SCALAR: Scalable Data Lookup and Replication Framework for Mobile Ad-hoc Networks *

Emre Atsan
Koç University
Computer Engineering Department
34450 Istanbul, Turkey
eatsan@gmail.com

Öznur Özkasap
Koç University
Computer Engineering Department
34450 Istanbul, Turkey
oozkasap@ku.edu.tr

Abstract

Data replication protocols proposed for MANETs are often prone to the 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. Extensive simulations are performed to analyze and compare the behavior of this framework. It is 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 devices (nodes). MANETs are adaptive networks, which are reconstructed in the case of network changes due to mobility. In MANET research, most of the effort has focused on the creation 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 availability, 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, due to limited resources or power considerations of network nodes, it is important to define good replication criteria that can select the most appropriate data items and best hosts for replication.

Possible solutions to replication problem can be very complex if the size of the network increases. Existing protocols for data lookup and replication in MANETs do not consider large scale networks. In [6], Hara et al. proposes three replica allocation schemes, 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 [7], [8], [9] also do not address the scalability problem by definition.

Our motivation in this work is that a scalable and efficient solution for data lookup and replication is necessary. For this purpose, we propose a SCAlable data Lookup And Replication framework, called *SCALAR*, which does not depend on the existence of an underlying MANET routing protocol, that may be prone to scalability problems [12]. In order to minimize the number of nodes involved in the data lookup process, SCALAR constructs a dynamic virtual backbone structure among the mobile nodes. The data lookup protocol takes advantage of this virtual backbone for low-cost data requests. Lastly, we extend the data lookup protocol with a data replication approach, which aims to replicate frequently accessed data items from distant places.

*This work is supported by TUBITAK (The Scientific and Technical Research Council of Turkey) under CAREER Award Grant 104E064.

SCALAR is a complete solution for large-scale, low-speed and highly connected ad hoc networks, in which disseminating request packets does not require an underlying routing protocol. The rest of this paper is organized as follows: Section 2 presents our SCALAR framework. Simulation results and analysis of them are given in Section 3. Finally, Section 4 concludes the paper and gives future directions.

2 SCALAR

SCALAR framework is composed of a *virtual backbone construction algorithm*, a *scalable data lookup protocol*, and a *reactive replication scheme*. Our solution is scalable since it does not create too much message overhead to the network with the increasing number of nodes, and it performs well in means of data accessability. The idea is 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 (CDS) construction problem in graph theory and proposed for ad hoc wireless networks [11]. 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.

2.1 System Model

System environment is assumed to be a meso-scale to large-scale mobile ad hoc network. We assign a unique host identifier to each node in the system. The set of all nodes in the system is denoted as $M = \{M_1, M_2, ..., M_N\}$, 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 $D = \{d_1, d_2, ..., d_N\}$. For the sake of simplicity, we assumed that all data items are of the same size and can be put into one network layer packet, i.e. *data size* $<$ *path MTU (maximum transmission unit)*. 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, in the network and can request every data item, d_j at any time. We assume an environment where every data item is unchangeable, so that *updates* to data items and related consistency problems are not considered.

2.2 Virtual Backbone Construction

MANETs assume wireless communication between mobile nodes without any physical infrastructure support. However, this infrastructureless communication causes increased 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 [3]. Such flooding/broadcasting packets in the network cause the creation of excessive redundant packets (*broadcast storm problem* [10]) and collision/contention problem in wireless channel. In large scale MANETs, problems become more pronounced [12]. To maximize the utilization of incapacitated node resources and to minimize drawbacks caused by flooding, many researchers proposed virtual backbone (or *spine*) structures which are inspired by physical internet backbones. 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. In this work, we utilize a simple and easily maintainable CDS construction algorithm [11] as the virtual backbone construction mechanism. For the construction of a virtual backbone, a CDS of the unit-disk graph of a corresponding network topology is used.

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*. A connected dominating set (CDS) is a dominating set whose induced subgraph is connected. Finding a minimum size dominating set of G is proven to be NP-Complete by a reduction from the *vertex cover* problem [5].

CDS construction algorithm Basically, the algorithm [11] first finds an initial CDS, *I*, and then prune certain redundant nodes from this CDS. *I* consists of all nodes which have at least two non-adjacent neighbors. Then all nodes in *I*, which has either a neighbor in *I* with larger ID which dominates all other neighbors of this node, or two adjacent neighbors with larger IDs which together dominate all other neighbors of this node, are dropped from this initial CDS, *I*.

Our distributed implementation of above algorithm is as follows: Each node first broadcasts its neighbor information to all of its one-hop neighbors, and after receiving the same information from all 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 above algorithm applies. Message complexity of this implementation is $\Theta(n)$, where n is the number of nodes in the network.

2.3 Scalable Data Lookup Protocol

Scalable data lookup protocol searches a data item in the entire system by sending the request to the virtual backbone members, to which every other node is directly connected. Requests are only forwarded between the backbone nodes, and as a result the number of nodes involved in the lookup process kept limited.

A straightforward solution to the data lookup problem in ad hoc networks is *flooding* the request to the entire network. Another solution may be 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 [4]). Both of these solutions are shown to result in serious contention and collision in large scale wireless networks [10], [12].

Node Types There are two types of nodes in the network classified based on their roles: *(a)* backbone (dominator) node, and *(b)* end system (dominatee). 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 (*request injection*).

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 are satisfied: (a) d_j is owned by the backbone node M_i. (b) 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 (or 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.

End systems 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 entire connected 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, if it exists. Similar to the backbone nodes, if a requested data item exists in an 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 needs to save 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>*. This data structure will then be used to route the received data packet to the destination. 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.

Data Receive A backbone node participates in the data receive process *(a)* if it receives a request for the data item, which it owns or *(b)* if it receives a data packet from a backbone or end system node for which a data request is forwarded during the searching phase or finally *(c)* if it is the originator of the request.

In the case (a), 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 ,dest.of packet, req.packet id>*. If case (b) holds, backbone node checks its requested items data structure (a 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 (c) 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.

In end systems, every received data should be this node's request. Multiple copy reception for one data is not possible because it is handled by the backbone nodes before coming to end systems. In SCALAR, obviously, backbone nodes consume much more energy and bandwidth to support the requests and data receives 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.

2.4 Reactive Replication

Reactive Replication (RR) is constructed on top of proposed scalable data lookup protocol. It is *reactive* in the sense that making replication decisions and replicating data items is completed after a data packet is received. It does not require the exchange of explicit replication control packets. Thus, *RR* completely eliminates the control overhead. Replication decision of data items is based on *request frequency* of an item and the *distance* of the received data's owner. *RR* aims to replicate the distant data items in order to decrease the number of requests propagated in the virtual backbone. Besides the hop count, proposed replication approach also considers the request frequency of items during its replica allocation decision. *RR* regulates the local caches of the nodes so that costly requests are cached preferably. Cost of a request to a requester node is calculated using a common function for all node types as described next.

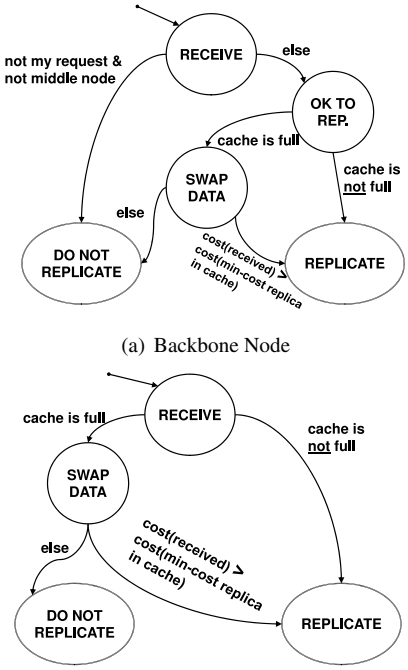
Cost Function The cost function aims to minimize the cost of the data request to the system based on the previous request frequency information and distance to the owner of a data item. Closer the requested data items are to the nodes, lower the message complexity of our lookup protocol is. *RR* aims to replicate frequently accessed distant data by using the cost function at node M_j for data item d_i:

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

where α_i is the local request *frequency* of data item d_i until that time 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 that have higher probability of being the next packet requested and that are distant to the requester node.

Replication Decision A backbone node gives a data replication decision if one of the following cases is true, as illustrated in Figure 1(a) with the replication decision tree. In either case, backbone nodes have a tendency to replicate data items with higher cost more.

- 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 lowest-cost item is replaced with the newly received data item. If cache space has still vacant positions for a new item, then node does not need to make a cost comparison before adding the data item to its cache space.
- if the data item is received due to a forwarded request, than receiving 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 mid node in the path, it may decide to replicate the data item. Decision of replication is again based on the costs of received data item and lowest-cost item in a full memory. If cache space has still a vacant position, then node caches received data without questioning.



(a) Backbone Node

(b) End System

Figure 1. Replication decision trees.

End system's replication decisions are similar to the ones for backbones as illustrated in Figure 1(b). However, since end systems are not allowed to forward data, they are not supposed to give replication decisions about being on the mid point of a data path, like backbone nodes do.

3 Results and Analysis

Network simulations have been performed using SWANS (Scalable Wireless Ad hoc Network Simulation) tool [2]. SWANS is built on Java-based simulation framework JiST (Java in Simulation Time). Its capabilities are similar to *ns-2* or *GloMoSim*, morever it is able to simulate much larger networks, thanks to JiST's high performance structure [2]. 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 sim-

ulation, 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 given 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 set to Random Waypoint mobility model, 802.11b MAC protocol, 100m radio range, 1 to 3 m/s node speed, IP network and UDP transport protocols.

We compare our results with a simple data request 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 and reactive property, 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. Our performance metrics are (a) *success ratio*,(b) *average query deepness*, (c) *packets sent per node*. Success ratio is the ratio of the number of successful access requests to the number of all access requests issued. The average number of nodes (or hops) traversed by a successful query when finding the requested data is called as average query deepness. In our simulations, every node can save up to 5 data items, and each simulation runs for 300 seconds.

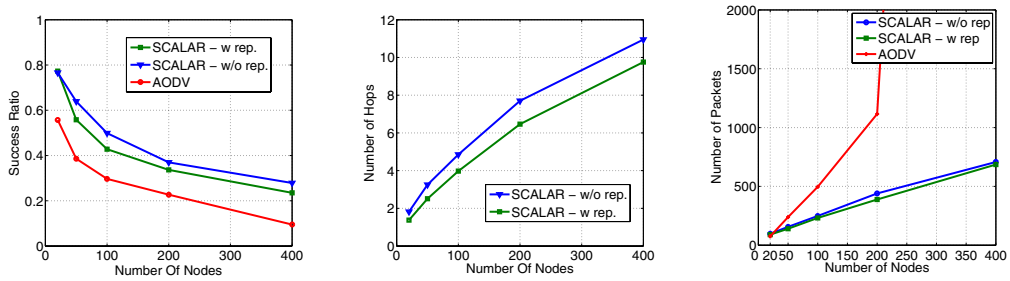
Figure 2(a) shows that as the number of nodes in the network scales up, in every data lookup approach the success ratio decreases. This can explained by the increase in the average query deepness as the number of nodes increases (see Figure 2(b)). As the average query deepness increases, the number of nodes involved in the transmission of a data request increases. We know that 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 probably will cost as 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. To understand why AODV based data request performs worse, we need to check results in Figure 2(c), which the average number of packets sent per node is given. It is obvious that as the number of nodes increases in the network, message overhead introduced by AODV routing protocol 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.

In Figure 3(a), SCALAR achieves considerable increase in the success ratio as density increases. However, AODV based data lookup scheme does not scale to the increasing densities in terms of data accessibility. This is due to the high message overhead introduced in AODV with the 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 Figure 3(c) reveals that *SCALAR* tends to send fewer packets per node when the density of our network increases over a threshold density value (which 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, Figure 3(b) shows that 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 decreasing the delay and message cost of the system.

We also investigate the effects of varying data requests per node which is called *network load*. It is observed that in lower network loads, AODV performs better. 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. In terms of *fairness* in load balancing, SCALAR have some known deficiencies. For example, nodes with larger IDs have a higher probability of becoming a backbone node, since the virtual backbone construction algorithm used is tend to prune the backbone nodes with smaller IDs. Also, nodes which are in center of a network have more responsibility and load [1]. We have also investigated the *effects of different mobility models* on the replication performance. Due to page limitation of the current paper, we refer the interested reader to [1] for details of these results.

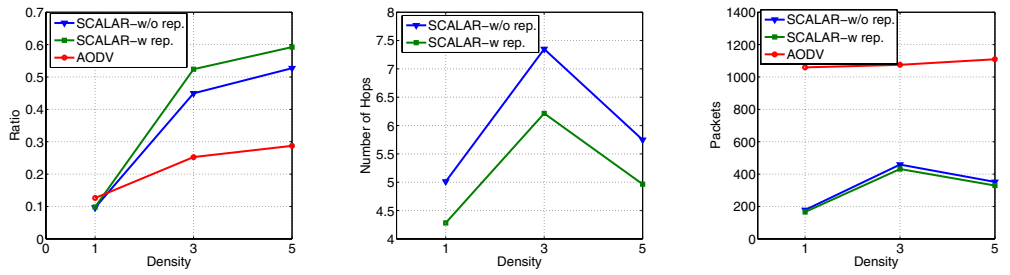
4 Conclusions

We presented a scalable data lookup and reactive replication (SCALAR) framework 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: virtual backbone construction, scalable data lookup protocol and reactive replication approach. We compared the performance of SCALAR with AODV-based data lookup process. It is shown that, even in small networks, SCALAR outperformed in each of the performance metric defined. Moreover, SCALAR can perform quite well in very high node



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

Figure 2. Impact of increasing number of nodes. Simulation area size is calculated based on density value 2. Data request per node is 10 during the entire simulation.



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

Figure 3. Impact of increasing node density. Simulation area size is calculated based on constant 200 nodes in simulation. Data request per node is 10 during the entire simulation.

density networks. We observed that performance loss in the AODV approach is due to the exponentially increasing message traffic caused by increasing number of nodes. As further work, mathematical modeling of the protocol and network conditions can be developed in order to optimize specific node or network parameters for better performance. Another future work might be analyzing the effects of other CDS construction algorithms to SCALAR's performance.

References

[1] E. Atsan. A Scalable and Reactive Replication Framework For Mobile Ad-hoc Networks. MSc Thesis, Koc University, September 2007.

[2] R. Barr. Java in Simulation Time (JiST)/Scalable Wireless Ad hoc Network Simulator (SWANS), 2004.

[3] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A Performance Comparison of Multi-hop Wireless Ad hoc Network Routing Protocols. *Proc. of MOBICOM*, 1998.

[4] S. Das, C. Perkins, and E. Royer. Ad hoc On Demand Distance Vector (AODV) routing. *Mobile Ad-hoc Network (MANET) Working Group, IETF, Jan*, 81, 2002.

[5] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman & Co. New York, NY, USA, 1979.

[6] T. Hara and S. K. Madria. Data Replication for Improving Data Accessibility in Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 5(11):1515–1532, 2006.

[7] A. Mondal, S. Madria, and M. Kitsuregawa. CADRE: A Collaborative Replica Allocation and Deallocation Approach for Mobile-P2P Networks. *Proc. of IDEAS*, 2006.

[8] A. Mondal, S. Madria, and M. Kitsuregawa. CLEAR: An Efficient QoS-based Dynamic Replication Scheme for Mobile P2P Networks. *Proc. of DEXA*, 2006.

[9] P. Pabmanabhan and L. Gruenwald. DREAM: A Data Replication Technique for Real-Time Mobile Ad-hoc Network Databases. *Proc. of ICDE*, pages 134–134, 2006.

[10] Y. Tseng, S. Ni, Y. Chen, and J. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. *Wireless Networks*, 8(2):153–167, 2002.

[11] J. Wu and H. Li. On Calculating Connected Dominating Set for Efficient Routing in Ad hoc Wireless Networks. *Proc. of DIALM*, pages 7–14, 1999.

[12] X. Zhang and G. Riley. Scalability of an Ad Hoc On-Demand Routing Protocol in Very Large-Scale Mobile Wireless Networks. *Simulation*, 82(2):131–142, 2006.