A Scalable and Reactive Replication Framework For Mobile Ad-hoc Networks

by

Emre Atsan

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Electrical & Computer Engineering

Koç University

September, 2007

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Emre Atsan

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Assist. Prof. Öznur Özkasap (Advisor)

Assist. Prof. Sibel Salman

Assist. Prof. Serdar Taşıran

Date: _____

To my parents and sister
Anneme, babama ve Esra'ya...

ABSTRACT

Scalability is an important criterion which is not taken into consideration in the design of most Mobile Ad hoc Network (MANET) protocols. However, requirements of a scalable protocol are different from the ones for small or meso-scale networks. Therefore, MANET protocols exhibit poor performance when the network size is scaled up. Novel approaches that consider the requirements and performance bottlenecks of large-scale networks are needed to achieve better performance.

Data replication, as one of the popular services in 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. Prior replication protocols proposed in the literature for MANETs are prone to the scalability problems due to their definitions and/or underlying routing protocols they are based on. In this thesis, we propose a scalable and reactive data replication framework (SCALAR) combined with a low-cost data lookup protocol. It is a virtual backbone based solution, in which the network nodes construct a connected dominating set by considering the connectivity information. Theoretical message-complexity analysis of the proposed protocols is given. SCALAR is implemented on JiST/SWANS network simulator. Extensive simulations are performed to analyze and compare the behavior of SCALAR for varying parameters such as size of the network, node densities, memory space of nodes and request rate. SCALAR is shown to outperform the compared protocols in terms of data accessibility, message overhead and query deepness. Furthermore, it is demonstrated as an efficient solution for high-density, high-load ad hoc networks.

ÖZETÇE

Ölçeklenirlik, gezici amaca-yönelik ağ protokollerinin bir çoğunun tasarımında dikkate alınmayan önemli bir kriterdir. Fakat, büyük ölçekli bir ağın gereksinimleri küçük ve orta ölçekli ağlarınkinden farklıdır. Bu yüzden, gezici amaca-yönelik ağ protokolleri ağ boyutu büyüdüğü zaman daha düşük başarım sergilemektedirler. Büyük ölçekli ağlardaki gereksinimleri ve başarım engellerini dikkate alan yeni yaklaşımların daha iyi başarım sonuçları vermesi beklenir.

Gezici amaca-yönelik ağlarda veri kopyalama, sık kullanılan verilerin yerel kopyalarını oluşturmak yoluyla veri erişilebilirliğini arttırmak, iletişim masraflarını düşürmek, ve hata hoşgörüsü ile yük dengelemesini başarmak için kullanılır. Literatürde gezici amaca-yönelik ağlar için önerilen mevcut kopyalama teknikleri, tanımları veya kullandıkları yöneltme protokolleri nedeniyle ölçeklenirlik problemlerine açıktırlar. Bu tezde, düşük maliyetli veri arama protokolü ile birleştirilmiş, ölçeklenebilir bir veri kopyalama sistemini (SCALAR) öneriyoruz. Bu sistem, düğümler arası bağlantı bilgisinin temel alındığı, ağ düğümlerinin bağlantılı bir baskın küme (connected dominating set) oluşturduğu, sanal omurga tabanlı bir sistemdir. Önerilen protokolün teorik olarak karmaşıklık analizi ayrıca verilmiştir. SCALAR sistemi, JIST/SWANS ağ benzeticisi kullanılarak uygulanmıştır. Geliştirilen sistemin ağ boyutu, düğüm yoğunluğu, düğümlerin bellek alanı kapasitesi ve istek oranı gibi değişkenler karşısındaki davranışını karşılaştırmak ve analiz etmek için geniş kapsamlı benzetimler gerçekleştirilmiştir. SCALAR veri erişilebilirliği, mesaj yükü ve sorgu derinliği bakımından karşılaştırılan diğer protokolerden daha iyi sonuçlar vermiştir. Ayrıca yoğun ve yüklü amaca-yönelik ağlar için de etkin bir çözüm olduğu gösterilmiştir.

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor Assist. Prof. Öznur Özkasap for her valuable guidance, understanding and patience throughout my graduate study. Also, I would like to thank Assist. Prof. Sibel Salman and Assist. Prof. Serdar Taşıran for their valuable comments as members of my thesis committee.

I am grateful to my parents, Havva and Mehmet Emin and my sister Esra for their endless support, patience and love during my entire life.

I am also grateful to Selen for her valuable support and patience and being a great friend.

I was fortunate to have such great friends close to me during my graduate education. Burak Görkemli, Ekin Akkuş, Göktuğ Gürler were great officemates. It was a pleasure to share the same house with Burak Çavdaroğlu and Can Yıldırmaz. And also I would like to thank my friends and colleagues; Mehmet Ali, Erkan, Emre Güney, Ergun, Tuğba, Çağdaş and Tayfun.

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

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

MANET	Mobile Ad hoc Network
VANET	Vehicular Ad hoc Network
PDA	Personal Digital Assistant
DS	Dominating Set
CDS	Connected Dominating Set
WCDS	Weakly Connected Dominating Set
AODV	Ad hoc On Demand Distance Vector
DSR	Dynamic Source Routing
UDG	Unit Disk Graph
MIS	Maximal Independent Set
MAC	Media Access Control
IP	Internet Protocol
UDP	User Datagram Protocol
SAF	Static Access Frequency
DAFN	Dynamic Access Frequency and Neighborhood
DCG	Dynamic Connectivity based Grouping
MTU	Maximum Transfer Unit
IEEE	Institute of Electrical and Electronics Engineers
TTL	Time-to-Live
CH	Cluster Head
GN	Gateway Node
EVC	Eigenvector Centrality

Chapter 1

INTRODUCTION

Everyday, wireless devices, such as Bluetooth-enabled cell phones, PDAs, and laptops become more popular and indispensable for several computer applications. Wireless Local Area Networks are a necessity in our daily lives due to the increasing number of personal wireless electronic devices. However, in some situations, it turns out to be impossible or too costly to deploy permanent infrastructures (i.e. wireless routers, satellite links, cellular systems) for a wireless network, and networking the mobile or static nodes with wireless links in an ad hoc manner can be necessary and effective.

The idea behind networking wireless nodes in an ad hoc manner dates back to the DARPA packet radio network research [1]. During 1970s, the ALOHA project used broadcasting nature of radio signals to send and receive data packets in single hop domain. Later, multi-hop communications over packet radio network (PRNET) started in Advanced Research Project Agency (ARPA) [2], which would form the basis of ad hoc networks. In 1990s, Institute of Electrical and Electronic Engineering (IEEE) renamed PRNET to *ad hoc* network. Consequently, the Mobile Ad hoc Networking (MANET) working group was formed within the Internet Engineering Task Force (IETF) to standardize the necessary protocols and specifications.

A Mobile Ad Hoc Network (MANET) is a self-organizing, infrastructureless, dynamic wireless network of autonomous mobile devices (nodes). The network is *ad-hoc*, because there is no fixed and known network structure that every other node forwards data. Thus, the decision of forwarding among nodes is made dynamically based on the network connectivity. In a MANET, each node is not only an end system, but

also a router in the network's multi-hop communication structure. Laptop computers, personal digital assistants (PDAs) and mobile phones can construct mobile ad-hoc networks. MANETs are adaptive networks, which means they are reconstructed in the case of network changes due to mobility. Generally, nodes forming a MANET are battery powered devices and have energy and bandwidth constraints [3]. Broadcasting and multicasting in MANETs should be carefully implemented due to possible flooding of unnecessary information. Since devices are battery powered, inefficiency of communication protocols can shorten the active lifetime of these devices. Possible application areas of MANET network structure are military communication systems, personal area networks (PAN), wireless peer-to-peer networks. Furthermore, Vehicular Ad-Hoc Networks (VANETs), which aim to create a dynamic communication structure among vehicles, become a promising research field for MANETs.

In mobile ad hoc networking research, most of the effort has focused on the creation of dynamic routing protocols that aim to find routes between two mobile nodes efficiently. Although routing is a critical issue in MANETs, accessing remote data is an equally important topic since the ultimate goal of an ad hoc network structure is to provide the necessary data items to requester mobile nodes. Different than static, infrastructure-based conventional networks, locating the remote data (*data lookup*) and accessing is a challenging problem. It is because in this case mobile users need to learn the availability of data in an ad hoc manner without the help of any central server. Moreover, due to the unpredictable mobility behaviors of nodes, rapid changes in the MANET topology is presumable. These changes can result in partitioning, dividing the network into isolated subnetworks. Thus, data availability in MANETs is lower than fixed networks. One possible solution to this problem can be *replication* of popular data items at selected places 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 nearby can solve most of the communication latency related problems. On the other side, 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 network 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. Replication of data items to avoid data losses in case of a network partitioning is a promising solution for low-density, highly partitionable mobile networks [4]. Furthermore, by increasing the availability of data items, replication also decreases the cost of data lookup in a MANET.

Another possible solution to the problem of low data availability (accessibility) in mobile networks is *cooperative caching* techniques [5][6][7][8]. 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 delay times. Caching reduces communication cost of the system, which consequently results in the reduction of bandwidth and energy usage. Note that, caching techniques are developed as middleware layer between a routing protocol and an application layer protocol. It helps the lower layer communication protocols for finding an efficient route to the owners of the data item, which is requested by an application layer protocol. It does not try to obtain data items to replicate at specific locations, like a data replication approach does. It gives a caching decision based on the data on-the-fly. In general, caching approaches try 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[5]. Furthermore, caching and replication schemes helps to establish a load balancing among nodes.

As we can see from the above discussion that accessing a specific data in MANETs is an important problem and solutions to this problem can give better results when they are supported with a replication or caching approach. On the other hand, 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 mobile ad hoc settings. Most of the proposed techniques for data lookup and replication

solutions in MANETs do not consider scalability. Because of this, these solutions can give unexpected results in a large scale network 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 [9]. In some cases, this performance loss in large scale networks is due to proposed solution's large control message overhead. In small or mesoscale networks the message overhead may not be detrimental, but it effects the performance of the solution in large-scale scenarios.

Our motivation in this thesis is that a scalable solution for data lookup and replication is necessary. For this purpose, we constructed a **SCA**lable data **L**ookup **A**nd **R**eplication framework, called *SCALAR*. Our proposed system does not depend on a MANET routing algorithm, which may be prone to scalability problems. To the best of our knowledge, there is no data lookup and replication approach proposed specifically for large scale ad hoc networks so far in the literature.

SCALAR offers a scalable solution for data lookup and replication in MANETs. It consists of three parts: virtual backbone construction, data lookup and reactive replication. 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, which is based on an approximation of minimum connected dominating set problem in graph theory. Proposed data lookup protocol takes advantage of the dominating set behavior of constructed virtual backbone for low-cost data requests. Lastly, we extend this data lookup protocol with a data replication approach, which aims to replicate frequently accessed data items from far away places. 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.

Simulation results show that our proposed system outperforms the straightforward solutions for data lookup in MANETs when the scale of the network is large. Also, it is shown 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.

1.1 Contributions

Contributions of this study are the following:

1. We propose a novel virtual backbone based, scalable data lookup and reactive replication framework (*SCALAR*) for mobile ad hoc networks. SCALAR is composed of three main parts:

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

   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.

2. We give a worst-case message complexity analysis of each part in SCALAR framework.

3. A fully distributed implementation of SCALAR system in *Jist/SWANS* [10] network simulator is completed.

4. Extensive simulations are performed to analyze and compare the behavior of SCALAR for varying parameters such as size of the network, node densities, memory space of nodes and request rate. Two representative MANET application scenarios are also investigated. SCALAR is shown to outperform the

compared protocols in terms of data accessibility, message overhead and query deepness. Furthermore, it is demonstrated as an efficient solution for high-density, high-load ad hoc networks.

5. Comparative analysis is performed with the AODV [11] protocol for lookup, SAF and DAFN [12] algorithms for replication. For this purpose, implementations of SAF and DAFN, which are not publicly available, are contributed to the Jist/SWANS simulator.

6. Implementation of Random Walk mobility model is added to *Jist/SWANS* network simulator in order to investigate the effect of mobility model used to the performance of SCALAR.

1.2 Organization

Rest of this thesis is organized as follows: Chapter 2 gives a literature survey on existing data replication, caching, and connected dominating set (CDS) construction approaches. Chapter 3 presents SCALAR. We explain our simulation system model, simulation scenario descriptions and network simulator tool in Chapter 4. In Chapter 5, simulation results and analysis of them are given. Finally, Chapter 6 concludes this thesis and gives future directions.

Chapter 2

RELATED WORKS

To improve the data availability and access efficiency in MANETs, data replication and caching techniques can be used as discussed in Section 1. In this chapter, existing studies on data replication and cooperative caching techniques for MANETs will be investigated in Sections 2.1 and 2.2, respectively. In Section 2.3, we demonstrate a feature comparison of each replication and caching approach discussed in Sections 2.1 and 2.2, respectively. Finally, in Section 2.4, we give a survey on connected dominating set construction algorithms in the literature.

2.1 Data Replication Techniques for MANETs

Data replication is used to avoid data losses in case of unpredictable disconnections of mobile nodes by increasing system wide data availability [4]. Also, replication increases the efficiency by decreasing the number of hops that a data item is transmitted from source to destination. This section reviews the replication techniques developed for MANETs. A detailed survey on MANET replication techniques and a classification and comparison of them can be found in [13].

2.1.1 Hara's Data Replication Schemes

One of the most well-known schemes for data replication in MANETs are presented in [12]. It proposes three replica allocation methods; and each method has been studied for both update-less and periodically updated systems. In these replica allocation methods, the main parameter for replication decision is the *access frequencies* of data items. These techniques make the following assumptions:

- every mobile host has a limited memory space to replicate data items.

- the *access frequency* of every data item is known at the beginning and does not change.

For network environments without any update mechanism for data items, following three replica allocation methods are proposed:

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. Basically, each mobile host arranges data items in the descending order of their access frequencies and selects the top x nodes from this arrangement to replicate, where x represents the number of available slots for replication in the node's memory.

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 we use the SAF method for replication, than 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 (Dynamic Access Frequency and Neighborhood) 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. If one of the neighbor nodes is the *owner* of the data item, then the other node changes its replicated data item. In DAFN method, both message traffic and processing overhead is higher than the SAF method. However, 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 (Dynamic Connectivity based Grouping) method shares replicas among larger and stable groups of nodes. DCG method creates *biconnected* components of a network. Every biconnected 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. After creating biconnected components, every group determines the group access frequency of a data item. In the order of group access frequencies, every data item is replicated until the memory space of all mobile hosts in the group is used. Using DCG, many different kinds of replicas can be shared among the biconnected components of the network. So, we can expect better data availability. On the other side, both processing overhead and message traffic are higher than SAF and DAFN. This is due to exchange of more information in each allocation period by mobile nodes.

In [12], for network environments with data updates, previous three replica allocation methods are modified to handle the effects of updates:

1. Extended SAF plus (E-SAF+)

2. Extended DAFN plus (E-DAFN+)

3. Extended DCG plus (E-DCG+)

In E-SAF+, E-DAFN+ and E-DCG+, instead of access frequencies, *RWR* (Read Write Ratio) of a data item is used as the decision parameter for replica allocation. *RWR* value indicates the ratio of probability of an access request for data item d from mobile host M in Δt time period to probability of an update for data item d from its owner mobile host. Thus, if *RWR* value is high, there are more read requests then updates to data item d. If data items with *RWR* values are prioritized during replication, then replica consistency of the network is protected for a longer time.

An important disadvantage of DAFN (and E-DAFN+) and DCG (and E-DCG+) methods is that they can cause large *broadcast storms* [14] in the network when the

number of nodes is large. This is because every node broadcasts its access frequency or *RWR* value for each data item in every allocation period. Besides, each of these methods does not consider load balancing among the mobile nodes while replicating data items, which means that after replicating the data items based on access frequencies, total number of accesses to replicas for each mobile host may not be balanced. Nodes that replicate popular data items consumes more energy than others. Besides, in these techniques, due to mobility and node failures it may not be possible to find stable nodes to replicate the data items.

2.1.2 DREAM: A Data Replication Technique for Real-Time Ad Hoc Mobile Databases

Another data replication method for MANETs is DREAM [15]. It 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. In other words, data transaction classification, based on their time limitations (soft or firm) or read/write status, plays an important role during the selection of replicated data. DREAM is similar to [12], in means of replication decision based on access frequency of data items. However, differently, this technique extends the replication decision with new decision parameters: *link stability*, *transaction types*, *data types* and *remaining power*.

2.1.3 CADRE: Collaborative Allocation and Deallocation of Replicas with Efficiency

CADRE [16] tries to achieve fair replica allocation among the mobile nodes in the network and makes the deallocation decisions in mobile nodes *collaborative*. By doing allocation and deallocation decisions jointly among nodes in the network, CADRE can protect the network from the multiple re-allocations of the same data item, which can lead to a *trashing condition*, that mobile nodes spend more resources on re-allocation of data items than replying other nodes' requests. Most of the data replication techniques developed for MANETs give replication decisions based on the access frequen-

cies and read/write ratios of replicas. As a result of this, if a data item is accessed very frequently in the network, then it is replicated with high priority, even if it is accessed by only a small group of nodes in the network. However, for the sake of the entire network (or fairness in serving the needs of multiple nodes), kind of replication decisions should be avoided. CADRE suggests fairness among replicated items by assigning a score α to each data item d. α is the parameter that measures *the importance of data item to the whole network*. Thus, α should be higher for the data items that are preffered larger number of nodes in the network.

CADRE constructs a hierarchical network structure, in which some of the mobile nodes act as a *Gateway Node - GN*. *GN*s act as super-peers that are selected among the mobile nodes in the network with higher available bandwidth, higher energy, and higher capacity. They are responsible for replication and query propagation in the network. Unfortunately, if every node in the network has similar capabilities (with respect to bandwidth, energy, memory), than the nodes that are elected as *GN* will be affected from this situation in an *unfair* manner. This unfairness can cause increase in the network delay when serving data requests, due to overload in *GN*s.

2.1.4 CLEAR: Consistency and Load Based Efficient Allocation of Replicas

CLEAR [17] considers node loads, data sizes, consistencies of replicas and user schedules during replication decision. CLEAR assumes that the cost of maintaining a desired level of consistency for the replicas of a data item d can be estimated from the percentage of change in the value of an attribute of d.

Similar to CADRE [16], CLEAR is based on a hierarchical network structure. In this hierarchy, *Cluster Heads* (CHs) are responsible for the data validation and replica allocation decisions of their own clusters. Similar to other cluster-based schemes, CH is selected among the mobile hosts in the cluster with the best capabilities (in terms of features such as energy, memory space and bandwidth). In this kind of network structure, every node in the cluster sends its update logs and replica lists to their CHs. After collecting these data, a CH decides the data items that should

be replicated. Data size and access frequency of an item play role in this decision. After the candidate data item is selected, CH determines candidate mobile hosts in which the data item can be replicated. Load, access frequencies, memory and energy constraints of a mobile host are used for the replication decision. This technique calculates the *cost-effectiveness of the replication for data item d in mobile host M* function to give the final replication decision.

2.1.5 EcoRep: An Economic Model for Efficient Dynamic Replication in Mobile-P2P networks

EcoRep [18] proposes a data replication scheme for mobile peer-to-peer systems that is based on an economical model during replica allocation and query requests. For this purpose, EcoRep system puts a price for every data request in the network, and requires the requester to pay this price to the node that serves this request. Price of a data item is calculated by the *access frequency*, *number of users accessed*, *number of existing replicas of that item* and *consistency* (if it is a replica of the original item). Using a pricing system for every query in the system, EcoRep tries to minimize the number of *free-riding* nodes. Free-riding nodes are the ones that do not participate in replication of data items. In EcoRep, this kind of free-riding will be discouraged because every node needs some money (authors called currency) for sending query requests to the network. At the same time, nodes analyze the system and replicate the data items that provide the most *revenue* for themselves. EcoRep also considers *energy*, *load* and *network topology* as replication criterion (factors that affect the price of a data item). For example, if the serving node has a high load during a request, then the price of that request will decrease, because the serving node cannot give a high quality service in means of response time. EcoRep does not consider scalability in terms of message overhead.

2.1.6 Expanding Ring Replication

Expanding Ring [19] replication technique is proposed to increase data availability in pull-based information dissemination environments. In *pull-based* access methods, mobile nodes query the server for data they need. In this work, server is assumed to be static and known by every node in the system. By this way, the mobile ad hoc network constructed and data request type are different from the previously discussed techniques. In this technique, decision to replicate data is handled by the central static server. Server manages the replicas of the demanded data in such a way that probability of every node finding the data within its close neighborhood increases. This reduces the energy spent by nodes due to packet transmissions.

In this scheme, the data server maintains a set of hop count values for every data item (i.e. two sets [1,3,5] and [2,4,6]). Each time the access frequency of a data item exceeds a certain threshold, the server selects a set of hop count values (i.e. [1,3,5]) for that data item and starts to replication on each of the nodes in selected hop set.

2.1.7 Chen's Integrated Data Lookup and Replication Scheme

Chen et al. [20] proposes an integrated data lookup and replication scheme for the concept of group data access. Group data access means a group of nodes hosts a set of data items for all other members' usage in the group. Group members can communicate with each other via underlying routing protocol. Every node broadcasts a periodic *ad* message, in which a list of data items replicated in that node is carried. Data lookup is achieved by local data lookup tables, which was filled with the incoming *ad* messages.

[20] uses a predictive data replication scheme, which predicts the partitioning in a network and replicates selected data items in each partition. In this part, they assume that node movement behavior is predictable and with the help of location information and velocity of a node, system can predict the upcoming partitioning. It is stated that this information is taken from the routing layer, which implements a predictive location-based routing protocol. Routing protocol is also defined as part

of the cross-layer design given in [20].

In order to spend energy efficiently, this scheme places more replicas at the nodes that have higher capabilities (a combination of free memory, energy and processor workload). Moreover, in order to make data advertising scalable in large scale networks, they adapt the rate of message sending to the number of ongoing messages. This work (like all other replication approaches) is a proactive approach in which data is replicated on nodes explicitly.

Finally, this work is similar to our proposal in terms of the services it offers, namely data lookup and replication. However it may not be a scalable solution due to periodic flooding of ad messages. It is also a complicated cross-layer design, which may be hard to apply in real life scenarios.

2.1.8 DHTR: Distributed Hash Table Replication

Yu et al. proposes an optimistic replication scheme based on clustering of nodes into hierarchical groups [21]. It is *optimistic* because it tolerates replicas to be independently read or modified while data updates are in progress. It uses a distributed hash table for quicker data lookups in a mobile network. 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* (CH). CHs are responsible for controlling query and update processes in the network. Every mobile node in a cluster is registered to its group over a CH. Number of CHs is constant and assumed that it is known by all of the nodes in the network. In DHTR architecture, there are also *replica managers* (RM) responsible for holding replicas, replying to the replica requests and informing CHs about the state of their replicas. Also for every data item in the system, a Replica Keeper (RK) is associated. Every CH uses a common hash function to map a data id to the cluster id of that item's RK.

One of the important advantages of DHTR over flooding or gossiping based query and update propagation techniques is that DHTR constructs a tree-like communication architecture in which every query and update message is directed over CHs and RKs.

Yu et al. does not provide a detailed replica management protocol or any details of which data to replicate where. Generally, it targets a scalable query and update propagation technique that can be used in a cluster-based replication algorithm. They also do not define a limited memory space size for replicating nodes, and in their simulation results, they do not investigate data accessibility of the system. Instead, they show improvements on update propagation delay and consumed energy. DHTR also depends on an underlying routing protocol, which we believe could cause scalability problems in large network sizes. Another critic about DHTR is that constructed clusters may not be equally sized and even in the worst case, all other nodes other than CHs can be a member of a seperate cluster. Although DHTR suggests a scalable solution for data replication, it may not be the case mostly due to the problems explained above.

2.2 Cooperative Caching Techniques for MANETs

Cooperative caching is a widely used mechanism in the Internet for better Web performance [5]. Basically, it is the coordination of several nodes to share a cached data in an efficient way for all. In mobile ad hoc systems, cache-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 consumption. In the remaining of this section, some of the important caching techniques will be reviewed.

2.2.1 Yin and Cao's Caching Techniques

Yin et al. [5] propose three simple data caching techniques, which can be embedded as a middleware support between routing and application layer protocols. CachePath,

CacheData and HybridCache are the three techniques proposed.

CachePath 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. This scheme solves caching space compared to caching the data item.

CacheData 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. This scheme requires more memory space than previous, so it should be used wisely. To avoid redundant caching by several nodes in a path from requester to source, CacheData enforces a simple rule for all nodes: A node does not cache the passing-by data if all requests for that item is coming from the same node. In order to give some level of cache consistency support, both techniques use a simple consistency mechanism based on the time-to-live. Every data cached has a time-to-live and until the end of that time, if no new cache of the same item comes, it is invalidated and requests are forwarded to the source.

HybridCache 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 both of these techniques, mobile nodes consider two factors in selecting the cached item to be replaced: distance and access frequency.

2.2.2 COOP: A Cooperative Caching Service

COOP caching [8], similar to [5], 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. On the other hand, which data to cache is handled with a cache management strategy, which uses two simple rules to decide.

In cache resolution, when a node receives a data request, first of all it checks its own memory. If data is not in memory, it checks its RRT (Recent Requests Table), which keeps the previously received requests. This table avoids duplicated flooding for the same data item that was previously searched. If still no matching is found, then COOP 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. [8] names this three-step mechanism as *cocktail resolution* scheme.

COOP also studies how to decide which data item to keep in limited cache. They categorize the items as primary and secondary based on their existence in the neighborhood. If a data item is primary, it has priority in cache. If it is secondary, then LRU (Least Recently Used) cache management algorithm is applied to the existing cache. Similar to [5], COOP also uses TTL-based consistency control mechanism for cached data items.

2.2.3 Benefit-based Data Caching

Tang et. al [7] considers the optimization problem of minimizing total access cost of an ad hoc network with limited memory space and multiple data items. 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.

2.3 Feature Comparison of Data Replication and Caching

Replication refers to the action of creating local or nearly located copies of useful data items in order to avoid later communication overhead or even unavailability in case of on-demand retrieval from their original location [22]. 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 statistics.

In this section, we compare the data replication and caching techniques that we reviewed using common features that are critical for a replication/caching technique in Table 2.1. Compared features are listed below:

- Cluster-based Architecture : Does the technique require or construct a cluster-based architecture?
- Centralized Architecture: Is there a central server that manages every request?
- 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?

- Consider Scalability: Is the technique developed for scalable networks?

Table 2.1: Feature Comparison Table

	SAF, DAF and DCG [12]	DREAM [15]	CADRE [16]	CLEAR [17]	EcoRep [18]	Expanding Ring [19]	Chen et al. [20]	DHTR [21]	Yin et al. [5]	COOP [8]	Benefit-based Cache [7]
Cluster-based	no	no	yes	yes	yes	no	yes	yes	no	no	no
Centralized	no	no	no	no	no	yes	no	no	no	no	no
Consistency Mech.	yes	yes	no	yes	yes	no	no	yes	yes	yes	no
Limited Memory	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes
Routing Dep.	no	yes	no	no	no	yes	yes	yes	yes	yes	yes
Fairness	no	no	yes	yes	yes	no	yes	no	no	no	no
Consider Scalability	no	no	no	no	no	no	no	yes	no	no	no

2.4 Connected Dominating Set Construction Algorithms

Connected dominating set (CDS) of a graph is a special dominating set (DS), in which dominator nodes are connected. On the other hand, a dominating set is a set of graph vertices, which all other vertices not in this set are one hop neighbors of one of the dominating set member. We used CDS to construct a virtual backbone in ad hoc networks. Details of DS and CDS are given in Sections 3.2.3 and 3.2.4, respectively. In this section, we will give a survey of existing CDS construction algorithms for MANETs.

2.4.1 Performance Criteria and Costs of CDS Construction

An important performance criterion for an efficient CDS construction algorithm is its approximation factor (β). A β approximation algorithm guarantees that the ratio of the size of created dominating set to the optimal set does not exceed β. Costs of a CDS construction algorithm are message and time complexity of the algorithm [23]. Message complexity represents the number of messages sent in order to construct a CDS. Time complexity is basically the number of steps in the algorithm.

2.4.2 Previous Works

Chen et al. [24] proposes that even CDS provides an obvious routing structure for messages, the strict connectivity rule causes rather large dominating set size. In their algorithm, they relaxed this requirement and named the new structure as Weakly Connected Dominating Set (WCDS). In WCDS, direct connection between dominator nodes are not required anymore. As a result, WCDS is smaller in size compared to CDS. This complicates the routing decision, however simplifies the network structure. It is also shown that message overhead of this algorithm is larger compared to classical CDS construction algorithm due to messaging between neighbors in any state change [25].

Stojmenovic et al. [26] presented several distributed CDS construction algorithms for mobile ad hoc networks. In their CDS, there exist two types of network nodes: *(a)* cluster head (CH) and *(b)* border node. CHs form a maximal independent set (MIS) of the unit disk graph of given MANET. A node is a border node if it is not a CH and have at least two CHs within its 2-hop neighborhood. In order to satisfy a total ordering of all nodes, [26] assigns rankings to CHs based on their degrees and locations. *Degree* of a CH is the number of its one-hop neighbors. At the start of the distributed CH selection algorithm, every node is colored *white* and in every step the lowest ranked white node among all white neighbors is colored *black*. Then all the white nodes adjacent to black nodes are assigned *gray*. At the end of algorithm, every node is either gray or black; and black nodes become CHs. The approximation

factor of this algorithm is $\Theta(n)$, message complexity is $O(n^2)$ and time complexity is $\Omega(n)$, where n is the number of nodes [23].

[23] and [27] proposes two similar message-optimal CDS construction algorithms. Authors claim that [23] has an approximation factor of at most 8, $O(n)$ time complexity and $O(n\log n)$ message complexity. Similarly [27] has constant approxiation factor and linear time message and time complexities.

Both algorithms start with construction of a MIS. A brief definition of MIS is given in Section 3.2.3. Constructed MIS guarantees that the distance between any pair of its complementary subsets is exactly two hops. Then in [23], ranks of nodes are given by a distributed leader election algorithm. Differently, [27] does not depend on a third party distributed leader election algorithm. After these steps, both algorithms start their complex CDS constructions. According to these algorithms every node has to manage 10 to 15 different local variables; send 7 to 10 different messages.

Wu et. al proposes a simple and efficient CDS construction algorithm in [28]. Their algorithm first finds a CDS and then starts prune certain redundant nodes from the CDS by applying two defined rules. The initial CDS consists of all nodes which have at least two non-adjacent neighbors. Then the redundant dominators are cleared if their node ID is smaller than dominator node(s), which dominate(s) all its neighbors and itself in the given graph. In Section 3.2.5, this algorithm is explained in detail.

Table 2.2 summarizes previously mentioned performance criteria and construction costs of discussed algorithms.

Table 2.2: Summary of several CDS construction algorithms' performances: n is the number of nodes, Δ is the maximum degree of a node in the graph.

	[24]	[26]	[23]	[27]	[28]
Approx. Factor	$\ln(\Delta+1)$	O(n)	8	O(1)	n
Message Cost	>O(2n)	$O(n^2)$	O(nlogn)	O(n)	O(n)

Chapter 3

SCALABLE DATA LOOKUP AND REACTIVE REPLICATION FRAMEWORK

In this thesis, we propose a scalable data lookup and replication (SCALAR) framework for MANETs. This is an application layer solution composed of three parts:

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

Our solution aims to be scalable such that it does not create too much message overhead to the network with the increasing number of nodes. In addition to this, it performs well in means of data accessibility. The idea behind performing well in large scale MANETs with minimal message overhead is the construction of 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 and proposed by Wu and Li [28] for ad hoc wireless networks.

Proposed data lookup protocol takes the advantage of using a backbone which dominates the set of connected network nodes. Due to dominating behavior of constructed virtual backbone, every node in a connected network is a neighbor of at least one backbone member.

Third part of our proposed solution is a distributed data replication scheme on top of the scalable data lookup protocol. This scheme increases data availability and provides lower message overhead to the system without putting any extra message

cost over our scalable data lookup protocol. 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 of an item to the requesting node are our replication decision parameters in this scheme. Basically, in replication process, nodes are eager to replicate data items that are further from them with higher request frequencies.

This chapter is organized as follows: In Section 3.1, the basic system model is described. Then in Section 3.2 gives details of the construction of a virtual backbone. Basics of the dominating set theory and distributed CDS construction algorithm used are described in this section. Section 3.3 presents our scalable data lookup protocol. Finally, in Section 3.4 we describe the details our reactive data replication approach and the theory behind it.

3.1 System Model

System environment is assumed to be 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 host M_i is the owner of data item d_i, where the set of all data items are denoted as $D = \{d_1, d_2, ..., d_N\}$, where N is the total number of nodes. 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)*. Memory capacity of a node, $sizeOf(M_i)$ is fixed and equal for every node in M. Every M_i can save replicas of data items in set D, limited with its memory capacity.

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. Every node is assumed to be identical and there is no special node with different hardware or software capabilities. Each node shares its data items and cannot be in any kind of selfishness. Our proposed solution classifies

nodes into two types based on their assigned responsibilities in the system architecture: backbone node or end system. However, this classification is dynamic and purely according to node positions in the network graph. Every mobile node, M_i sends a request for a data item, d_j if it is not already available in its memory. A data request is successful, if requesting node holds the original/replica of requested data item or it receives the data item from any node in the network. In this thesis, we assume an environment where every data item is unchangeable, so that *no updates* to data items and related consistency problems are considered.

3.2 Virtual Backbone Construction Algorithm

MANETs assume wireless communication between mobile nodes without any physical infrastructure support. However, this infrastructureless communication causes increased communication costs and can be severe in large scale ad hoc networks. A common message overhead source in a MANET environment is blind flooding/broadcasting. In this type of communication, every node forwards incoming packets once they receive. Flooding-based communication is used in the route discovery phases of several routing protocols developed for MANETs [29]. However, this flooding/broadcasting packets in the network cause the creation of excessive redundant packets (*broadcast storm problem* [14]) and collusion/contention problem in wireless channel. For example, [9] points out that AODV [11] routing protocol has problems in large scale networks.

3.2.1 Virtual Backbones

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. There are several virtual backbone construction approaches available in the literature [30] [31]. Besides, in [32] authors show 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.

3.2.2 Unit Disk Graphs

Unit disk graphs are the intersection graphs of equal sized circles in the two-dimensional plane [33]. That means, we form a vertex for each circle in the plane, and connect two vertices by an edge whenever the corresponding circles cross each other.

The topology of a wireless ad hoc network can be modeled as a unit disk graph *G=(V,E)*. In this adaptation, it is assumed that all nodes have homogenous wireless communication capability and omnidirectional antennas. Intersection of circles in unit disk graphs represents that two network nodes are inside the wireless communication range of each other in an ad hoc network. Figure 3.1 illustrates a unit disk graph representation of a wireless ad hoc network.

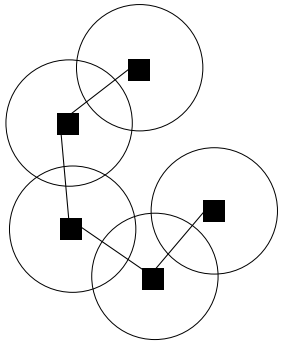


Figure 3.1: A wireless ad hoc network illustrated as a unit disk graph.

In general, for the construction of a virtual backbone, connected dominating set (CDS) of the unit-disk graph of a corresponding MANET is used. Definition of a dominating set and basic theory of dominating set problem will be given in the next section.

3.2.3 Dominating Sets

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.

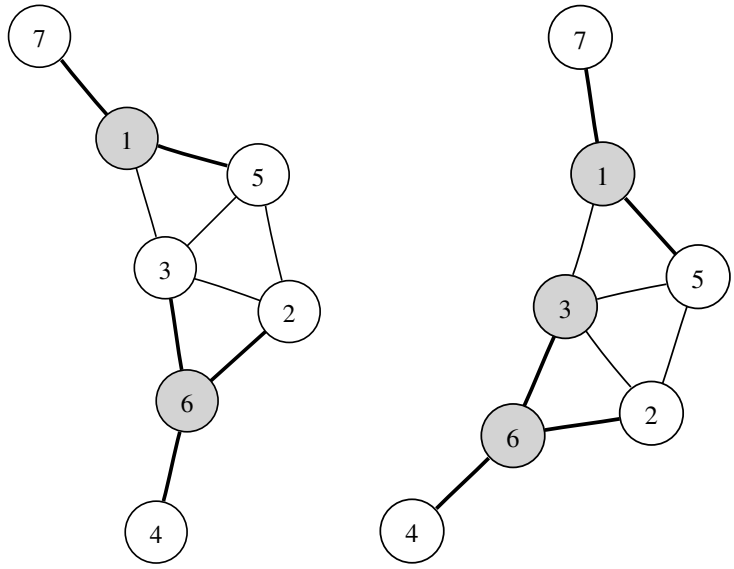
Figure 3.2.(a) highlights a dominating set in a connected graph. Dominating sets are closely related to independent sets. In graph theory, an independent set of a graph *G=(V,E)* is a subset of vertices *V'* such that no two vertices in the *V'* represent an edge in *E*. Actually, a *maximal* independent set (MIS) in a graph is also a *minimal* dominating set.

In practice, the vertices of a dominating set in a unit-disk graph can be used as cluster heads (CH) in ad hoc network clustering algorithms. Each vertex can be assigned to the CH which dominates it. By this way, it is guaranteed that every node in an ad hoc network environment is only one-hop away from its CH (i.e. within the communication range of a CH). This may be a useful feature in order to decrease the number of broadcast packets between cluster members.

Minimizing the size of dominating set in a given graph G can be useful in many application areas. For example in MANETs as the number of dominators decreases, the network structure becomes simpler and the number of packets sent between CHs decreases. However, finding a minimum size dominating set (called dominating set problem) of G is proven to be NP-Complete by a reduction from the *vertex cover* problem [34].

3.2.4 Connected Dominating Sets

A connected dominating set (CDS) of a graph *G=(V,E)* is a dominating set whose induced subgraph is connected (Figure 3.2.(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.



(a) DS (b) CDS

Figure 3.2: A Dominating Set (DS) and a Connected Dominating Set (CDS) illustrations of a given network

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 [35].

Several distributed CDS construction algorithms for MANETs are discussed in Section 2.4. We believe that these algorithms are harder to implement, execute or maintain in an actual mobile environment than [28]. In this thesis, we prefer to use Wu et al.'s simple and easily maintainable CDS construction algorithm [28] as the virtual backbone construction mechanism.

3.2.5 Wu's Connected Dominating Set Construction Algorithm for MANETs

[28] proposed a simple distributed algorithm in order to find a CDS in a unit disk graph *G=(V,E)* of a MANET. In general, the algorithm has the following properties:

- 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 (if the graph is a connected graph).

- 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 between 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: 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.

Initially all nodes start as marked false (not a dominator) and if G is fully connected, there will be no dominator in the network after the marking process, which is meaningful when 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. Wu and Li [28] propose 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. It is easy to show that 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) ⊆ N(u) ∪ 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', as a result.

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 connected dominating set.

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

3.2.6 Distributed Implementation of Wu's Connected Dominating Set Algorithm

In this section we will describe our distributed implementation of [28], and the way we adapted this algorithm to mobile scenarios. We also describe the relaxations applied to some of the strict assumptions in the algorithm for handling fault tolerance in case of mobility. In order to maintain the CDS in constant mobility scenarios, we recalculate the CDS by periodically running CONSTRUCT-CDS method given in Figure 3.3.

Before CONSTRUCT-CDS is executed,it is assumed that every node concurrently starts the algorithm within a small time period (Δt). During the execution of CONSTRUCT-CDS, if a data from the neighbor list of a node cannot be received due to collusion or mobility of nodes, the procedures can continue to work with lower performance values (i.e. increased CDS size).

After the execution of CONSTRUCT-CDS every node in the network is marked as TRUE or FALSE. If a node is marked TRUE, then it is a *dominator/backbone* node. If it is marked FALSE, then it is *dominatee/end system* node. Types of nodes play an important part in our scalable data lookup protocol described in Section 3.3.

Period of the CDS reconstruction depends on the mobility behavior of nodes. If nodes are moving fast, frequent calls of CONSTRUCT-CDS may be necessary to sup-

```
CONSTRUCT-CDS()
 1  neighborList ← NIL
 2  myNeighbors ← NIL
 3  dominatorList ← NIL
 4  mark ← FALSE
 5  x ← 0
 6  y ← 0
 7  BROADCAST(hello)                    ▷ Receive Hello concurrently!
            myNeighbors[y ← y + 1] ← RECEIVE()
 8  BROADCAST(myNeighbors)              ▷ Receive N. Lists concurrently!
            neighborList[senderID] ← RECEIVE()
 9  mark ← MARKING-PROCESS(myNeighbors, neighborList)
10  if mark = TRUE
11     then BROADCAST(Dominator-Announce)
                    ▷ Receive announcements concurrently!
                    dominatorList[x ← x + 1] ← RECEIVE()
12          mark ← APPLY-RULE-1(myNeighbors, neighborList, dominatorList)
13          if mark = FALSE                 ▷ I am not a dominator anymore!
14            then BROADCAST(Dominator-Pruned)
                          ▷ Receive pruned concurrently!
                          Remove from dominatorList(RECEIVE())
15            else mark ← APPLY-RULE-2(myNeighbors, neighborList, dominatorList)
16                 if mark = FALSE
17                   then BROADCAST(Dominator-Pruned)
                                ▷ Receive pruned concurrently!
                                Remove from dominatorList(RECEIVE())
```

Figure 3.3: CONSTRUCT-CDS method.

```
MARKING-PROCESS(myNeighbors, neighborList)
1  for i ← 0 to neighborList.size
2       do if ∀ myNeighbors ∈ neighborList[i]
3            then ▷ My neighbor i is connected to all my other neighbors!
4            else return TRUE
5  return FALSE              ▷ All my neighbors are connected to each others!

APPLY-RULE-1(myNeighbors, neighborList, dominatorList)
1  for i ← 0 to dominatorList.size
2       do if neighborList[dominatorList[i]] ⊇ myNeighbors
             AND dominatorList[i] > myID
3            then return FALSE
4  return TRUE

APPLY-RULE-2(myNeighbors, neighborList, dominatorList)
1  for i ← 0 to dominatorList.size
2       do for j ← i to dominatorList.size
3            if ((neighborList[dominatorList[i]] ∪ neighborList[dominatorList[j]])
                    ⊇ myNeighbors AND
                    myID = min(dominatorList[j], dominatorList[i], myID)
4            then return FALSE
5  return TRUE
```

Figure 3.4: MARKING-PROCESS, APPLY-RULE-1 and APPLY-RULE-2 methods.

port the connectedness of dominating set. However, we need to denote that according to this algorithm, even with frequent reconstruction of CDS, it is not guaranteed that nodes that are marked as true forms a CDS at any time instance. So this algorithm proposes a best-effort solution for the construction of CDS in MANETs. To the best of our knowledge, there exists no distributed CDS algorithm for MANETs that gives this kind of guarantee.

Cost Analysis

Message complexity of CONSTRUCT-CDS algorithm is $\Theta(n)$, where n is the number of nodes in the network. Distribution of messages sent by type is given below:

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

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

3.3 Scalable Data Lookup Protocol

In this section, we propose a lookup protocol for data items in a MANET. This protocol requires a constructed virtual backbone to operate and targets to increase success ratio and minimize the overhead in large scale MANETs.

3.3.1 Overview

Scalable Data Lookup Protocol is developed as a scalable solution to data lookup problem in large scale mobile ad hoc networks. Basically, it searches a data item in the entire system by sending the request to the virtual backbone, to which every node is directly connected. Request is only forwarded between the backbone nodes. This keeps the number of nodes involved in the lookup process limited. This limitation also decreases the message cost, which is an important performance burden in large scale mobile networks.

A straightforward solution to the data lookup problem in meso-scale ad hoc networks is *flooding* the request to the entire network. However it is shown that, flooding based solutions are very costly (*broadcast storm problem*) and result in serious contention and collusion in large scale wireless networks [14].

Another simple solution to the problem can 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 [11], DSR [36]). It is also shown that most of the popular MANET routing protocols has scalability problems due to route finding process based on flooding. [9] studies that the scalability issue for wireless multihop routing protocols is mostly concerned with excessive routing message overhead.

As a result, a scalable solution to the problem needs to overcome the drawbacks mentioned with a special design. 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.

3.3.2 Node Types

Scalable data lookup protocol assumes that every node in the system is either a :

1. backbone (dominator) node, or an

2. end system (dominatee) node.

Nodes are assigned to one of these types in virtual backbone creation phase, whose details are discussed in Section 3.2. Every node in the network is assumed to be identical and the decision of being a backbone node purely depends on the nodes' connectivity information during the virtual backbone creation.

Behavior of mobile nodes in the protocol depends on the node type. For example, forwarding is only allowed between backbone nodes during the search and transmission of requested data.

3.3.3 Roles of Node Types in the Protocol

In general, similar to the Internet backbone structure, every backbone node is connected to other backbone nodes and is placed in the core of message transmission from any source to destination. End systems are only allowed to send their data requests to one of their neighbors in the backbone. We named this as *request injection*. Call that a valid CDS construction guarantees that every node has at least one neighbor which is a backbone node (See Section 3.2.3, Dominating Sets).Our assumption is that a data item d is owned by the node with identification number *ID(d)* (e.g. node with id 3 owns the data item with id 3 by definition).

Backbone Nodes

Backbone nodes form the basis of the data lookup process. *Searching* for the requested data is the first part of our proposed scalable data lookup protocol. Success of searching and the message overhead of the system depend on the number of backbone nodes and the link stability between them.

Searching A backbone node participates in the data search process,

1. by *sending* a request generated by itself to a set of backbone nodes in its neighborhood.

2. by *forwarding* a received request to a set of backbone nodes in its neighborhood.

3. by receiving a request *generated* from an end system or backbone node.

4. by receiving a request *forwarded* from a backbone node.

In 1, a backbone node injects its own data request into the backbone. A backbone node decides to inject a new request for item d if none of the cases below are satisfied. Similarly, in 2, a backbone node forwards an incoming data request (as mentioned in item 3 or 4) for data item d (to a *random* set of its backbone neighbors), again if none of the three cases below are satisfied:

- d is owned by the backbone node.

- d is owned by one of the neighbors of the backbone node.

- d is owned by one of the neighbor of neighbors of this node in the backbone.

If one of the cases above is satisfied, it means that data item is either (a) owned by the backbone node or (b) found in the *two-hop vicinity* (recall that every node already knows its neighbors and neighbors of its neighbors from the virtual backbone creation phase).

- If (a) is the case, data is put into a packet (we assume that every data item can fit into one network packet) and sent directly to the node that the request is received from. Note that the node which the request was *originated* and the node which the request was *received* may be different due to forwarding of requests in backbone nodes.

- If (b) is the case, the request is forwarded to the appropriate backbone neighbor who either owns the data or is on the path to the owner.

During this request forwarding or new request injection, every backbone node 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 id, originator node of the request, req.  packet id>
```

This data structure will then be used to route the received data packet to the destination. This is the second part of our scalable data lookup protocol: *data receive*.

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 received. By this way, the system can be avoided from *infinite* request message transmission in the network. On the other hand, this situation may increase the unsuccessful searches, since some of the requests may be blocked and cannot forwarded anymore due to this ignorance of previously received requests. However, this is a very unlikely situation, since a search is done in multiple paths in the backbone. Although one request is blocked, others may still reach to the destination node.

Data Receive A backbone node participates in the data receive process,

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 the case of 1, requested data is packed into a network packet with an appropriate 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 (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 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.

Figure 3.5 demonstrates the state transitions of data lookup protocol for a backbone node. This diagram is a summary of the protocol explained above and provides a clear understanding of role of the backbone nodes. Every node in this system starts at *Listen Socket* state and gives state transition decisions based on received or sent packet types. At *Listen Socket* state, a node can receive 2 types of packet: Request or Data Packet. Based on the received packet, it changes its state to *Receive Request* or *Receive Data*. Dotted lines in the diagram represents that all the state transitions are per packet received or sent. This means when a new packet received and processed, nodes do not need to remember any state information (or wait at a state) for the next incoming packet. This property simplifies the implementation complexity of our protocol in event-driven network simulators and in real life applications.

End Systems

End systems (or host nodes) do not have so much role and responsibility in our data lookup protocol. They have simple decisions and do not forward any request or data packets that they received. Basically, end systems inject their requests to the backbone in order to lookup 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. An end system injects the

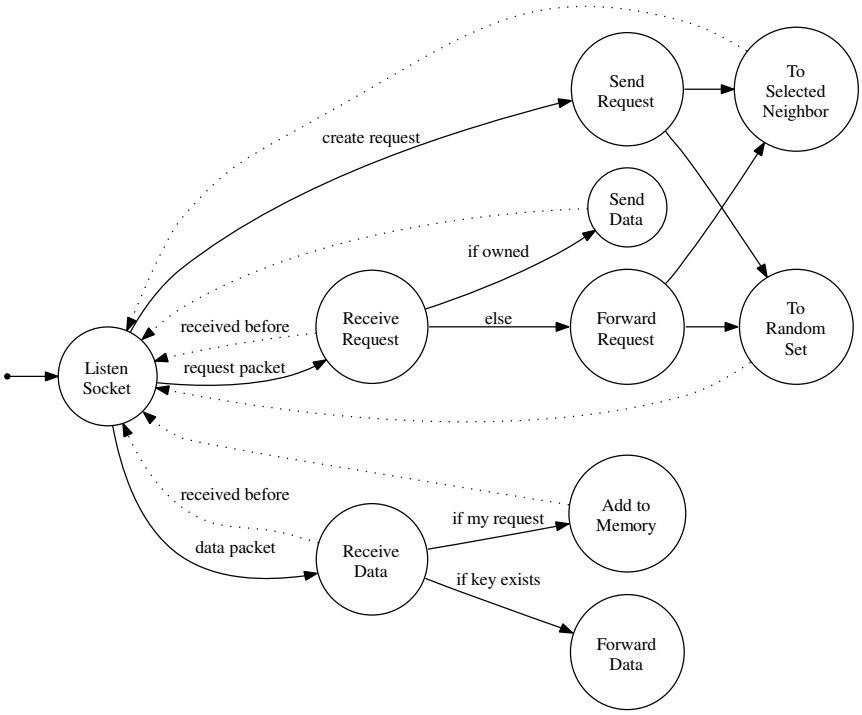


Figure 3.5: Backbone Node State Transition Diagram

request to the backbone by selecting one of its neighboring backbone nodes randomly and sending the request directly to it.

Searching An end system performs the following procedures to request an item d:

- If d is owned by one of the neighbors of the end system or one of the neighbors of the closest backbone node to this end system, then request is sent to the appropriate neighbor who either owns the data item or is on the path to the owner.

- else request is sent one of the neighbors in the backbone (*request injection*).

In either case, a unique key of the requested item and requester id is put into requested items data structure as explained for the backbone nodes (Section 3.3.3).

An end system can also be the *owner* of a data item requested in the network. In that case, a request is received from a backbone node or a neighboring node, which does not need to be a backbone member. In both, end system packs the requested data item and sends to the requester. End systems cannot be involved in the request forwarding mechanism in this protocol.

Data Receive Data receive is very simple in end systems: Every data packet received is the request of this node. Multiple reception for the same request is not possible because this problem is handled in backbone nodes before coming to end systems.

Figure 3.6 gives the state transition diagram of an end system. It is similar to the diagram given for backbone nodes (Figure 3.5), but it has less decision paths and possible states.

In the proposed protocol, 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. So it is highly possible that some of the end systems become backbone nodes at some period or vice versa.

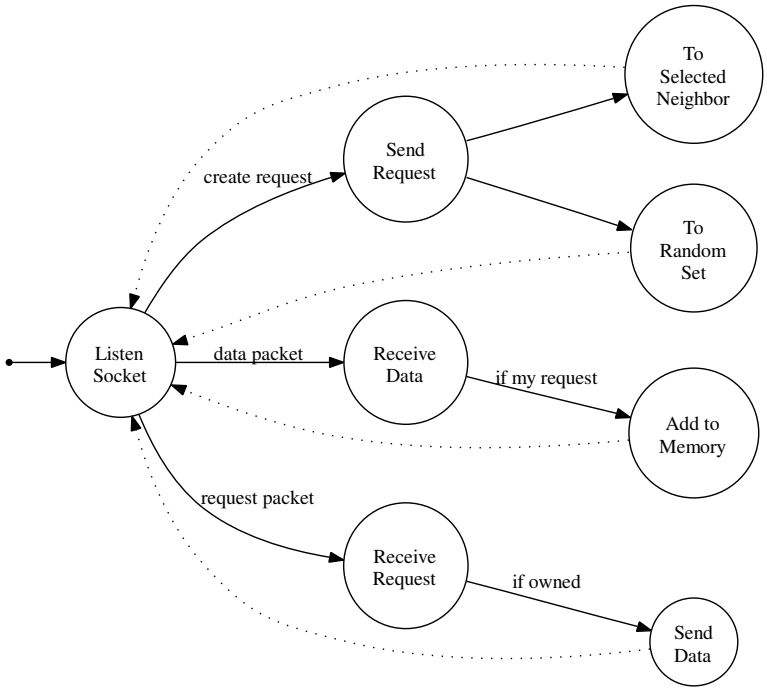


Figure 3.6: End System State Transition Diagram

3.3.4 Example Scenario

In this section, we present a simple data lookup scenario that helps the reader to understand how our proposed protocol operates. In the given figure, highlighted network nodes are backbone nodes and links connecting backbone nodes are bold. All other nodes are end systems.

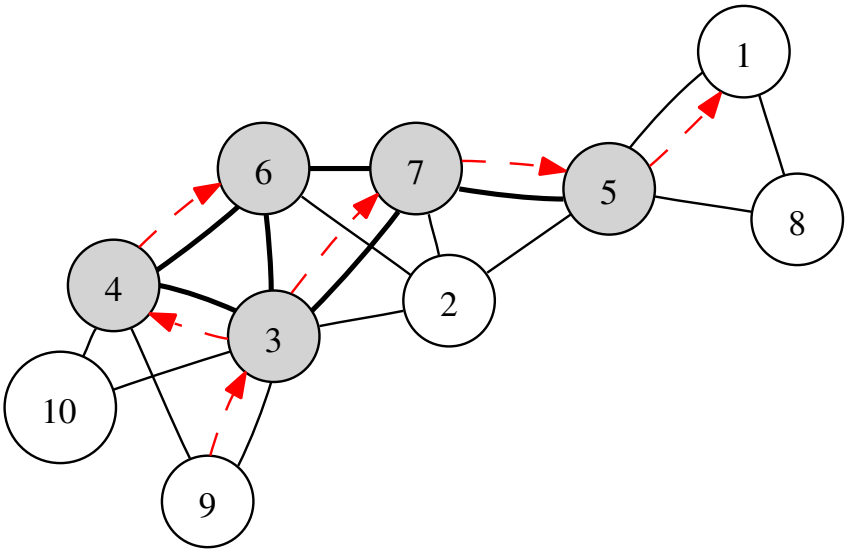


Figure 3.7: Example Scenario - 1: Scalable Data Lookup Protocol - Node 9 requests data item 1. Dashed lines represents the propagation of request packets.

First scenario discusses the case in which node 9 requests data item 1 at an arbitrary time instant t. Between t and $t+\Delta t$, unit disk graph representation of a MANET is shown in Figure 3.7. We assumed that Δt is a time period enough to complete the given steps below:

1. Node 9 decides to request data from the network, checks its neighboring nodes list looking for a backbone node. Selects one of them randomly (in this case 3) and sends (or *injects*) 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. So 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 (collected in virtual backbone creation phase). So it forwards the data to a set of random backbone nodes in its vicinity (set of 2, in this case).

3. When nodes 4 and 7 receives the request from 3, they do the same checks as 3. During these checks, 7 finds that node 1 is in the vicinity of its backbone neighbor 5. 7 forwards the request to 5. On the other hand, 4 have 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 six).

4. 6 forwards it to a random set of backbone neighbors, again. In our illustration, we did not show the ignored requests due to duplication. Actually, in this case 3 and 7 ignores the request coming from 6.

5. When 5 receives the request from 7, it can see that 1 is its neighbor and forwards the request to node 1. Node 1 packs the data and send it to 5. After that every node involved until request come to node 1 is used as the part of *return path*. Details of this return path construction is given in Section 3.3.3.

3.3.5 Cost Analysis

In this section, we analyze the worst-case message cost complexity of proposed scalable data lookup protocol. We will also define and analyze the parameters, namely, number of nodes in the backbone and the distance between requester and source, that effect the messaging cost of our algorithm.

Number of Nodes in the Backbone

The idea behind using a backbone structure for data lookup in an ad hoc network is to decrease the number of possible hosts that received the request while keeping the

availability of the data item at the same level with flooding-based solutions. According to our protocol, backbone nodes forward requests to other backbone neighbors. So if the size of backbone is large, cost of our protocol is large either. However, as discussed in Section 3.2.4, finding a minimal size virtual backbone from a given graph is a NP-Complete problem.

In a given connected graph *G=(V,E)*, some of the nodes are backbone nodes (*V'*) and they construct a reduced graph *G'=(V',E')* of *G*. *G'* is still a connected graph and every vertex (or backbone node) in *G'* can send requests to other vertices in *G'*.

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 node can forward data requests, number of messages sent in the network for a data request can be bounded to *O(k * V')*. Since k is a constant number, we can conclude that worst-case message complexity of scalable data lookup protocol in SCALAR framework is O(V'), where G'=(V',E') represents the reduced graph constructed by the backbone nodes (*V'*).

Also note that, the worst-case data receive cost of our protocol is $O(V')$, because in the worst case data is transferred from the leaf node of the requester rooted tree. So, *O(V')* + $O(V')$ is still *O(V')*.

In other words, message cost of this protocol is directly related to the size of constructed backbone (or the *approximation factor (β)* of the selected virtual backbone construction algorithm). Figure 3.8 shows the increase in the size of virtual backbone as the total number of nodes in the network increases. This result is taken from our simulations of distributed implementation of Wu's CDS algorithm (see Section 3.2.6).

It reveals that the algorithm can keep the size of constructed backbone at a satisfactory level even in large scale networks. Therefore, in the worst-case, our protocol can be described as reasonably *scalable*.

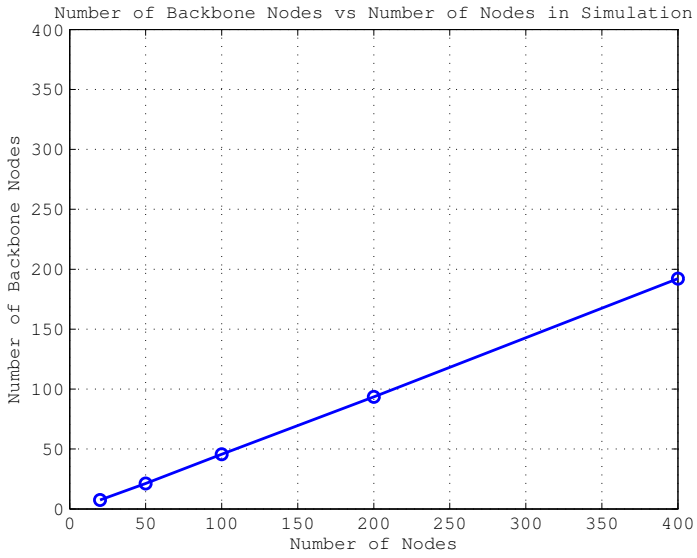


Figure 3.8: Number Of Backbone Nodes.

Distance between Requester and Requested Item Host

Another important criteria for the calculation of message cost in our protocol is the distance (number of hops) between the node who requested the item and the node who owns it. For instance, in the absence of a replication mechanism for data items, in Figure 3.9, if node 9 requests a data item from 7, message cost will be minimum, however if it requests from node 11, every backbone member will receive the request and forward it (which is the definition of worst case, discussed in previous section). So, closer the requested data items are to the nodes, better the message complexity of our protocol is. This proposition is the idea behind our replication algorithm discussed in the next section.

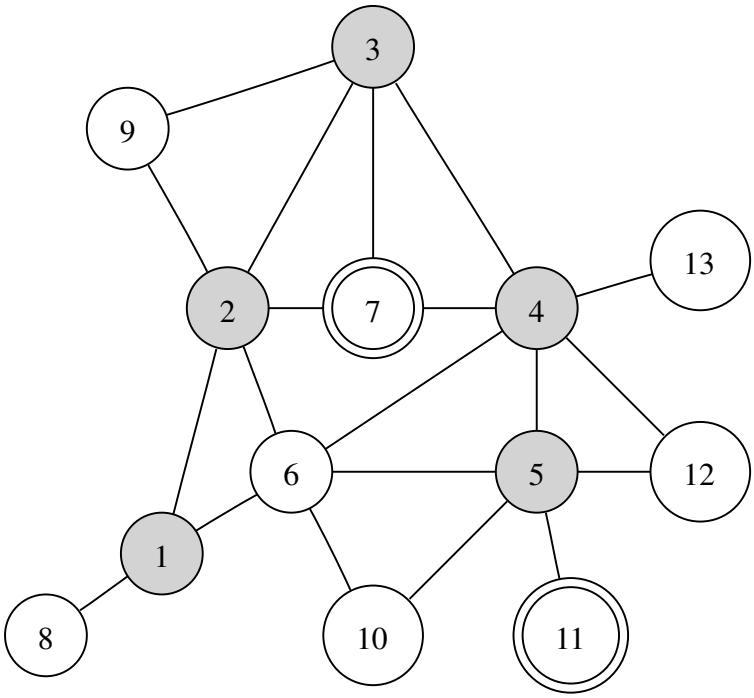


Figure 3.9: Messaging Cost: If the requested item is close (in terms of number of hops) to the requester, total message sent is lower. For node 3, requesting a packet from node 7 or node 11 has different message costs to the system.

3.4 Reactive Replication Approach

3.4.1 Overview

In this section, we propose a novel *reactive* data replication approach for large scale networks. This approach 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 packets is received. It does not require the exchange of explicit replication control packets. Thus, it completely eliminates the control overhead caused by other active replication schemes. Replication decision of data items is based on request frequency of an item and the distance of the received data's owner. Reactive Replication approach in SCALAR framework 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 approah also considers the request frequency of items during its replica allocation decision. 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 the best option for the next replication decision.

3.4.2 Properties of Reactive Replication

Here are some of the general properties of this approach:

- Replication decisions are based on node's local information; it requires no explicit control packet exchange, therefore it is pretty scalable in the number of messages sent.

- It 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.

- Every node dedicates some memory space in return for a global decrease in the messaging cost and query delays.

3.4.3 Roles of Backbone Nodes

A backbone node gives a data replication decision if one of the following cases is true:

- if received data is a personal request of this backbone node (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. This is for *maximum utilization* of memory spaces.

- if the data item is received due to a forwarded request message, 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 most node in the path, it may decide to replicate the data item, if not replicated already. 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.

In either case, backbone nodes has a tendency to replicate data items with higher cost more. Cost function of a data item received is given as follows:

Cost Function

This 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. Definition of the cost function is placed in the heart of our replication approach and critical to the performance of it. A high-performance replication approach is the one that increases the availability of data items, while decreasing message sent and query delays vastly. In Section 3.3.5, it is explained that closer the requested data items are to the nodes, lower the message complexity of our lookup protocol is.

Reactive replication aims to make frequently accessed distant data closer to the nodes using the cost function at node M_j for data item d_i:

$$cost(\alpha_i, h_{ij}) = \frac{\alpha_i}{\sum_{k \in D} \alpha_k} * h_{ij} \tag{3.1}$$

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.

Obviously, by looking at this cost function it is expected that replicating the data items with larger costs will help to *minimize* the cost of the next request to the system. This is because it will be highly probable that h_{ij} will be reduced in the cost calculation of next requested data item using a system-wide replication. An extra replica in the mid point of the data path will help to increase data availability if the data path is very large. If it is not, mid point nodes probably would not give a caching decision for data received. So replication in the mid points is a dynamic behavior, which adapts to the data path length. We believe that this behavior makes sense in large scale networks.

With the replication mechanism present in the system during the search for the owner of a data item, if a replica is found in the backbone, then searching process is stopped there and data receive process is started.

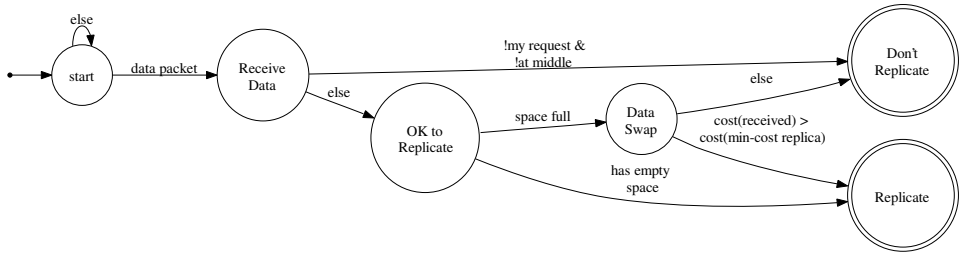


Figure 3.10: Backbone Node - Replication Decision Tree.

Figure 3.10 summarizes the replication process for a backbone node.

3.4.4 Roles of End Systems

End System's replication decisions are similar to the ones for backbones. However, because of end systems cannot forward data items, they do not give replication decisions about being on the mid point of data path, like end systems do. So, an end system gives a data replication decision if and only if the condition below is true:

- if cost of the received data item (calculated by the same cost function defined before) is at least as large as the lowest-cost item replicated in a full cache, then lowest-cost is replaced with the newly received data item. If cache space has still vacant position for a new item, then node does not need to make a cost comparison before adding the data item to its cache space.

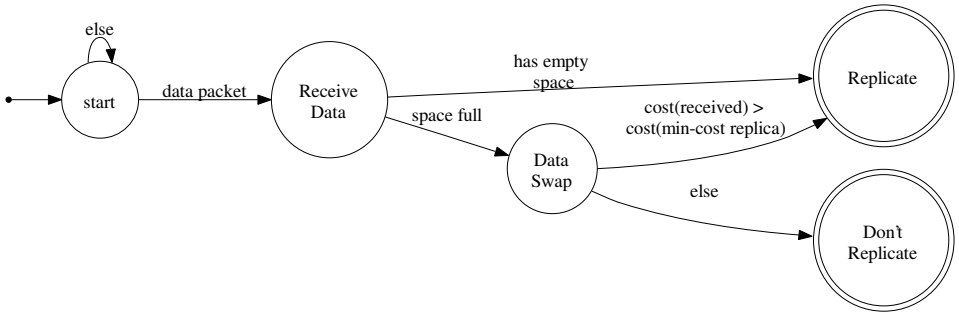


Figure 3.11: End System - Replication Decision Tree.

Figure 3.11 summarizes the caching process for an end system.

3.4.5 Example Scenario

Figure 3.12 represents a replication decision example for 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 possibly 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 as defined in Section 3.4.4: If the cost of the received data item is as large as the minimum-cost item in the memory(d_6 in this case), then put the received data in place of it. In our case, cost of the received data (calculated with *cost function*) is larger than already existing data, d_6. So M_9 replicates d_1 instead of d_6.

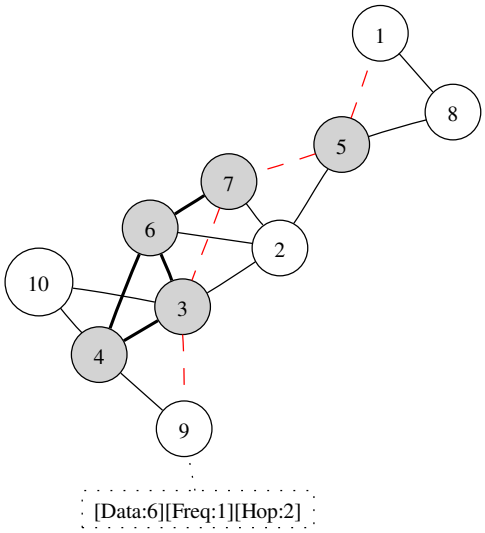


Figure 3.12: Example Replication Scenario: Node M_9 requests data item d_1, while its memory is full with d_6.

Also M_7 may replicate the data item d_1 during its travel to M_9. M_7 is in the middle of the path from M_1 to M_9 and definition of our proposed replication algorithm states that middle points on the backbone path starts a replication decision procedure for the data items that they forward.

3.4.6 Cost Analysis

Proposed reactive replication approach puts no extra cost on to cost of scalable data lookup protocol. It creates no explicit messaging cost. On the other hand, it increases the probability of finding a data item in closer, by this way it decreases the overall messaging cost of the scalable data lookup protocol in general.

Chapter 4

EXPERIMENTAL PRELIMINARIES

In this chapter, firstly the network simulator that we used to model SCALAR framework is introduced to the reader and the criteria for this selection is explained. Next, in Section 4.2, we introduce the simulation environment that we used for our simulations. Then, in Section 4.3, details of mobility models used in simulations are given. Finally, Section 4.4 defines two application scenarios that we used to evaluate performance of SCALAR along with other data replication systems, such as SAF and DAFN [12].

4.1 JiST/SWANS: Scalable Wireless Ad hoc Network Simulator

In this thesis, network simulations have been performed using SWANS (Scalable Wireless Ad hoc Network Simulation) tool [10]. SWANS is built on Java-based simulation framework JiST (Java in Simulation Time). JiST is a high-performance discrete event simulation engine that runs over a standard Java virtual machine. It is general-purpose discrete event simulator which is a type of virtual machine-based simulation. Since it is specifically developed for large-scale ad hoc network simulations, it outperforms simulation frameworks such as *ns-2* [37] and *GloMoSim* [38] in terms of event throughput, message-passing overhead and memory utilization. It is not necessary that simulation code running on JiST should be written in a domain-specific language invented specifically for writing JiST simulations. Instead, JiST converts an existing virtual machine into a simulation platform, by embedding simulation time semantics at the byte-code level. That's why, JiST simulations are written in pure Java language and compiled using a regular Java compiler. Consequently, created simulation executables can be run over a standard, unmodified Java virtual machine.

SWANS is a scalable wireless ad hoc network simulator package that is constructed upon JiST platform. SWANS is a combination of independent network components that forms wireless network or sensor network configurations. 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 [10]. Scalability of SWANS is critical in our network simulator selection process for the performance results of SCALAR system. Because, SWANS can simulate networks that are two times larger than what *GloMoSim* and *ns-2* can simulate, using the same amount of time and memory, and at the same level of detail.

4.2 Simulation Environment

Simulation environment is a meso-scale (from 20 nodes) to large-scale (up to 400 nodes) mobile ad hoc network. Nodes in simulation are identical and implements IP (Internet Protocol) and UDP (User Datagram Protocol) as network and transport layer protocols, respectively. In our simulation environment, network layer packet losses are ignored. However, MAC (Media Access Control) layer packet losses (i.e. MAC layer queue drops) are still applied. Every node is equipped with IEEE 802.11b capable wireless communication cards operating (in ad-hoc mode) on the 2.4 GHz band with communication range of 100 meters. Number of nodes in the simulation is varied from 20 to 400 in order to analyze the scalability properties. 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 collusion and contention on the shared wireless channel. Density of a network is calculated given as follows:

$$density = \frac{N * S}{A} \tag{4.1}$$

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.

Table 4.1: Simulation Defaults

MAC Protocol	IEEE 802.11b
Network Protocol	IP
Transport Protocol	UDP
Radio Comm. Range	100 m.
Radio Frequency	2.4 Ghz
Number of Nodes (N)	20 to 400
Node Speed	1 to 3 m/s

Simulation area is a square-shaped two-dimensional flat surface in which every simulation node move freely without any obstacles. Node movements are based on one of the described mobility models in the next section. Nodes select a random speed within the human walking speed limits (1 to 3 meters/second) during the simulations. A summary of simulation parameters are given in Table 4.1.

4.3 Mobility Models

Movement behavior of mobile entities is one of the most important concepts for the realistic simulation scenarios in MANETs [39]. As argued in [40], choice of a mobility model may affect the results of simulations significantly. Therefore, a better understanding of the behavior of mobility models and using the appropriate ones give us a chance to achieve realistic conclusions, which improves the validity of our results.

In this section, we will introduce two of the popular mobility models used in the literature. We implemented these models on JiST/SWANS and conducted simulations based on them.

4.3.1 Random Walk Mobility Model

The mathematical principles of the Random Walk Mobility Model go back to the Einsteins works on Brownian motion [40]. In some sources, this mobility model is called as Brownian Motion Mobility Model [41] or Brownian Walk [42]. In this mobility model, entities (mobile nodes) move randomly choosing a speed and direction from pre-defined ranges (*[minspeed, maxspeed]* and *[0, 2π]*, respectively) in constant time intervals (Δt). Border behavior of this model is defined as *bounce-back* . This means that when a mobile node reaches to a simulation boundary, it bounces back to simulation area.

Due to its simplicity of implementation, Random Walk is a commonly used mobility model in simulations. On the other hand, because of its *memoryless* behavior (i.e. decision of the next state doesn't depend on previous states) it may create unrealistic mobility patterns with sharp turns and sudden stops. It also exhibits a static behavior when it has a lower velocity range.

4.3.2 Random Waypoint Mobility Model

Random Waypoint Mobility is the most widely used model in simulations by the research community [40]. In this model, a mobile node selects a random position *(x, y)* in the simulation area as a destination point and a velocity *(v)* from a uniformly distributed range *[speedmin, speedmax]*. Then node starts to travel to the chosen destination point with the constant selected speed, $v_{selected}$. When the node arrives to the destination point, it pauses for a specific time *(pause time)* defined as a simulation parameter. After this time, node selects a new destination and speed and repeats the process [29].

Studies on the properties of Random Waypoint Model show that it creates a *non-homogenous* spatial distribution of nodes if it is used with specific (e.g. delete and bounce-back) border behavior selections [43]. This means that nodes tend to concentrate on the middle of the field. Several solutions to this problem are proposed in [44]. Another statistical behavior of Random Waypoint Mobility Model described in [45]

states that the average number of neighbors recognized at a given node periodically increases and decreases during the simulation. Frequency of this increase-decrease is relative to the speed of the nodes. This behavior creates a situation in which nodes unite and separate in the center of the simulation area. Authors called this situation *density waves*.

4.4 Application Scenarios

In this section, we introduce two application scenarios. Performance results of each scenario is given in Section 5.9. Our motivation behind creating these scenarios is to observe behaviour of replication approaches in realistic application settings of MANETs. In university campus and shopping mall scenarios, mobile agents are considered as people with handheld PCs, laptops or cell phones.

4.4.1 Scenario 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 communication capable devices belonging to pedestrian university students with velocity range 1 to 3 m/s. This is the average walking speed of a normal human being. These mobile devices are capable of keeping at most 5 data items for the purpose of replication in its memory space. Some of the data items in the network are popular and they are requested with higher probability ($\sigma = 0.6$) than other items in the network. Besides, only a small portion (10%) of all data items are set as popular in the network. Popular data items are selected randomly among all data items at the beginning of the simulation and not change. Every node requests exactly 10 data items during the entire simulation with the given probabilities or in other words probability of requesting a popular item at each request is 0.6.

A possible example of this scenario in a university campus can be as follows: In a university campus (which we assume a large campus area in comparsion to student population with low density), a group of students having cellphones, PDAs

Table 4.2: Scenario 1 - Parameters

Node Density = 2

Number of Nodes	Area Size (m2)	Node Velocity	1 to 3 m/s
20	316 x 316	σ	0.6
50	500 x 500	% of Popular Items	10%
100	707 x 707	Mobility Model	Random Waypoint
200	1000 x 1000	Request per Node	3 requests
400	1414 x 1414	Memory Size	5 items/node

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 to this system. Mobile students are walking in the campus area and pausing and talking to other people during their travels. Table 5.1 summarizes the simulation parameters of Scenario 1.

4.4.2 Scenario 2: Shopping Mall Scenario

Table 4.3: Scenario 2 - Parameters

Node Density = 5

Number of Nodes	Area Size (m2)	Node Velocity	1 to 3 m/s
20	200 x 200	σ	0.6
50	317 x 317	% of Popular Items	10%
100	447 x 447	Mobility Model	Random Waypoint
200	632 x 632	Request per Node	3 requests
400	895 x 895	Memory Size	5 items/node

Shopping mall scenario is similar to university campus scenario. It only differs in terms of node density. As you can expect a shopping mall is a denser place. Similar to scenario 1, nodes are mobile with a velocity selected randomly between [1-3] m/s. Concept of popular data items and their request probabilities are similarly applied to this scenario also. Table 4.3 summarizes the simulation parameters of Scenario 2.

4.5 Simulation Algorithms and Protocols

4.5.1 AODV - Ad Hoc On Demand Distance Vector Routing

In our simulations, we used the default AODV routing protocol implementation of JiST/SWANS tool, which captures all AODV capabilities.

AODV is a source-initiated, on-demand and table-driven routing protocol [11]. This type of protocols create routes only when they are desired by the source node. In other words, the network is silent until a path to a destination is needed. At that point the network node that needs a path broadcasts a route request message. Other AODV nodes forward this message, and record the node that they heard it from, creating an *explosion* of temporary routes back to the path requester. When a node receives such a message and already has a route to the desired node, it sends a message backwards through a temporary route to the requesting node. The path requester node then begins using the route that has the least number of hops through other nodes. When a route maintenance is required, a routing error is passed back to a transmitting node, and the same process repeats.

An additional aspect of the protocol is the use of *hello* messages which are periodic broadcast messages sent by each node to inform its neighbors about its existence. AODV is a costly protocol for high density and mobile ad hoc networks, due to its excessive usage of broadcast messages (hello, route maintenance, route request messages, etc.)

4.5.2 SAF and DAFN replication approaches

A brief explanation of SAF (Static Access Frequency) and DAFN (Dynamic Access Frequency and Neighborhood) approaches are presented in Section 2.1.1. Here, we will give simple execution of each approach in our simulation environment.

In SAF, at the start 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 flooded 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 his neighbors. If the requester receives the data, it puts it into his memory space and finishes the SAF algorithm.

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 same items are selected as candidate for replication, then it compares the access frequencies of its and its neighbor's 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.

Chapter 5

PERFORMANCE RESULTS

This chapter analyzes the performance of proposed *SCALAR* framework and compares it with data lookup using an underlying ad hoc routing protocol. Due to its popularity and reactive property, we choose Ad-hoc On Demand Distance Vector (AODV)routing algorithm as the underlying protocol for our base-case data lookup simulations. AODV is a source-initiated reactive(on-demand) protocol, which initiates a route discovery whenever a node requires a path to a destination. Then, we compare SCALAR with well-known data replication approaches developed for MANETs by Hara [12]. In these simulations, we aim to examine the performance of SCALAR in *large-scale*, *high density* and *high-load* network conditions, in which most of the data lookup and replication approaches fail. To analyze the performance of our system, we need to define metrics that points out the desired performance. For this purpose, we define four performance metrics, that are described as follows:

5.1 Metrics and Parameters

- *Data Accessibility:* This is an important criterion for both a data lookup and a replication protocol. Basically, data accessibility is the ratio of the number of successful access requests to the number of all access requests issued. A data replication (or caching) 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.

- *Packets Sent per Node:* Packets sent per node is defined as the average value

of number of packets sent from 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.

- *Packets Received per Node:* This is the average number of packets received per node. This performance metric is meaningful (with average packets sent per node) to show the average energy consumption of a node in the system. Basically, mobile nodes consume most of their energy to send and receive packets over a wireless channel. Also receiving a packet and processing it (even if it will be discarded at the application layer) requires some processing and it also consumes power.

- *Average Query Deepness:* The average number of nodes (or hops) traversed by a successful 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 (cache) 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 of the networking applications (such as VoIP (Voice over IP), real-time video broadcasting applications, etc.) require strict timing constraints. Lower average query deepness values result in lower delay for data requests.

The simulation parameters play an important role in the performance analysis of a protocol. Furthermore, they are helpful for understanding the protocol response to

Table 5.1: Simulation Area Sizes in meters (n x n)

Nodes/Density	0.5	1	2	3	4	5
20	632	447	316	258	224	200
50	1000	707	500	408	354	316
100	1414	1000	707	577	500	447
200	2000	1414	1000	816	707	632
400	2828	2000	1414	1155	1000	894

different network conditions and by this way provides more comprehensive analysis. We define simulation parameters and then present the effects of each parameter to the performance of SCALAR.

- *Number of nodes in the simulation*: We perform simulations from 20 nodes to 400 nodes to show the scalability of different approaches. In order to keep the node density of a network at a constant rate, we created a density versus number of nodes concept to find an appropriate simulation area size for each density level. Our aim is to keep the average connectivity of a network at a constant rate while increasing the number of nodes in the simulation. By this way, we can see the pure effects of increasing number of nodes in the simulation. Table 1 gives the simulation area size for each density level and number of nodes. Calculation of these values is based on the definition given in Section 4.2.

- *Simulation Area Node Density*: This is defined as the ratio of the sum of coverage areas of all nodes in the simulation to the simulation area. Basically, density=1 for 20 nodes indicates that 20 times coverage area of a node is equal to the total simulation area. Details of node density calculation are previously given in Section 4.2.

- *Available Memory of a Node*: In our simulation we assumed that every node in

the simulation has a local memory space dedicated for the replication of remote data items. For the sake of the simplicity, we gave the memory size of a node in terms of data item capacity, while assuming every data item is equally sized. For example, if a memory can contain 16 data items at maximum, then available memory of a node is given as 16 according to our definition.

- *Data Request per node (Request Rate)*: Request rate is defined as the total number of data requests per node during the entire simulation. This value is the same for every node and it is known by every node prior to simulation start. In other words, request rate defines the load imposed on the network.

- *Mobility Model*: Several studies showed that modeling of the movement characteristics of a mobile node in the simulation can affect the performance results of a protocol in MANETs. To show the effects of different mobility models on the SCALAR system, we implemented Random Walk mobility model for JiST/SWANS network simulator and conducted simulations based on it. We compare these results with the results obtained for the Random Waypoint model. Details of these mobility models are given in Section 4.3.

Before giving the effects of different simulation parameters on the SCALAR system, firstly we present the number of backbone nodes and average connectivity relations for different network sizes and density levels. We believe that number of backbone nodes in the virtual construction of a backbone plays an important role in the performance of other metrics. Since in SCALAR definition, data request, data receive and replication are constructed on a virtual backbone structure, size of this structure is critical to the data accessibility, packet overhead, query delay and deepness values. Results for each data point are averages of 10 simulations conducted with identical settings.

5.2 Number of Backbone Nodes and Connectivity Relations for Different Network Sizes and Density Levels

Figure 5.1 depicts the average number of backbone nodes for increasing number of nodes in the simulation for different density levels. As we can expect, for larger networks, we need more backbone nodes to protect the connected dominating set property of the given network graph. Note that, in our simulations in order to keep the density of a network at a constant rate, we increased the simulation area while increasing the number of nodes in the simulation. From Figure 5.1, we can also see that for high density networks, we can construct a virtual backbone with less number of backbone nodes. Actually, this is a natural result of Wu's CDS construction algorithm, because in denser networks Wu's algorithm can prune more backbone nodes at Rule 1 and 2 defined. 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 increase; which is the necessary condition for Rule 1 and 2 to prune a backbone node. So, we can conclude that SCALAR scales well to the increasing node densities in terms of number of backbone nodes.

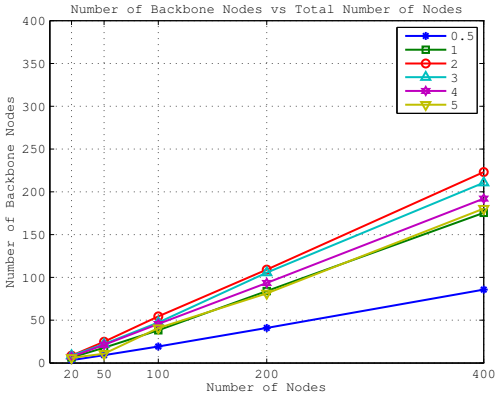


Figure 5.1: Average number of backbone nodes for increasing number of nodes in each density level.

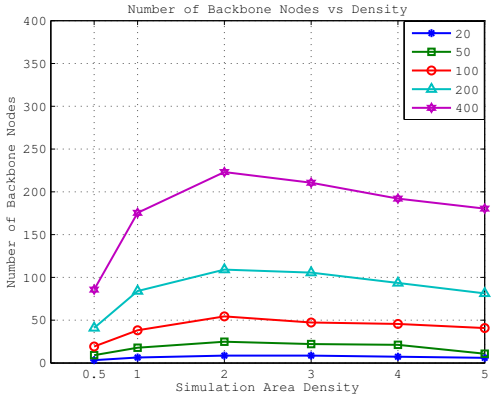


Figure 5.2: Average number of backbone nodes for increasing densities in each network size.

Figure 5.2 presents a similar result for the number of backbone nodes in the

simulation. In this case, it can be seen that as the simulation area density increases, number of backbone nodes increases up to a level and starts to decrease then. 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 a 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 as expected. However, after some threshold (2 in this case) value, density increase results in a lower number of backbone nodes due to increase in the probability of finding a backbone node that covers all the neighbors of another backbone node. From these results, we can conclude that on average the number of backbone nodes in a simulation is not more than 50% of all the nodes. 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 all nodes in the simulation. This property of *SCALAR* helps it to achieve low overhead in large and dense networks.

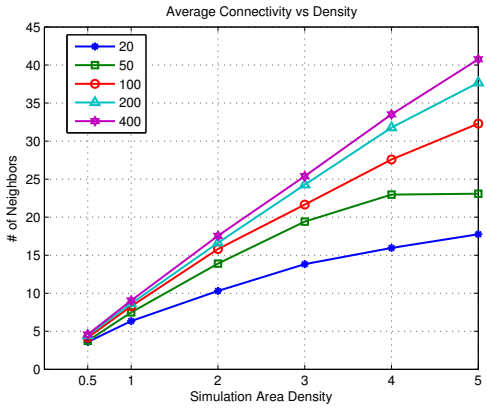


Figure 5.3: Average number of neighbors per node for increasing density levels of a simulation area.

In Figure 5.3, 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.

5.3 Effect of Number of Nodes - Scalability Analysis

In this section, we analyze the performance metrics defined while increasing the number of nodes in the network. Other than number of nodes, all other metrics and simulation parameters are kept constant and given in Table 5.3. In order to see the positive effects of using *SCALAR* as a data lookup scheme with/without replication in a MANET, we compare our results with a simple data request scheme, in which the path between requester and data source nodes is found using AODV routing protocol. AODV is a popular routing protocol in the literature and finds a route in an on-demand fashion. A brief description of AODV protocol can be found in Section 4.5.1.

Table 5.2: Scalability Analysis: Fixed Simulation Parameters

Simulation Parameters	
Popular Items	No
Mobility Model	R. Waypoint [1-3] m/s
Density	2
Node Memory Size	5 (data items)
Data Request per Node	10 (requests)
Total Simulation Time	300 sec.

Figure 5.4 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 5.5). 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

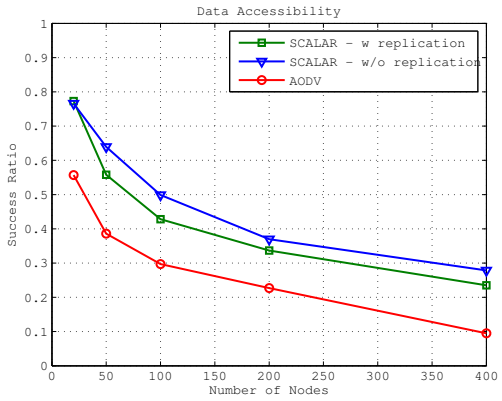


Figure 5.4: Average success ratio for increasing number of nodes.

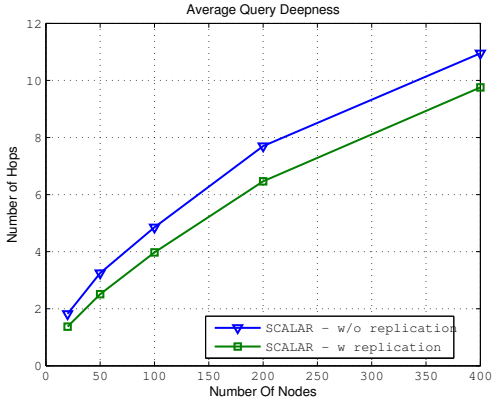


Figure 5.5: Average hop count of a completed data request.

hand, it can be concluded that *SCALAR* performs better than simple AODV based data request in terms of data accessibility.

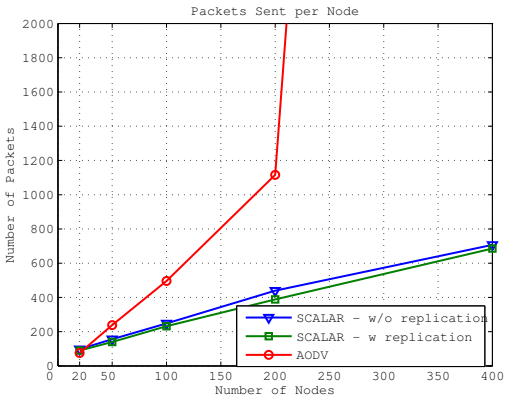


Figure 5.6: Average packets sent per node for increasing number of nodes in simulation.

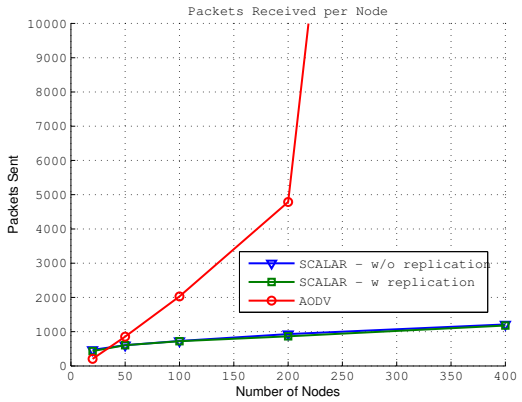


Figure 5.7: Average packets received per node for increasing number of nodes in simulation.

To understand why AODV-based data request performs worse, we need to look at Figure 5.6, in 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

collusion, 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. This is an important gain for a system which aims scalability. In Figure 5.7, we show the average packets received per node during our simulations. As the packets received increases, nodes start to serve the requests of other network nodes more and consume more energy, and hence create more overhead. Since most of the packets sent in AODV routing protocol are broadcast packets, the number of packets received per node is very high compared to SCALAR, in which requests packets are sent in unicast to the selected backbone nodes. In Figure 5.5, we present the average query deepness values for only SCALAR simulation. We did not include AODV's query deepness, since we could not access this value in the implementation of AODV in our simulator. From the given figure, we can see that using *SCALAR* with replication, we achieved slightly better in terms of delay times and query deepness, as expected. The improvement with the replication case can be higher if the number of data items that every node replicates increases.

5.4 Effect of Simulation Area Node Density

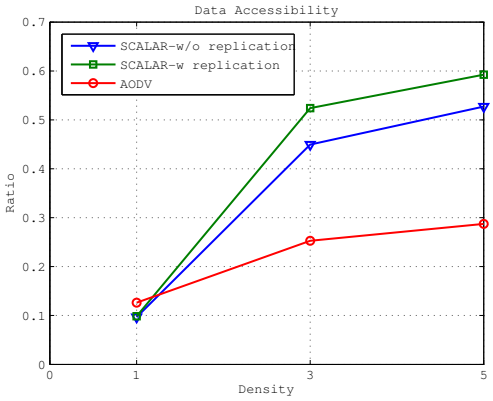


Figure 5.8: Average success ratio for increasing simulation area density levels.

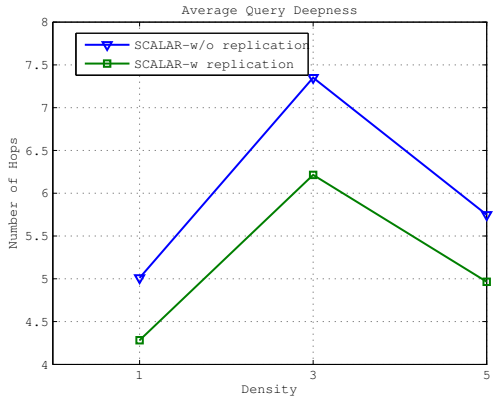


Figure 5.9: Average hop count of a completed data request for increasing densities.

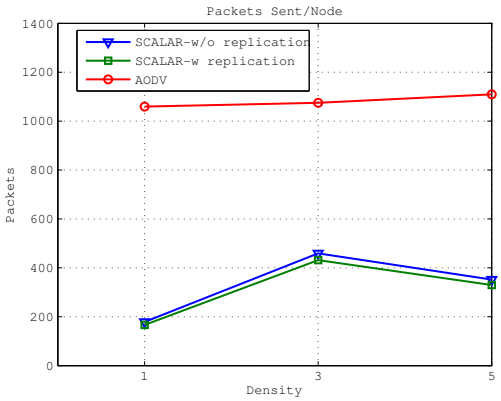


Figure 5.10: Average packets sent per node for increasing simulation densities.

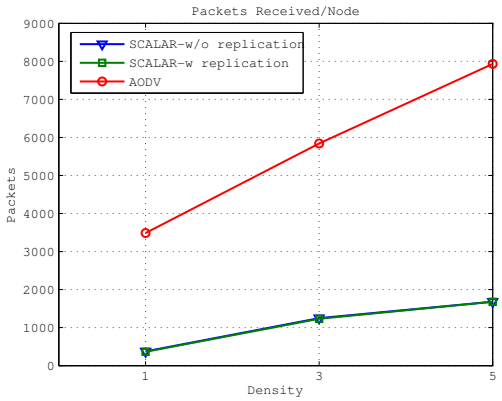


Figure 5.11: Average packets received per node for increasing simulation densities.

In Figure 5.3, we demonstrated that as the density increases the connectivity of the system increases as well. So we expect an increase in the average success ratio of nodes in the network as shown un Figure 5.8. *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. We believe that this is due to the high message overhead introduced in AODV protocol 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. Figure 5.10 and 5.11 reveals the marked difference between two approaches in terms of packets sent and received, respectively. *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), due to previously discussed reasons in virtual backbone construction phase. On the other hand, AODV keeps increasing the average packets sent per node, due to its broadcast flooding behavior. In dense networks, broadcasting causes more redundancy. In Figure 5.9, it is shown that *SCALAR* with replication achieves lower query deepness than without replication. Actually, this is the target of replication in our proposed method: to lower the query deepness of a request and hence decreasing the delay and message cost of the system.

Table 5.3: Node Density Analysis: Fixed Simulation Parameters

Simulation Parameters	
Popular Items	No
Mobility Model	R. Waypoint [1-3] m/s
Number of Nodes	200
Node Memory Size	5 (data items)
Data Request per Node	10 (requests)
Total Simulation Time	300 sec.

5.5 Effect of Node Memory Space

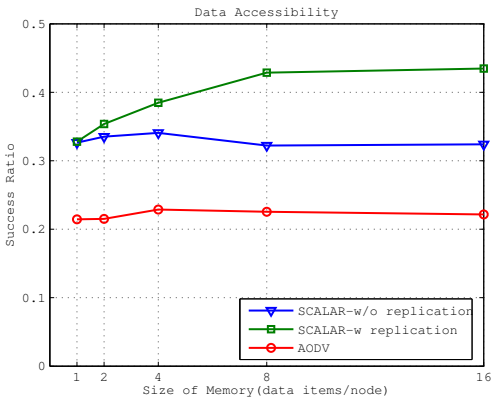


Figure 5.12: Average success ratio for increasing size of memory space of a node.

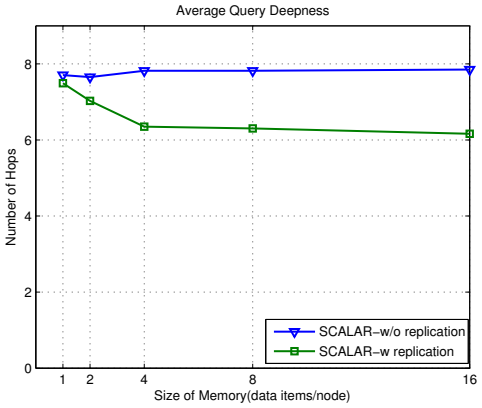


Figure 5.13: Average hop count of a completed data request for increasing size of node memory space.

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 Figure 5.12, since *SCALAR* without replication and AODV does not replicate data items, increasing the size of memory, does not affect success ratio of the simulation. However, when we used *SCALAR* with

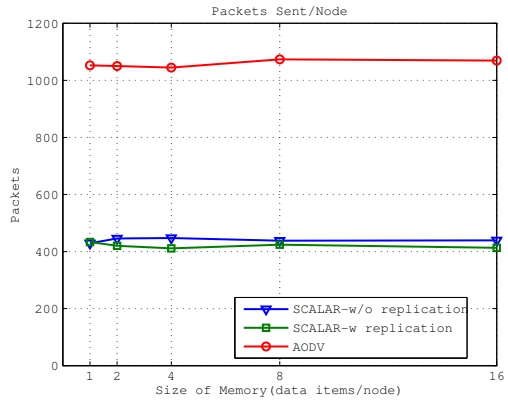


Figure 5.14: Average packets sent per node for increasing size of memory space of a node.

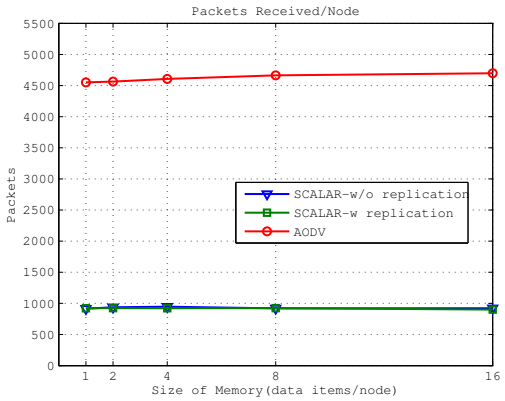


Figure 5.15: Average packets received per node for increasing size of memory space of a node.

replication we can see the increase in data accessibility as expected. Figure 5.13 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 Figure 5.12 and 5.13 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 nodes simulation, every node replicating 8 data items gives the same data variability as replicating 16 data items per node. This threshold value can be different for different node sizes, and network conditions. Optimal memory space for a specific network condition can be analyzed further and represented as an optimization problem. Figure 5.14 and 5.15 is similar to previous overhead analysis in this chapter: using AODV creates a lot of message overhead in the network. Since most of the transmitted packets are broadcast packets, packets received per node increases vastly.

5.6 Effect of Data Request per Node

In previous analysis, we fixed the number of data requests per node during the entire simulation to 10 requests. In this section, we investigate the effects of varying data

Table 5.4: Memory Space Analysis: Fixed Simulation Parameters

Simulation Parameters	
Popular Items	No
Mobility Model	R. Waypoint [1-3] m/s
Density	2
Number of Nodes	200
Data Request per Node	10 (requests)
Total Simulation Time	300 sec.

Table 5.5: Network Load Analysis: Fixed Simulation Parameters

Simulation Parameters	
Popular Items	No
Mobility Model	R. Waypoint [1-3] m/s
Density	2
Number of Nodes	200
Node Memory Size	5 (data items)
Total Simulation Time	300 sec.

requests per node (which is also called network load) on the data accessibility, average query deepness and packets sent/received. In Figure 5.16, it is shown that in low network load case, AODV performs better. When the network load is low, number of packets transmitted in the network at any given time is also low. As a result of this, packet loss due to collusion and contention in wireless channel is at minimum in the network. In this case, AODV performs better than *SCALAR*. 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 the same level

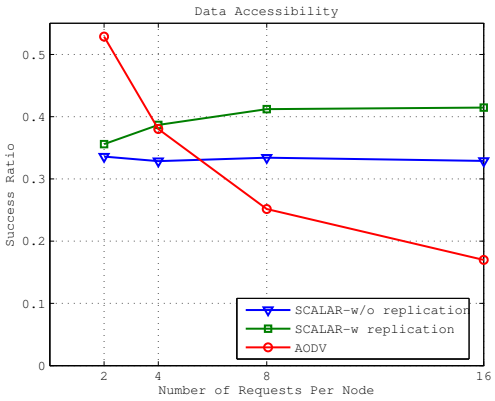


Figure 5.16: Average success ratio for increasing data request per node.

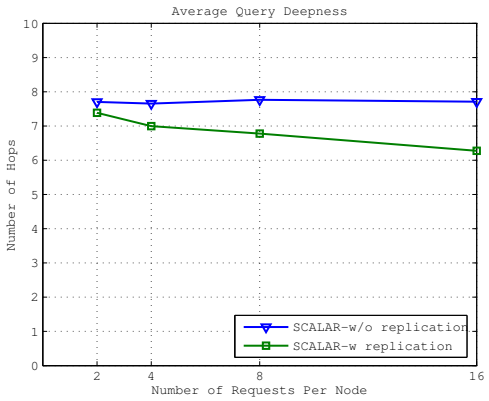


Figure 5.17: Average hop count of a completed data request for increasing data request per node.

Table 5.6: Mobility Model Parameters.

Random Walk Properties	Random Waypoint Properties
Pause Time = 1 sec.	Pause Time = 5 sec.
Fixed Radius = 1 meter	Minimum Speed = 1 m/s
Random Radius = 2 meter	Max. Speed = 3 m/s

of accessibility ratio even at higher network loads. This is an important property of *SCALAR*, which is proposed as a scalable solution for large-scale, high density and highly loaded networks.

5.7 Effect of Node Mobility Models

Previous studies [39] showed that the mobility model used for the simulation can play a significant role on the performance of a protocol. In some cases, a protocol behaves well with one mobility model used, however it can be badly affected from another one. In order to show the behavior of our framework in case of different mobility models, we implemented Random Walk mobility model on JiST/SWANS simulator.

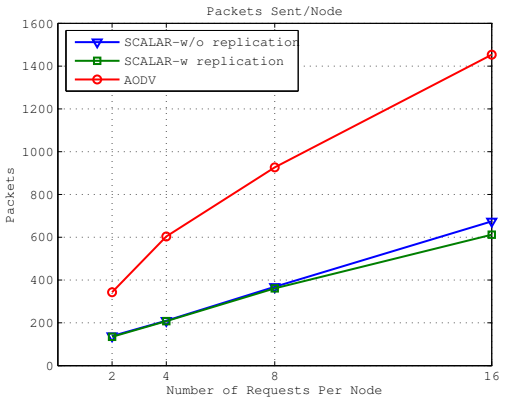


Figure 5.18: Average packets sent per node for increasing data request per node.

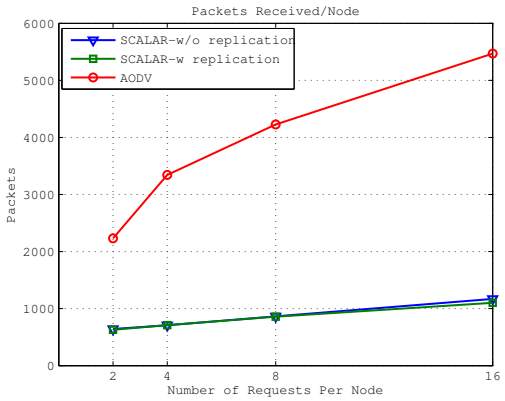


Figure 5.19: Average packets received per node for increasing data request per node.

As shown in Figure 5.20, compared to random waypoint mobility model, simulations using random walk model performed a worse in terms of data accessibility. However, in both mobility model *SCALAR* performed similar. The slight difference between two models can be explained by definitions of models. Random Walk model, by definition, does not introduce pause times for nodes. However in random waypoint model, every node waits during pause period (5 seconds in this case, see Table 5.7) 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. Details of these mobility models are given in Section 4.3.

5.8 Fairness

Results given in this section show the fairness of SCALAR among the nodes in the network. Figure 5.21 shows us that nodes with larger IDs have a higher probability of becoming a backbone node, since the virtual backbone construction algorithm that we used is tend to prune the backbone nodes with smaller IDs. Simulations are done for 100 nodes in a simulation area with node density of 2. x-axis of Figure 5.21 shows the node IDs and y-axis represents the number of times that each node become

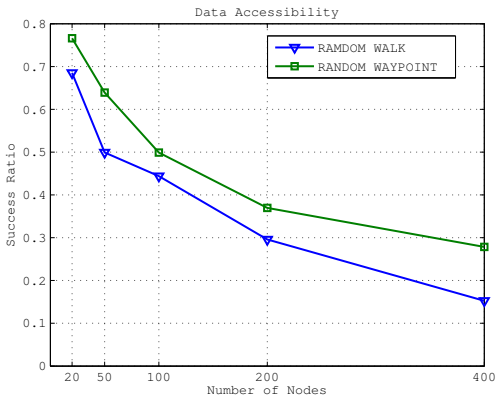


Figure 5.20: Average success ratio for simulations with different mobility models used.

a backbone node during the whole simulation time (500 seconds). Since a virtual backbone is reconstructed in every 12 seconds, a node can become a backbone node 42 times at maximum.

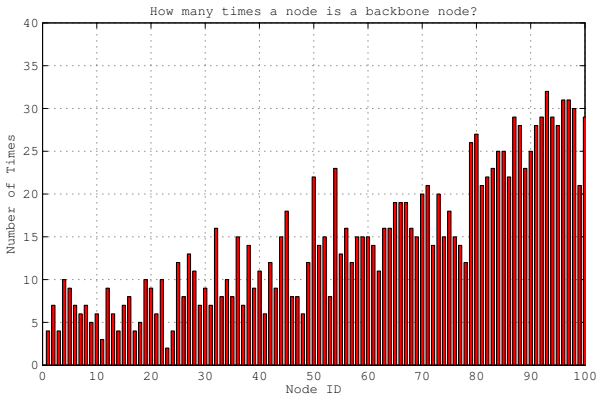


Figure 5.21: How many times node x (node id given in the x-axis) become a backbone node during whole simulation time (500 seconds)?

Figure 5.22 presents the message overhead in the form of number of packets sent per node. As mentioned earlier, on average nodes with larger IDs have a little more message overhead than nodes with smaller IDs. This due to the definition of virtual backbone construction algorithm [28] that we used in SCALAR.

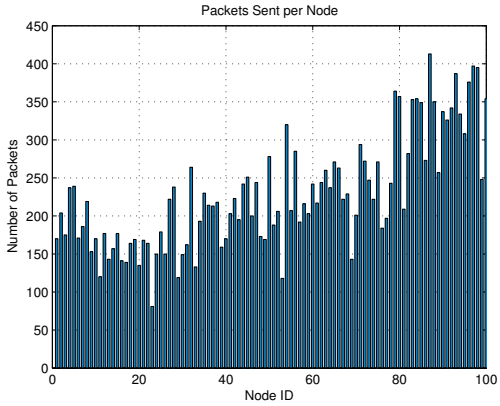


Figure 5.22: Message overhead for each 100 node in the simulation.

5.9 Comparative Results for Application Scenarios

Details of each of these scenarios are given in Section 4.4. In this section, we present the comparative simulation results of SCALAR, SAF (Static Access Frequency) and DAFN (Dynamic Access Frequency and Neighborhood) data replication approaches. DCG (Dynamic Group Connectivity) method proposed in the same study [12] is not used in our simulations due to its inapplicability to the distributed implementation in real network scenarios. These approaches basically do not provide a complete data lookup and replication solution as in our case. 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 [12]. Our implementation of SAF and DAFN replication systems is based on AODV routing algorithm to request and receive data items, since it will be less costly compared to flooding of request packets in the entire network. However, simulation results in Figure 5.23 and 5.24 show that for even smaller number of nodes, SAF and DAFN cause high message overhead and very low data accessibility. Table 5.7 summarizes the simulation parameters used in the simulation of both scenarios.

Table 5.7: Application Scenarios: Fixed Simulation Parameters

Simulation Parameters	
Popular Items	10% popular items,0.6 popular request ratio.
Mobility Model	R. Waypoint [1-3] m/s
Density	1 (Scenario 1) and 3 (Scenario2)
Node Memory Size	5 (data items)
Data Request per Node	3 (requests)
Total Simulation Time	300 sec.

5.9.1 Scenario 1 - University Campus Example

In this low node density scenario, SCALAR achieved very high success ratio when the number of nodes in the simulation is low. However, as the number of nodes increases, performance of every replication approach dropped significantly. In each number of node simulations, SCALAR outperformed SAF and DAFN replication in this scenario (Figure 5.23). It is shown in Figure 5.24 that number of packets sent per node is extremely high in DAFN due to periodic relocation of replicas.

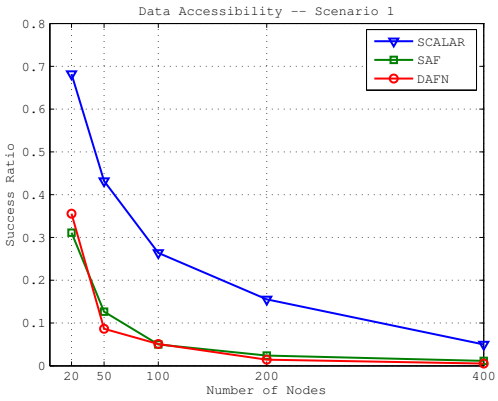


Figure 5.23: Average success ratio comparison.

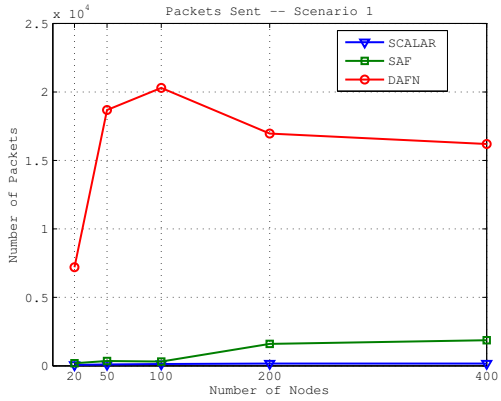


Figure 5.24: Average packets sent per node comparison.

5.9.2 Scenario 2 - Shopping Mall Example

In this high density scenario, SAF and DAFN performed similar to Scenario 1 in terms of data accessibility (Figure 5.25). On the other hand, SCALAR improved 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 collusion in the wireless channel increases. So, they cannot increase their performances even though connectivity increased.

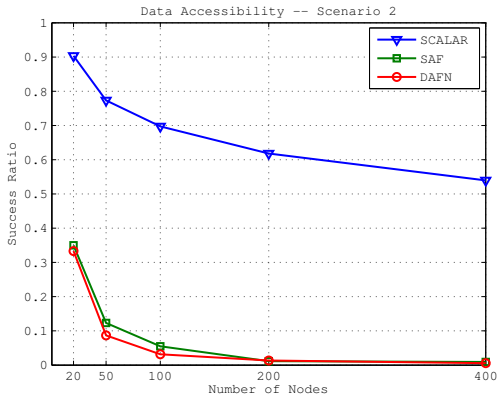


Figure 5.25: Average success ratio comparison.

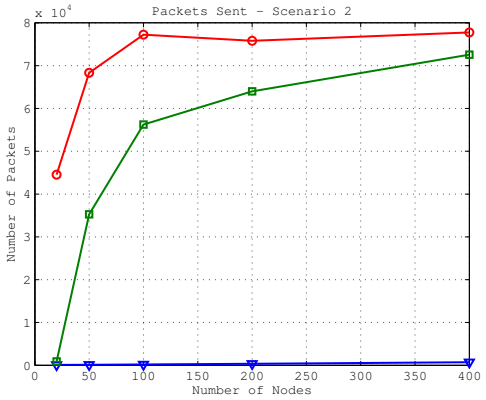


Figure 5.26: Average packets sent per node comparison.

Chapter 6

CONCLUSION

6.1 Concluding Remarks

In this thesis, we proposed 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 for 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. Each part is explained thoroughly with a theoretical message cost complexity analysis. Furthermore, we implemented proposed framework in JiST/SWANS network simulator, which is capable of simulating large scale wireless ad hoc networks. We compared the performance of SCALAR with another straightforward solution (AODV) for data lookup process in ad hoc networks. It is shown that, even in small networks, SCALAR outperformed in each of the performance metric defined. Moreover, simulation results showed that SCALAR 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 for increasing number of nodes and node density in the network. We observed that performance losses in other approaches are due to exponentially increasing message traffic caused by increasing number of nodes or density. Reactive replication approach for our scalable data lookup protocol is proposed as the third part of SCALAR framework. Proposed 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. It is called a "reactive" replication because of this

behavior. Replication decision is given based on the local access request statistics of received data item and distance of it to the requester node. Simulation results revealed that replication increases the performance of SCALAR for every performance metric defined, such as data accessibility, average delay and total message overhead. It is also shown that SCALAR do better than two of the popular data replication techniques (SAF and DAFN) in two different application scenarios. Looking at these results, 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. It is obvious that low message traffic causes low rate of packet loss in the wireless channel and in node message queues. Finally, we conclude that, existing solutions in the literature for data lookup and replication problem do not meet the requirements of large scale ad hoc network environments. We observed that this is basically because of the high message traffic created by the protocols. In SCALAR framework, we aimed to develop a lightweight lookup and replication approach, which can adapt equally well to large scale or high density network conditions.

6.2 Future Works

A possible future work on this thesis can be mathematical modeling of the protocol and network conditions in order to optimize specific node or network parameters for better performances. For example, in our simulations we realized that increasing the number of memory space for a node does not increase the performance of the protocol after a threshold value. Or in the definition of our protocol, backbone nodes forward incoming requests to two randomly selected backbone members. However, this value is not decided after a mathematical analysis; instead it is based on simulation results. So, in order to strengthen the mathematical roots of our protocols, we believe modeling of proposed protocols is necessary in the future.

In the complexity analysis of SCALAR protocols, we showed that messaging overhead of SCALAR is mostly dependent on the number of backbone nodes. For the

creation of the virtual backbone, we used a distributed implementation of Wu et al.'s connected dominating set construction algorithm. A future work might be analyzing the effects of other connected dominating set construction algorithms existing in the literature to SCALAR's performance. Since finding a minimum connected dominating set in a graph is a NP-hard problem, better approximation algorithms are expected to increase the performance of SCALAR.

It is also possible to use a modified version of our algorithm in which paths to forwarded data items are saved in the backbone nodes for the future data requests. Details of this modification and its effects to performance can be studied as a future work.

Another possible and interesting future work might be the centrality analysis of backbone nodes in the network. In one of our previous studies [46], we investigated the applicability of eigenvector centrality (EVC) principle to the replication point selection in MANETs. A similar applicability can also be investigated for reactive replication approach of SCALAR.

BIBLIOGRAPHY

[1] J. Jubin and J. Tornow, "The DARPA packet radio network protocols," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 21–32, 1987.

[2] C. Toh, *Ad Hoc Wireless Networks: Protocols and Systems*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2001.

[3] J. Mackar and S. Corson, "RFC 2501, Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," 1999.

[4] V. Gianuzzi, "Data replication effectiveness in mobile ad-hoc networks," *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 17–22, 2004.

[5] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," *Mobile Computing, IEEE Transactions on*, vol. 5, no. 1, pp. 77–89, 2006.

[6] W. Lau, M. Kumar, and S. Venkatesh, "A cooperative cache architecture in support of caching multimedia objects in MANETs," *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, pp. 56–63, 2002.

[7] B. Tang, H. Gupta, and S. Das, "Benefit-based Data Caching in Ad Hoc Networks," *Network Protocols, 2006. ICNP'06. Proceedings of the 2006 14th IEEE International Conference on*, pp. 208–217, 2006.

[8] Y. Du and S. Gupta, "COOP-A cooperative caching service in MANETs," *Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005. ICAS-ICNS 2005. Joint International Conference on*, pp. 58–58, 2005.

[9] X. Zhang and G. Riley, "Scalability of an Ad Hoc On-Demand Routing Protocol in Very Large-Scale Mobile Wireless Networks," *Simulation*, vol. 82, no. 2, pp. 131–142, 2006.

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

[11] S. Das, C. Perkins, and E. Royer, "Ad hoc on demand distance vector (AODV) routing," *Mobile Ad-hoc Network (MANET) Working Group, IETF, Jan*, vol. 81, 2002.

[12] T. Hara and S. K. Madria, "Data replication for improving data accessibility in ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 11, pp. 1515–1532, 2006.

[13] P. Padmanabhan, L. Gruenwald, A. Vallur, and M. Atiquzzaman, "A Survey Of Data Replication Techniques For Mobile Ad-hoc Network Databases," *Journal of Very Large Data Bases*, 2006.

[14] Y. Tseng, S. Ni, Y. Chen, and J. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," *Wireless Networks*, vol. 8, no. 2, pp. 153–167, 2002.

[15] P. Pabmanabhan and L. Gruenwald, "DREAM: A Data Replication Technique for Real-Time Mobile Ad-hoc Network Databases," *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pp. 134–134, 2006.

[16] A. Mondal, S. Madria, and M. Kitsuregawa, "CADRE: A Collaborative replica allocation and deallocation approach for Mobile-P2P networks," *Database Engineering and Applications Symposium, 2006. IDEAS'06. 10th International*, pp. 21–28, 2006.

[17] ——, "CLEAR: An efficient qos-based dynamic replication scheme for mobilep2p networks," *Proc. of International Conference on Database and Expert Systems Applications. DEXA*, 2006.

[18] ——, "EcoRep: An Economic Model for Efficient Dynamic Replication in Mobile-P2P networks."

[19] V. Thanedar, K. Almeroth, and E. Belding-Royer, "A lightweight content replication scheme for mobile ad hoc environments," *Lecture Notes in Computer Science*, pp. 125–136.

[20] K. Chen, S. Shah, and K. Nahrstedt, "Cross-Layer Design for Data Accessibility in Mobile Ad Hoc Networks," *Wireless Personal Communications*, vol. 21, no. 1, pp. 49–76, 2002.

[21] H. Yu, P. Martin, and H. Hassanein, "Cluster-based Replication for Large-scale Mobile Ad-hoc Networks," *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, vol. 1, 2005.

[22] K. Constantinos, "Data Management in Mobile Environments."

[23] P. Wan, K. Alzoubi, and O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks," *Mobile Networks and Applications*, vol. 9, no. 2, pp. 141–149, 2004.

[24] Y. Chen and A. Liestman, "Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks," *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pp. 165–172, 2002.

[25] J. Yu and P. Chong, "A survey of clustering schemes for mobile ad hoc networks," *Communications Surveys & Tutorials, IEEE*, vol. 7, no. 1, pp. 32–48, 2005.

[26] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks," *IEEE Transactions on Parallel and Distributed Systems,*, vol. 13, no. 1, 2002.

[27] K. Alzoubi, P. Wan, and O. Frieder, "Message-optimal connected dominating sets in mobile ad hoc networks," *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pp. 157–164, 2002.

[28] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pp. 7–14, 1999.

[29] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 85–97, 1998.

[30] M. Min, F. Wang, D. Du, and P. Pardalos, "A reliable virtual backbone scheme in mobile ad-hoc networks," *1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2004.

[31] J. Wu and F. Dai, "Virtual Backbone Construction in MANETs using Adjustable Transmission Ranges," *Mobile Computing, IEEE Transactions on*, vol. 5, no. 9, pp. 1188–1200, 2006.

[32] P. Sinha, R. Sivakumar, and V. Bharghavan, "Enhancing ad hoc routing with dynamic virtual infrastructures," *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2001.

[33] B. Clark, C. Colbourn, and D. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, no. 1-3, pp. 165–177, 1991.

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

[35] S. Guha, "Approximation Algorithms for Connected Dominating Sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.

[36] 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.

[37] N. Simulator, "ns-2," *URL–http://www. isi. edu/nsnam/ns*.

[38] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," *Workshop on Parallel and Distributed Simulation*, pp. 154–161, 1998.

[39] E. Atsan and Ö. Özkasap, "A classification and performance comparison of mobility models for ad hoc networks." in *ADHOC-NOW*, ser. Lecture Notes in Computer Science, T. Kunz and S. S. Ravi, Eds., vol. 4104. Springer, 2006, pp. 444–457.

[40] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communications & Mobile Computing*, vol. 2, pp. 483–502, 2002.

[41] M. Musolesi, S. Hailes, and C. Mascolo, "An ad hoc mobility model founded on social network theory," *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp. 20–24, 2004.

[42] I. Stepanov, P. Marron, and K. Rothermel, "Mobility Modeling of Outdoor Scenarios for MANETs," *Proceedings of the 38th Annual Simulation Symposium (ANSS'05)-Volume 00*, pp. 312–322, 2005.

[43] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *Mobile Computing, IEEE Transactions on*, vol. 2, no. 3, pp. 257–269, 2003.

[44] C. Bettstetter, "Smooth is better than sharp: a random mobility model for simulation of wireless networks," *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pp. 19–27, 2001.

[45] E. Royer, P. Melliar-Smith, and L. Moser, "An analysis of the optimum node density for ad hoc mobile networks," *Communications, 2001. ICC 2001. IEEE International Conference on*, vol. 3, 2001.

[46] E. Atsan and O. Ozkasap, "Applicability of Eigenvector Centrality Principle to Data Replication in MANETs," *International Symposium on Computer and Information Sciences, 2007 IEEE 22th*, 2007.

VITA

EMRE ATSAN was born in Mersin, Turkey on November 22, 1982. He received his B.Sc. degree in Computer Engineering from Koc University, Istanbul, in 2005. From September 2005 to September 2007, he worked as a teaching and research assistant in Koc University, Turkey and had studied on mobile ad hoc networks with Assist. Prof. Öznur Özkasap. He has published several papers for the following conferences: ADHOC-NOW06 (Ottawa, Canada), ISCIS07 (Ankara, Turkey), ICSPC07 (Dubai) and SIU07 (Eskişehir, Turkey). He has recently become a Ph. D. candidate in Computer Engineering at Bilkent University, Ankara, Turkey.