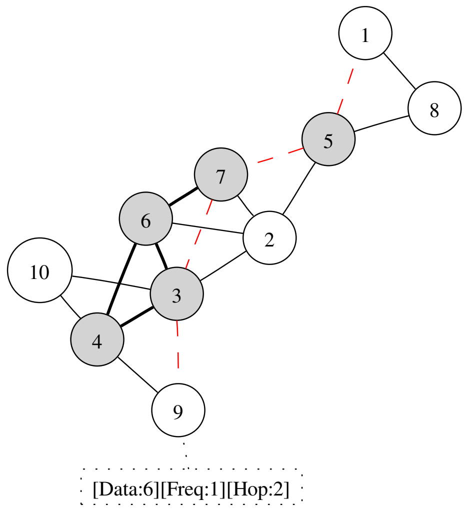


Fig. 6. Example replication scenario: Node M_9 requests data item d_1, while its memory is full with d_6.

4.1. Simulation environment

We have developed the model of SCALAR on SWANS (Scalable Wireless Ad hoc Network Simulation) tool [35]. SWANS is built on Java-based simulation framework JiST (Java in Simulation Time). Its capabilities are similar to *ns*-2 or *GloMoSim*, moreover it is able to simulate much larger networks, thanks to JiST's high performance structure [35].

Simulation environment is a meso-scale (from 20 nodes) to large-scale (up to 400 nodes) mobile ad hoc network. When we increase the number of nodes in the simulation, we also apply a proportional increase to the simulation area while keeping the density of all areas constant. Our motivation for a constant-density network is to avoid from the side effects of possible collision and contention on the shared wireless channel. Density of a network is calculated as follows: *density* $= \frac{N*S}{A}$, where N is the total number of nodes in the simulation, S is size of the coverage area of a mobile node and A indicates the total simulation area size. Default simulation parameters are given in Table 2. In our simulations, every node can save up to 5 data items, and each simulation runs for 300 s.

4.2. Performance metrics

We analyze the performance of our system using the following metrics:

- *Success Ratio:* This is an important criterion indicating data accessability for data lookup and replication protocols. Success ratio is the ratio of the number of successful access requests to the number of all access requests issued. A data replication protocol aims to increase the accessibility of data items in the network. Different than conventional static networks, in mobile environments achieving 100% data accessibility is nearly impossible, due to mobility of nodes and changing network topology.
- *Average Query Deepness:* The average number of nodes (or hops) traversed by a query when finding the requested data is called as average query deepness. Essentially, query deepness is the distance of the requester to the requested data item in terms of number of nodes on the successful path found. *SCALAR* with replication aims to replicate the data items found at further nodes in order to decrease the query deepness of a data request. Since the number of nodes involved in a request forwarding decreases with the lower query deepness values, average cost of a data request to the whole system is also expected to decrease. Number of hops that a data item found is also directly related with query completion delay. Some networking applications (such as VoIP (Voice over IP), and real-time video broadcasting applications) require strict timing constraints. Lower average query deepness values result in lower delay for data requests.
- *Packets Sent per Node:* Packets sent per node is defined as the average number of packets sent by a node. It includes every packet sent from a node, i.e. routing layer control packets, and upper layer protocol packets such as data request and data packets. Broadcast packets and unicast packets are both counted as one packet in this calculation. Basically, this metric shows the message overhead of a solution to the data lookup problem with/without replication.

5. Performance results

In this section, we analyze the performance of SCALAR framework in various network conditions and compare our results with a data lookup scheme, in which the path between requester and data source nodes is found using Ad-hoc On Demand Distance Vector (AODV) routing protocol. Due to its popularity, reactive property, and usage in existing replication methods [15], we choose AODV routing as the underlying protocol for data lookup comparison. In our simulations, we aim to examine the performance of SCALAR in *large-scale* and *high density* network conditions, in which most of the data lookup and replication approaches fail.

Table 2
Simulation defaults.

MAC protocol	802.11b
Mobility model	Random waypoint
Network protocol	IP
Radio range	100 m
Transport protocol	UDP
Node speed	1–3 m/s

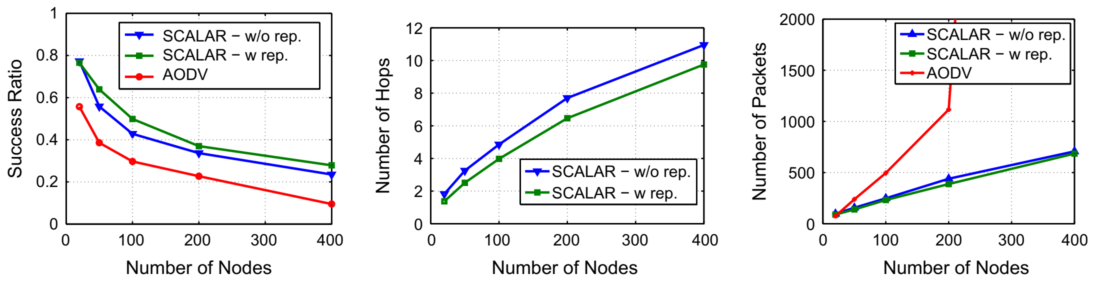
5.1. Scalability

Fig. 7(a) shows that as the number of nodes in the network scales up, in every data lookup approach the success ratio decreases. This can explained by the increase in the average query deepness as the number of nodes increases as shown in Fig. 7(b). As the average query deepness increases, the number of nodes involved in the transmission of a data request increases. The nodes are mobile and as the number of nodes involved in this process increases, the probability of occurrence of a disconnection in the path from requester to data source increases. This disconnection could cost an incomplete (or unsuccessful) data request to the system. On the other hand, it can be concluded that SCALAR performs better than AODV based data request in terms of data accessibility.

Fig. 7(b) also shows that using SCALAR with replication achieves better in terms of query deepness and hence delay times, as expected. The improvement with the replication case can be higher if the number of data items that every node replicates increases. Fig. 7(c) shows the average number of packets sent per node. It is obvious that as the number of nodes increases in the network, message overhead introduced by AODV based data request increases exponentially. This causes a lot of contention and collision, plus fills the message queues of nodes resulting in packet drops at the MAC layer. On the other hand, SCALAR can bound the message overhead to very low levels.

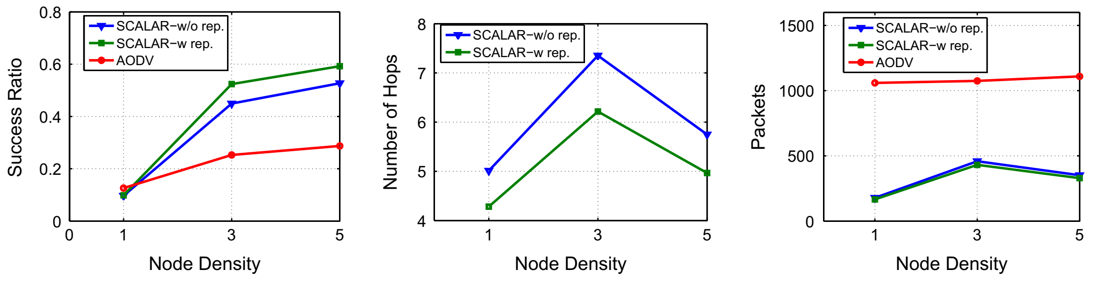
5.2. Node density

As shown in Fig. 8(a), SCALAR achieves considerable increase in the success ratio as node density increases. However, AODV based data lookup scheme does not scale in terms of data accessibility with the increasing node densities. This is due to the high message overhead introduced in AODV with flooding of routing packets. As the density increases, negative side-effects of flooding broadcast messages become visible in the performance values of AODV. An interesting result in Fig. 8(c) reveals that SCALAR tends to send fewer packets per node when the density of the network increases over a threshold value (where the number of backbone nodes is at maximum), i.e. 3. This is because in dense networks, the size of connected dominating set is smaller due to the characteristics of the algorithm used in virtual backbone construction. On the other side, Fig. 8(b) shows that SCALAR with replication achieves lower query deepness than without replication. In fact, this is the target of RR in our proposed method: to lower the query deepness of a request and hence decrease the delay and message cost.



(a) Avg. Success Ratio (b) Avg. Number of Hops (c) Avg. Packets Sent per Node

Fig. 7. Effect of increasing number of nodes. Simulation area size is calculated based on density value 2. Number of data requests per node is 10.



(a) Avg. Success Ratio (b) Avg. Number of Hops (c) Avg. Packets Sent per Node

Fig. 8. Effect of increasing simulation area node density. Simulation area size is calculated based on constant 200 nodes. Number of data requests per node is 10.

5.3. Effect of network load

We investigate the effects of varying data requests per node that is *network load*. Network load is varied from 2 requests/s to 16 requests/s on a 200-node MANET scenario. Results in Fig. 9(a) show that as the network load gets larger, SCALAR, both with and without replication, performs better than AODV-based protocol in terms of data accessability. However, in low network load of 2 requests/s, AODV achieves higher data accessability. This is due to the fact that when the network is not highly loaded, number of packets transmitted in the network at any given time is also low. As a result of this, packet loss due to collision and contention in wireless channel is at minimum in the network. In this case, AODV performs better than SCALAR, since SCALAR performs a probabilistic search method in the network. However as the network becomes more and more loaded with data request packets, AODV cannot adapt itself to this increase. On the other hand, SCALAR can keep higher levels of accessibility ratio even at higher network loads.

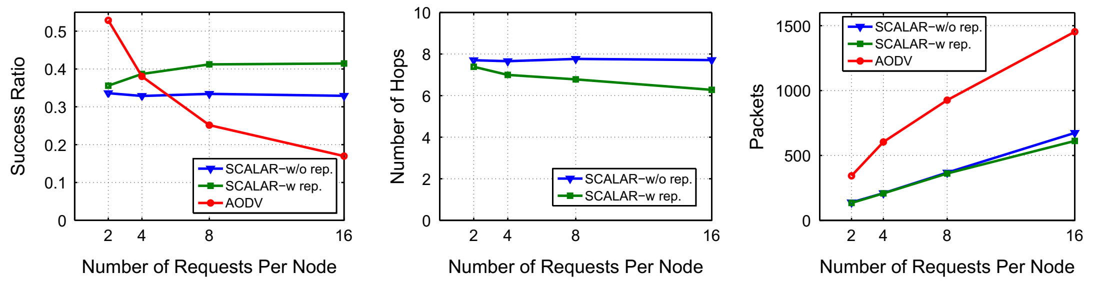
As shown in Fig. 9(b), average query deepness for SCALAR with replication decreases as the network load gets larger. SCALAR without replication keeps the average query deepness stable with the increasing load. Fig. 9(c) depicts the message overhead of each protocol as a function network load. The increase in average number of packets sent per node is much larger for AODV in comparison to SCALAR for all network load simulations, in particular for higher loads.

5.4. Node memory space

The number of data items that a node can save in its memory can increase the performance of data replication, and large memory sizes help us to see the effect of replication on data accessibility better. In Fig. 10(a), since SCALAR without replication and AODV do not replicate data items, increasing the size of memory does not affect the success ratio. However, when using SCALAR with replication, we observe the increase in data accessibility as expected. Fig. 10(b) also supports the effectiveness of using data replication by demonstrating the decrease in the average query deepness of completed data requests. An important observation from Fig. 10(a) and (b) is that, after a threshold value of size of memory space (8 in this case), increase in accessibility and decrease in query deepness decelerates. We believe that this happens because for 200-node simulation, every node replicating 8 data items gives the same data variability as replicating 16 data items per node. This threshold value could be different depending on the network size and network conditions. Optimal memory space for a specific network condition can be analyzed further and represented as an optimization problem. Fig. 10(c) reports overhead results showing that AODV creates larger message overhead in the network.

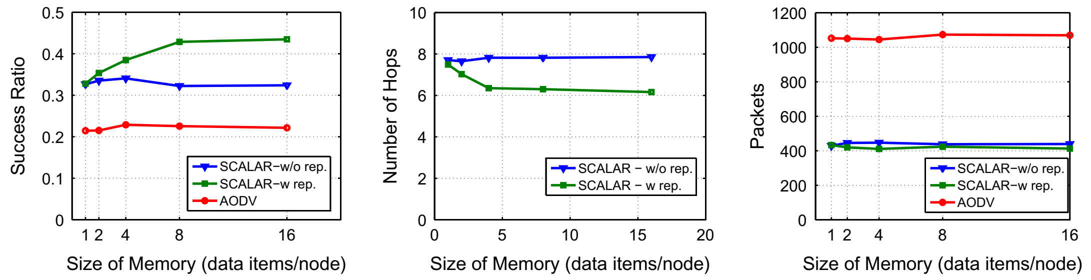
5.5. Backbone nodes and connectivity relations

Fig. 11(a) depicts the average number of backbone nodes as a function of number of nodes in the network,



(a) Avg. Success Ratio (b) Avg. Number of Hops (c) Avg. Packets Sent per Node

Fig. 9. Effect of increasing data requests per node. Simulation area size is calculated using 200 nodes and simulation area density as 2.



(a) Avg. Success Ratio (b) Avg. Number of Hops (c) Avg. Packets Sent per Node

Fig. 10. Effect of increasing memory space per node. Simulation area size is calculated using 200 nodes and simulation area density as 2.

for different density levels from 0.5 to 5. For larger networks, more backbone nodes are needed to protect the CDS property of the given network graph. Note that, in order to keep the density of a network at a constant rate, we increase the simulation area while increasing the number of nodes in our simulations. As shown in Fig. 11(a), for high density networks, virtual backbones with less number of backbone nodes can be constructed. Actually, this is a natural result of CDS construction algorithm, because in denser networks the algorithm can prune more backbone nodes. In other words, as the density of network increases, the probability of finding a backbone node that covers all the neighbors of another backbone node increases which is the necessary condition for Rules 1 and 2 to prune a backbone node. Thus, we can conclude that SCALAR scales well to the increasing node densities in terms of the number of backbone nodes.

Likewise, Fig. 11(b) presents results for the number of backbone nodes as a function of node density, for different network sizes. In this case, it is observed that as the density increases, number of backbone nodes in the network increases up to a level and then starts to decrease. In very low density levels, most of the nodes are disconnected and probably there are not many nodes in each disconnected partition and results in less total backbone nodes. When the density increases, the network starts to become more connected. As a result, the number of backbone nodes also increases. However, after some threshold value (2 in this case), increase in density results in a lower number of backbone nodes due to the increase in the probability of finding a backbone node that covers all the neighbors of another backbone node. We observe that on average the number of backbone nodes is not more than 50% of network size. This means that using a virtual backbone, in the worst-case we can flood a request packet to half of all nodes, and can give the same accessibility of flooding to all nodes in the network. This property of SCALAR helps it to achieve low overhead in large and dense networks. In Fig. 11(c), we present the average connectivity of network nodes for increasing simulation area node densities. It depicts a natural result: as the density increases, the average connectivity increases.

5.6. Node mobility models

Movement behavior of mobile entities is an important concept for the realistic simulation scenarios in MANETs [36]. As argued in [37], choice of a mobility model may affect the results of simulations significantly. In order to demonstrate the behavior of SCALAR in case of different mobility models, we implemented Random Walk mobility model on JiST/SWANS simulator and conducted simulations. As shown in Fig. 12(a), simulations using random walk model performed slightly worse than random waypoint mobility model in terms of data accessibility. As shown in Fig. 12(b), average query deepness for random waypoint is less than random walk as the network size gets larger. Likewise, Fig. 12(c) depicts that average message

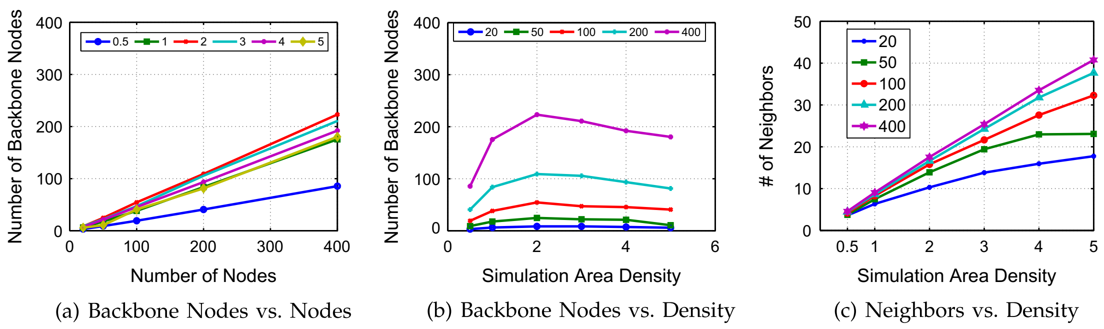


Fig. 11. Number of backbone nodes and connectivity relations for different network sizes and density levels.

overhead of SCALAR in random waypoint mobility is less than random walk case. However, in both mobility models, the behavior of SCALAR is similar. The slight difference between the mobility model scenarios can be explained by the definition of models. Random walk model does not introduce pause times for nodes. However in random waypoint model, every node waits during pause period (set to 5 s in simulations) after reaching a destination (or waypoint). Pause times increase the stability of network nodes during the simulation, and as a result random waypoint can give better accessibility ratios compared to random walk.

5.7. Fairness

We investigate the fairness of SCALAR in balancing the load of becoming a backbone node in the network. Fig. 13 shows that nodes with larger IDs have a higher probability of becoming a backbone node, since the virtual backbone construction algorithm tends to prune the backbone nodes with smaller IDs. Simulations are performed for 100 nodes in a simulation area with node density of 2. x-axis of Fig. 13 shows the node IDs and y-axis represents the number of times that each node become a backbone node during the entire simulation (500 s). Note that, in these simulations, since a virtual backbone is reconstructed in every 12 s, a node can become a backbone node 42 times at maximum.

6. Comparative results: Application scenarios

In this section, we compare the performance of SCALAR with SAF and DAFN [5] replication approaches on realistic application scenarios. We also examine the effect of data request probability and percentage of popular items, and perform analysis of SCALAR's cost function, its comparison with random replica replacement, and investigation of success ratio per node.

6.1. Application scenarios

We present two application scenarios in which university campus and shopping mall settings are simulated. These scenarios mainly differ in terms of node density where mobile agents are assumed to be people using wireless devices such as laptop computers, PDAs and cellular phones. By means of these scenarios, our aim is to investigate the behavior of SCALAR in realistic application settings for MANETs.

6.1.1. University campus scenario

In this scenario, we create a simulation environment where node density is low (average number of neighbors is smaller than 10). Mobile nodes are assumed as wireless devices belonging to pedestrian university students with velocity range 1–3 m/s which is the average walking speed of human being. The memory spaces of the devices are

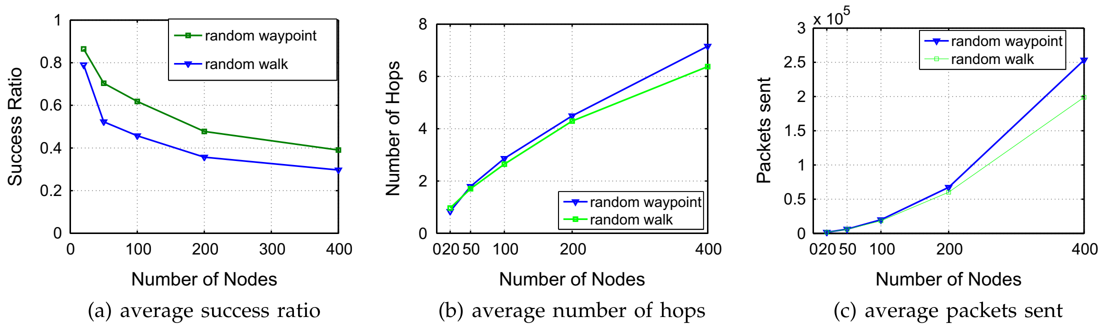


Fig. 12. Effect of mobility models: random walk and random waypoint.

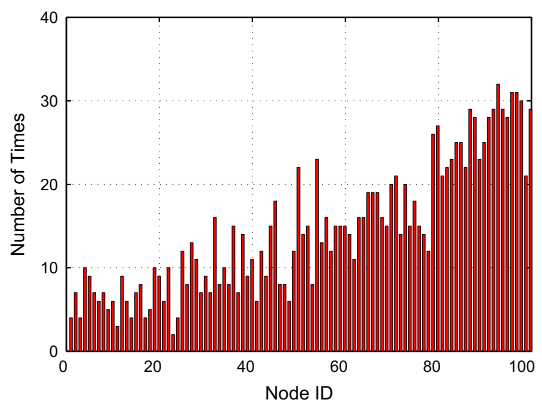


Fig. 13. Number of times each node (x-axis: node id) become a backbone node during the simulation.

available to keep 5 data items for replication. Some of the data items in the network are set to be popular. These popular data items have a higher request probability (σ). In other words, probability of requesting a popular item (at each data request generated) is determined by σ. The probability σ is set to 0.6 by default, and we also investigate the effect of varying σ. The popular data items in the system are selected randomly. Percentage of popular data items in the system is set to two different values (10% and 70%). Table 3 summarizes the simulation parameters.

A possible example of this scenario can be as follows: In a university campus (which we assume a large campus area in comparison to student population with low density), a group of students having cell phones, PDAs and laptops are communicating over IEEE 802.11b in ad hoc mode. Every student owns a unique data item (personal lecture notes, MP3s, etc.) that every other student in the network can be interested in. Students can be mobile or static at any given time and are eager to participate in this system. Mobile students are walking in the campus area, and pausing and talking to other people during their travels.

6.1.2. Shopping mall scenario

In this scenario, a shopping mall is simulated. It mainly differs from the university campus scenario in terms of node density, assuming a shopping mall is a denser place. Concepts of node mobility, popular data items and their request probabilities are similarly applied to this scenario. Table 4 summarizes the simulation parameters. Overall, by means of these scenarios, we examine the effect of node density over the system as well as the percentage of popular data items and their request probabilities.

6.2. SAF and DAFN replication approaches

We compare SCALAR with the SAF (Static Access Frequency) and DAFN (Dynamic Access Frequency and Neighborhood) approaches in the application scenarios. A brief explanation of these approaches are presented in Section 2.1. Since DCG (Dynamic Group Connectivity) method proposed in the same study [5] is not applicable to the distributed implementation in real networks, it is not used in our simulations. In contrast to SCALAR, these approaches do not offer a complete data lookup and replication solution. They only provide a data replication decision algorithm based on data access frequencies and neighborhood information. Data lookup phase and communication details between requester and source node are not specified in [5]. Our implementation of SAF and DAFN replication systems is based on AODV routing algorithm to request and receive data items, since it is less costly compared to flooding of request packets in the entire network. We now describe execution of each approach in our simulation environment.

In SAF, at the beginning of simulation, every node checks its access frequencies table and selects the highest frequency x data items and decides to replicate them. Node broadcasts a request message for each item and this request is sent into network with the help of neighboring nodes. When a node receives a request message of another node, it checks its memory space for the requested item. If it can find the item, it sends the data to the requester. If it cannot find the item, it broadcasts the received request to its neighbors. If the requester receives the data, it puts it into its memory space and finishes the SAF process.

DAFN starts with an initial SAF execution. But before requesting data items, every node checks what its neighbors are replicating (this information is provided with a periodic broadcast message received from its neighbors). If the same item is selected as candidate for replication, then it compares its access frequencies with its neighbors' for that data item. Node with the highest access frequency for that data item replicates the data. Other one selects another item to replicate. This process is periodically executed.

6.3. Comparison

Our findings that compare SCALAR, SAF and DAFN data replication approaches are as follows. In the low

Table 3
University campus scenario – Parameters.

Nodes	Area (m^2)	Velocity	1–3 m/s
20	316×316	σ	0.6 (default)
50	500×500	% of Pop. items	10% and 70%
100	707×707	Mobility model	R. waypoint
200	1000×1000	Req. per node	3 Requests
400	1414×1414	Memory size	5 Items/node

Table 4
Shopping mall scenario – Parameters.

Nodes	Area (m^2)	Velocity	1–3 m/s
20	200×200	σ	0.6 (default)
50	317×317	% of Pop. items	10% and 70%
100	447×447	Mobility model	R. waypoint
200	632×632	Req. per node	3 Requests
400	895×895	Memory size	5 Items/node

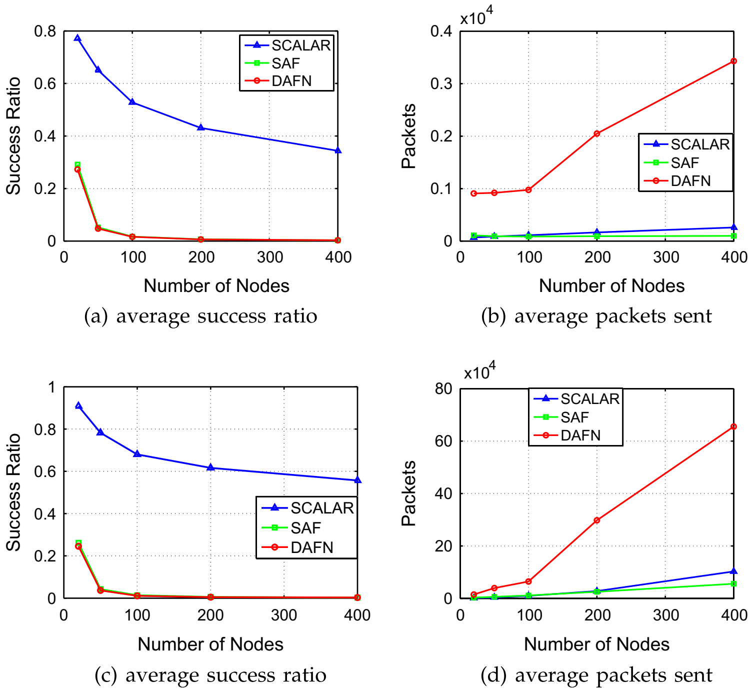


Fig. 14. Comparative simulation results: (a) and (b) University campus scenario, (c) and (d) Shopping mall scenario.

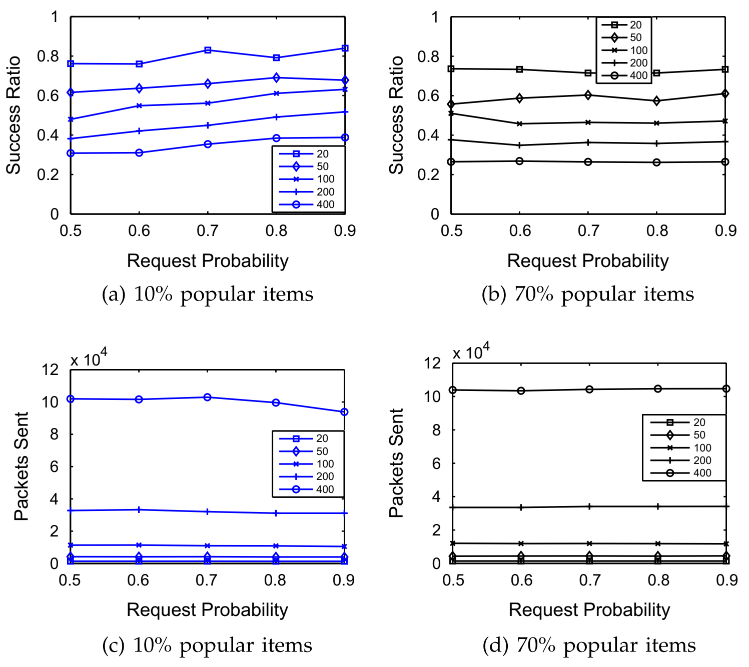


Fig. 15. University campus scenario: (a) and (b) Success ratio as a function request probability, (c) and (d) Packets sent as a function request probability.

node density university campus scenario, SCALAR achieves very high success ratio in comparison to DAF and SAFN. Simulation results in Fig. 14(a) and (b) show that for even smaller number of nodes, SAF and DAFN cause very low data accessibility and high message overhead. However, as the number of nodes increases, performance of each replication approach drops significantly. In each simulation, SCALAR outperforms SAF and DAFN replication in terms of data accessability. Besides, it is shown in Fig. 14(b) that number of packets sent per node is extremely high in DAFN due to periodic relocation of replicas.

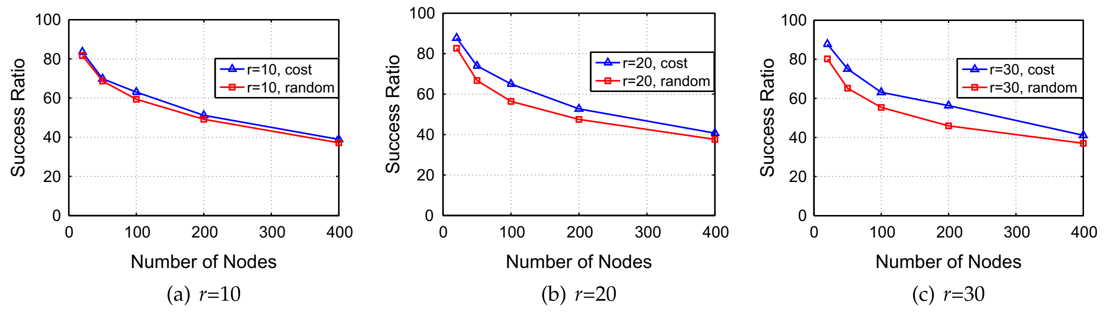


Fig. 16. Cost function analysis and comparison with random replacement.

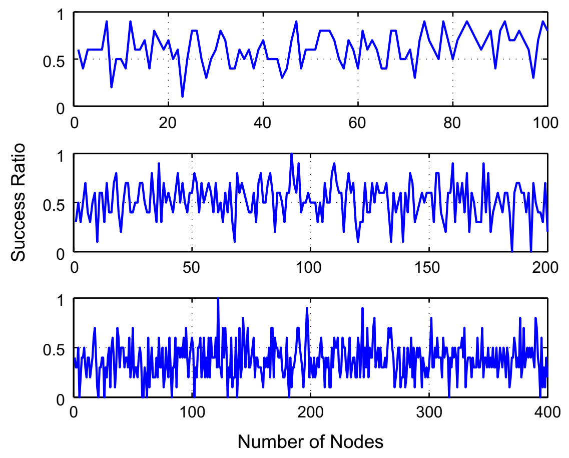


Fig. 17. Success ratio per node: 100, 200, 400-node networks.

In the high density shopping mall scenario, SAF and DAFN perform similar to previous scenario in terms of data accessibility as shown in Fig. 14(c). On the other hand, SCALAR improves its accessibility performance significantly. In high density networks, connectivity is high; so it is expected that data accessibility would increase. However, in SAF and DAFN, when the network becomes denser, broadcast storms and collisions in the wireless channel increase. As shown in Fig. 14(d), this results in high message overheads. Thus, even though connectivity is increased, data accessability performance does not improve for SAF and DAFN in this scenario.

6.4. Data request probability and percentage of popular items

The data request probability (σ) for popular data items is varied from 0.5 to 0.9. For the percentage of popular data items over all data items in the system, we consider two cases. The first case (10% popular data items) reflects the fact that a small percentage of data items are popular in the system, whereas the second case (70% popular data items) considers a system in which majority of data items are popular. Fig. 15 show the success ratio and overhead results obtained for the university campus scenario for different request probability and percentage of data items. We also analyzed corresponding results obtained for the shopping mall scenario. These performance results are evaluated as follows.

As shown in Fig. 15(a), when only a small portion of data (10%) is popular in the system and requested with a higher probability, success ratio increases proportionally for all network sizes as σ increases from 0.5 to 0.9. This improvement in success ratio as a function of σ also causes a drop in the number of packets sent (Fig. 15(c)). Thus, SCALAR performs better for the cases when a small percentage of data items are popular in the system and they are requested frequently in comparison to other data items. This behavior is valid for both low and high node density applications, and it is scalable as the network size increases. In fact, an example for the small portion of data being popular in the system could be a flash-crowd scenario. As soon as a popular data is released, a flash-crowd of data requests is expected in which many users strive to achieve a copy of the popular data.

When the ratio of popular data items is large (70%) in the system, the success ratio stays stable as σ increases. As shown in Fig. 15(b), this is observed for all network sizes in both application settings. Likewise, overhead in the form of number of packets sent stays almost constant as σ increases (Fig. 15(d)). Thus, SCALAR' s performance is not adversely affected in the cases where majority of the data in the system is popular and requested with a high probability (in comparison to a smaller portion of data not requested very frequently).

Table 5
Success ratio per node: Standard deviations.

Network size	20	50	100	200	400
Standard deviation	16.82	14.98	17.82	18.88	17.92

6.5. Cost function analysis and success ratio per node

In order to evaluate the benefit of the cost function during the replication decision, we compared it with the case where random replacement is performed when the cache of a node is full. That is, instead of using cost function and comparing the cost of newly received data item with the minimum cost replica in the cache, random replacement picks a data item in the cache randomly for replacement with the new item. We have run simulations for different number of data requests per node (r). Fig. 16 show results for different network sizes and corresponding average success ratios obtained using cost function based replacement and the random replacement. As the number of data requests per node (r) gets larger, benefit of using cost function becomes more pronounced. For example, as shown in Fig. 16(c) for $r = 30$, success ratio is about 8–10% better when using cost function in comparison to random replacement.

We also investigated the individual success ratios per node. Fig. 17 shows sample results for runs on 100, 200 and 400-node networks. Success ratio values are stable among the nodes, and also standard deviations of the success ratio values computed over all nodes for different network sizes are very similar. Thus, success ratio variation over the network is not affected by the network size as shown in Table 5.

7. Conclusions

We proposed a scalable data lookup and reactive replication (SCALAR) protocol for MANETs. It is a low-cost solution in terms of message overhead so that it can be easily adapted to large scale network scenarios. On the other hand, it is as successful as other high-cost lookup solutions when searching the requested data in the network. SCALAR consists of three main parts: (1) virtual backbone construction, (2) scalable data lookup protocol and (3) reactive replication approach. We compared the performance of SCALAR with an AODV-based data lookup process as well as well-known replication protocols in different application scenarios. SCALAR outperforms the other approaches and it can perform quite well in very high node density networks. On the other hand, other replication and data lookup approaches failed to keep their performances at a certain level as the number of nodes and node density in the network increase. We observed that these performance losses are due to exponentially increasing message traffic caused by increasing number of nodes or density.

Our reactive replication approach does not expose any explicit control messages to the system and gives replication decisions when a new data item is received by a backbone node. Replication decision is based on the local access request statistics of received data item and distance of it to the requester node. We can conclude that connected dominating set based backbone structure of SCALAR, combined with the scalable data lookup protocol can keep the message overhead of the proposed protocol at very low levels. In SCALAR, we developed a lightweight lookup and replication approach, which can adapt equally well to large scale or high density network conditions. As future work, effect of keeping history of requests to the same data item on improving data availability can be investigated. Furthermore, mathematical modeling of the protocol and network conditions can be developed in order to optimize specific node or network parameters for better performance.

Acknowledgment

This study is supported by TUBITAK (The Scientific and Technical Research Council of Turkey) under CAREER Award Grant 104E064. A preliminary version of this work appeared in [38].

References

[1] L. Yin, G. Cao, Supporting cooperative caching in ad hoc networks, IEEE Transactions on Mobile Computing 5 (1) (2006) 77–89.
[2] W. Lau, M. Kumar, S. Venkatesh, A cooperative cache architecture in support of caching multimedia objects in MANETs, in: Proceedings of the 5th ACM International Workshop on Wireless Mobile Multimedia, 2002, pp. 56–63.
[3] B. Tang, H. Gupta, S. Das, Benefit-based data caching in ad hoc networks, network protocols, 2006, in: 14th IEEE International Conference on Proceedings of the 2006, ICNP'06, 2006, pp. 208–217.
[4] Y. Du, S. Gupta, COOP-a cooperative caching service in MANETs, in: Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005, ICAS–ICNS, 2005, pp. 58.
[5] T. Hara, S.K. Madria, Data replication for improving data accessibility in ad hoc networks, IEEE Transactions on Mobile Computing 5 (11) (2006) 1515–1532.
[6] A. Mondal, S. Madria, M. Kitsuregawa, CADRE: a collaborative replica allocation and deallocation approach for mobile-P2P networks, in: 10th International Database Engineering and Applications Symposium, 2006, IDEAS'06, 2006, pp. 21–28.
[7] A. Mondal, S. Madria, M. Kitsuregawa, CLEAR: an efficient context and location-based dynamic replication scheme for mobile-p2p networks, in: Proc. of International Conference on Database and Expert Systems Applications. DEXA, 2006.
[8] P. Pabmanabhan, L. Gruenwald, DREAM: a data replication technique for real-time mobile ad-hoc network databases, in: Proceedings of the 22nd International Conference on Data Engineering, 2006, ICDE'06, 2006, pp. 134.
[9] T. Hara, S.K. Madria, Consistency management strategies for data replication in mobile ad hoc networks, IEEE Transactions on Mobile Computing 8 (7) (2009).
[10] X. Zhang, G. Riley, Scalability of an ad hoc on-demand routing protocol in very large-scale mobile wireless networks, Simulation 82 (2) (2006) 131–142.
[11] V. Gianuzzi, Data replication effectiveness in mobile ad-hoc networks, in: Proceedings of the 1st ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, 2004, pp. 17–22.
[12] A. Derhab, N. Badache, Data replication protocols for mobile ad-hoc networks: a survey and taxonomy, IEEE Communications Surveys and Tutorials Journal 11 (2) (2009).
[13] P. Padmanabhan, L. Gruenwald, A. Vallur, M. Atiquzzaman, A survey of data replication techniques for mobile ad-hoc network databases, Journal of Very Large Data Bases (2006).
[14] K. Chen, S. Shah, K. Nahrstedt, Cross-layer design for data accessibility in mobile ad hoc networks, Wireless Personal Communications 21 (1) (2002) 49–76.
[15] H. Yu, P. Martin, H. Hassanein, Cluster-based replication for large-scale mobile ad-hoc networks, in: International Conference on Wireless Networks, Communications and Mobile Computing, vol. 1, 2005.
[16] C.-A. La, P. Michiardi, C. Casetti, C.-F. Chiasserini, M. Fiore, A lightweight distributed solution to content replication in mobile networks, in: WCNC, 2010.
[17] Z. Torbey, N. Bennani, L. Brunie, D. Coquil, Performance evaluation for cream: User-centric replication model, in: 11th Annual International Conference on New Technologies of Distributed Systems (NOTERE), 2011, pp. 1–8.
[18] M. Fiore, C. Casetti, C.-F. Chiasserini, Caching strategies based on information density estimation in wireless ad hoc networks, IEEE Transactions on Vehicular Technology 60 (5) (2011).
[19] D. Hirsch, S. Madria, A resource-efficient adaptive caching scheme for mobile ad hoc networks, in: Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems, SRDS, 2010.
[20] J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1998, pp. 85–97.
[21] Y. Tseng, S. Ni, Y. Chen, J. Sheu, The broadcast storm problem in a mobile ad hoc network, Wireless Networks 8 (2) (2002) 153–167.
[22] B. Williams, T. Camp, Comparison of broadcasting techniques for mobile ad hoc networks, MobiHoc '02, in: Proc. of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2002, pp. 194–205.
[23] D.C.A.S.Y. Sasson, Probabilistic broadcast for flooding in wireless mobile ad hoc networks, in: IEEE Wireless Communications and Networking Conference (WCNC), 2003.

[24] J.H.Z. Haas, L. Li, Gossip-based ad hoc routing, IEEE/ACM Transactions on Networking 14 (3) (2006).
[25] M. Min, F. Wang, D. Du, P. Pardalos, A reliable virtual backbone scheme in mobile ad-hoc networks, in: 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), 2004.
[26] J. Wu, F. Dai, Virtual backbone construction in MANETs using adjustable transmission ranges, IEEE Transactions on Mobile Computing 5 (9) (2006) 1188–1200.
[27] C. Chiang, H. Wu, W. Liu, M. Gerla, Routing in clustered multihop, mobile wireless networks with fading channel, in: Proceedings of IEEE SICON, vol. 97, 1997, pp. 197–211.
[28] P. Sinha, R. Sivakumar, V. Bharghavan, Enhancing ad hoc routing with dynamic virtual infrastructures, INFOCOM 2001. In: Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3, 2001.
[29] L. Ritchie, H. Yang, A. Richa, M. Reisslein, Cluster overlay broadcast (COB): MANET routing with complexity polynomial in source-destination distance, IEEE Transactions on Mobile Computing 5 (6) (2006) 653–667.
[30] H. Yang, L. Ritchie, A. Richa, M. Reisslein, MANET routing with provably low complexity through constant density clustering and route request broadcast, Wireless Personal Communications 43 (2) (2007) 605–621.
[31] S. Guha, Approximation algorithms for connected dominating sets, Algorithmica 20 (4) (1998) 374–387.
[32] J. Wu, H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, in: Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing And Communications, 1999, pp. 7–14.
[33] S. Das, C. Perkins, E. Royer, Ad hoc on Demand Distance Vector (AODV) Routing, Mobile Ad-hoc Network (MANET) Working Group, IETF, vol. 81, January 2002.
[34] D. Johnson, D. Maltz, Y. Hu, et al., The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR), IETF MANET Working Group (Draft 10), 2004.
[35] R. Barr, Java in Simulation Time (JiST)/Scalable Wireless Ad hoc Network Simulator (SWANS), 2004.
[36] E. Atsan, Ö. Özkasap, A classification and performance comparison of mobility models for ad hoc networks, in: T. Kunz, S.S. Ravi (Eds.), ADHOC-NOW, Ser. Lecture Notes in Computer Science, vol. 4104, Springer, 2006, pp. 444–457.
[37] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wireless Communications & Mobile Computing 2 (2002) 483–502.
[38] E. Atsan, Ö. Özkasap, SCALAR: scalable data lookup and replication framework for mobile ad-hoc networks, in: Workshop on Wireless Ad hoc and Sensor Networks (WWASN), International Conference on Distributed Computing Systems (ICDCS), China, 2008.

Emre Atsan received the B.S. degree in Computer Engineering and and M.S. degree in Electrical & Computer Engineering from Koc University, in 2005 and 2007, respectively. He is currently a PhD student at EPFL. His research interests include mobile ad hoc networks and distributed systems.

Oznur Ozkasap received the M.S. and Ph.D. degrees in computer engineering from Ege University in 1994 and 2000, respectively. From 1997 to 1999, she was a graduate research assistant at Cornell University, Department of Computer Science, where she completed her Ph.D. dissertation. She is currently an associate professor in the Department of Computer Engineering at Koc University, which she joined in 2000. Her research interests include distributed computing systems, multicast protocols, peer-to-peer systems, bio-inspired distributed algorithms and computer networks.