Spring 2005

# Clustering and Hybrid Routing in Mobile Ad Hoc Networks

Lan Wang
*Old Dominion University*

# CLUSTERING AND HYBRID ROUTING IN MOBILE AD HOC NETWORKS

by

Lan Wang
B.S. June 1992, Harbin Engineering University
M.S. June 1995, Harbin Engineering University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
May 2005

Approved by:

[illegible]
Stephan Olariu (Director)

[illegible]
Hussein Abdel-Wahab

[illegible]
Ravi Mukkamala

[illegible]
Ivan Stojmenovic

[illegible]
Larry Wilson

# ABSTRACT

## CLUSTERING AND HYBRID ROUTING IN MOBILE AD HOC NETWORKS

Lan Wang
Old Dominion University, 2005
Director: Dr. Stephan Olariu

This dissertation focuses on clustering and hybrid routing in Mobile Ad Hoc Networks (MANET). Specifically, we study two different network-layer virtual infrastructures proposed for MANET: the explicit *cluster* infrastructure and the implicit *zone* infrastructure. In the first part of the dissertation, we propose a novel clustering scheme based on a number of properties of *diameter-2* graphs to provide a general-purpose virtual infrastructure for MANET. Compared to virtual infrastructures with central nodes, our virtual infrastructure is more symmetric and stable, but still light-weight. In our clustering scheme, cluster initialization naturally blends into cluster maintenance, showing the unity between these two operations. We call our algorithm *tree-based* since cluster merge and split operations are performed based on a spanning tree maintained at some specific nodes. Extensive simulation results have shown the effectiveness of our clustering scheme when compared to other schemes proposed in the literature. In the second part of the dissertation, we propose TZRP (Two-Zone Routing Protocol) as a hybrid routing framework that can balance the tradeoffs between pure proactive, fuzzy proactive, and reactive routing approaches more effectively in a wide range of network conditions. In TZRP, each node maintains two zones: a Crisp Zone for proactive routing and efficient bordercasting, and a Fuzzy Zone for heuristic routing using imprecise locality information. The perimeter of the Crisp Zone is the boundary between pure proactive routing and fuzzy proactive routing, and the perimeter of the Fuzzy Zone is the boundary between proactive routing and reactive routing. By adjusting the sizes of these two zones, a reduced total routing control overhead can be achieved.

# ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Stephan Olariu, who has put a great deal of time and effort guiding me during the past four years. I also want to thank my dissertation committee members - Dr. Ivan Stojmenovic, Dr. Hussein Abdel-Wahab, Dr. Ravi Mukkamala, and Dr. Larry Wilson - for their valuable suggestions and guidance.

Many thanks to Dr. Kurt Maly and Dr. Mohammad Zubair. This dissertation cannot be finished without their financial support.

I also wish to extend my appreciation to the faculty of CS ODU and my fellow students for all the help they have given to me during my study at ODU.

Finally, I want to thank my father, mother, and brother for their encouragement and unconditional support. This dissertation is dedicated to them.

# TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

# CHAPTER I
# INTRODUCTION

A mobile ad hoc network (MANET) is a collection of mobile nodes that communicate using a wireless medium to form a rapidly deployable untethered network. In addition to attending to its own business, each node also acts as a router, forwarding packets on behalf of other nodes. Examples of MANET applications include: tactical operations, search-and-rescue missions, law enforcement, and virtual classrooms, among many others. Compared to wireline networks and to cellular networks, MANET has the following distinguishing characteristics: (1) lack of pre-existing infrastructure, (2) potential for accommodating rapid node mobility, and (3) all communications are carried over the bandwidth-constraint wireless medium. Given the dynamic network topology, decentralized control, and multi-hop connections, providing reliable end-to-end communications in MANET is a very challenging problem.

This dissertation focuses on the network-layer problems in large-scale MANET. Specifically, we study two different types of network-layer virtual infrastructures: the explicit *cluster* infrastructure and the implicit *zone* infrastructure.

In the first part of the dissertation, we propose a novel clustering scheme based on a number of properties of *diameter-2* graphs. We view our clustering scheme as the first step towards achieving a general-purpose virtual infrastructure for MANET. Compared to virtual infrastructures with central nodes, our virtual infrastructure is more *symmetric* and *stable*, but still *light-weight*. In our clustering scheme, cluster initialization naturally blends into cluster maintenance, showing the unity between these two operations. We call our algorithm *tree-based* since cluster merge and split operations are performed based on a spanning tree maintained at some specific nodes. Extensive simulation results have shown the effectiveness of our clustering scheme when compared to other schemes proposed in the literature.

In the second part of the dissertation, we focus on hybrid routing protocols for MANET. We develop a theoretical model for computing the total routing control overhead of *zone*-based routing framework, which provides a deeper insight into the power of hybrid routing. Further, we propose TZRP (Two-Zone Routing Protocol)

The journal model followed by this dissertation is *IEEE Transactions on Parallel and Distributed Systems*.

as a general hybrid routing framework that can balance the tradeoffs between pure proactive, fuzzy proactive, and reactive routing approaches more effectively in a wide range of network conditions. In contrast with the original ZRP where a single zone serves a dual purpose, TZRP aims to decouple the framework's ability to adapt to traffic pattern from the ability to adapt to mobility. In TZRP, each node maintains two zones: a Crisp Zone for proactive routing and efficient bordercasting, and a Fuzzy Zone for heuristic routing using imprecise locality information. The perimeter of the Crisp Zone is the boundary between pure proactive routing and fuzzy proactive routing, and the perimeter of the Fuzzy Zone is the boundary between proactive routing and reactive routing. By adjusting the sizes of these two zones, a reduced total routing control overhead can be achieved. The effectiveness of TZRP has been demonstrated through both detailed *ns-2* simulations and theoretical analysis.

The remainder of this dissertation is organized as follows: Chapter II provides a succinct survey of clustering schemes and hybrid routing protocols for MANET. Chapter III presents our tree-based clustering scheme. Chapter IV presents our two-zone hybrid routing framework. Chapter V offers concluding remarks and maps out directions for further investigations.

# CHAPTER II
# LITERATURE REVIEW

## II.1 A REVIEW OF CLUSTERING SCHEMES IN MANET

A significant number of clustering (*cluster initialization* and *cluster maintenance*) schemes for MANET have been proposed in various contexts. For example, at the medium access layer, clustering helps increase system capacity by promoting the spatial reuse of wireless channel [47]; at the network layer, clustering helps broadcast efficiently [78], reduce the size of routing tables [37], and strike a balance between reactive and proactive routing control overhead [49]. Although, on the surface, these clustering schemes are quite different, they can be broadly classified into two categories — *node-centric* and *cluster-centric* — depending on what is considered to be atomic. In the node-centric schemes the atomic entities are the nodes, and clustering amounts to identifying special nodes, commonly referred to as *cluster-heads*, that attract neighboring nodes into clusters. By contrast, in the cluster-centric schemes the cluster is atomic: here, clustering amounts to merging and splitting clusters to keep certain properties.

In each category, we further group schemes according to different *clustering goals*, i.e. the desirable properties of the virtual infrastructure that the clustering schemes *generate* and *maintain*. In the *node*-centric schemes, the clustering goals include *dominating set*, *maximal independent set*, *connected dominating set*, etc. In the *cluster*-centric schemes, the clustering goals include $k$-clustering, $(\alpha, t)$-clustering, $MMWN$ clustering, etc. In our discussion, we choose to focus more on the general properties of the proposed virtual infrastructures than on the optimizations targeted at specific applications since we believe that such a relatively application-independent discussion can help identify and compare the contributions and limitations of different clustering schemes more fairly and clearly in the broader context of achieving scalability in MANET.

For each clustering goal, we present a representative sample of the various approaches proposed in the literature. In particular, we are more interested in those approaches that exhibit *local* behavior. A *localized algorithm* was originally defined as a distributed computation in which nodes only communicate with nodes within some neighborhood, yet the overall computation achieves a desired global objective. In [22], a *strictly localized protocol* is defined as a localized protocol in which all information processed by a node is either: (a) local in nature (i.e. they are properties of the node itself or of its neighborhood); or (b) global in nature (i.e. they are properties of the network as a whole), but obtainable immediately (in constant time) by querying only the node's neighbors. For example, consider a protocol that builds a *spanning tree* by performing a distributed Breadth-First Search involving only local communications. Such a protocol is localized but *not* strictly localized since it takes time proportional to the *diameter* of the network and the entire network must be traversed before the spanning tree can be constructed. This definition of a *strictly localized* algorithm better characterizes the capability of a good localized algorithm to perform *independent* and *simultaneous* operations which is especially desirable for MANET. In this chapter, the *strictly localized* criterion is adopted as an important yardstick for comparing different clustering schemes.

### II.1.1 Node-centric schemes

In node-centric schemes, a subset of the network nodes is selected to perform network control functions. For example, these special nodes can work as local transmission coordinators [29]; they also naturally form a network backbone to achieve efficient broadcasting [78].

Using graph theory terminology, these nodes form a *dominating set*, *maximal independent set*, or *connected dominating set* of the network. A more precise definition of these structures follows. Consider a graph $G = (V, E)$, a subset $D$ of $V$ is a *dominating set* (DS) if each node in $V - D$ is adjacent to some node in $D$. If the subgraph induced by $D$ is connected, then $D$ is a *connected dominating set* (CDS). In general graphs, the complexity of finding a *minimum dominating set* (MDS) or a *minimum connected dominating set* (MCDS) is NP-hard. A subset $S$ of $V$ is an *independent set* (IS) if there is no edge between any pair of nodes in $S$. If no proper superset of $S$ is also an IS, then $S$ is a *maximal independent set* (MIS). Note that a MIS is a DS in which no two nodes are adjacent.

## Basic heuristics: LCA and LCA2

Baker and Ephremides propose two basic clustering heuristics — LCA [7] and LCA2 [26]. In LCA (Linked Cluster Algorithm), a node $x$ becomes a cluster-head if at least *one* of the following conditions is satisfied: (1) $x$ has the highest nodeID among all its 1-hop neighbors; (2) there exists at least one neighboring node $y$ such that $x$ is the highest ID node in $y$'s 1-hop neighborhood. The distributed implementation of the LCA heuristic terminates in $O(1)$ message rounds under the synchronous network model [48]. Amis et al. [4] generalize LCA to $d$ hops (i.e., each node in the cluster is up to $d$ hops away from the cluster-head), and the corresponding *max-min* heuristic terminates in $O(d)$ message rounds.

The LCA heuristic was revised in LCA2 to decrease the number of cluster-heads. In LCA2, a node is said to be *covered* if it is in the 1-hop neighborhood of a node that has declared itself to be a cluster-head. Starting from the lowest ID node to the highest ID node, a node declares itself to be a cluster-head if it has the lowest ID among the *un-covered* nodes in its 1-hop neighborhood. A distributed implementation of the LCA2 heuristic is described in [47]. It terminates in O(*diam*) message rounds (*diam* is the *diameter*, or strictly speaking, the *blocking diameter* [11], of the network), and each node transmits exactly one message during the execution of the algorithm.

It is interesting to compare the differences between LCA and LCA2: LCA requires the nodeIDs of both 1-hop and 2-hop neighbors, while LCA2 only requires the nodeIDs of 1-hop neighbors; on the other hand, LCA is *strictly localized*, while LCA2 is not. In addition, the cluster-heads in LCA form a DS, while the cluster-heads in LCA2 form a MIS.

Many heuristics are derived from LCA and LCA2, such as the degree-based heuristic described in [29, 54]. All of these heuristics make the implicit assumption that each node has a globally unique ID. MAC address or IP address are examples of such IDs. However, in some form of ad hoc networks, such a globally unique ID may not be available in advance. The Clubs [52] algorithm tries to do clustering in such a scenario. In Clubs, the nodes compete by choosing random numbers from a fixed integer range [0, R). Then each node counts down from that number. If it reaches zero without receiving a message, the node becomes a cluster-head and broadcasts a cluster-head declaration message. A node that hears the cluster-head declaration message before it has had the chance to declare itself a cluster-head becomes a member of the cluster-head node from which it first receives the cluster-head declaration. The Clubs algorithm takes exactly $R$ rounds to terminate. When duplicate random numbers are chosen, neighboring cluster-heads (*leadership conflict*) may happen. The expected number of *leadership conflicts* is proved to be at most $\frac{D_{avg}*N}{2R}$ ($D_{avg}$ is the average node degree, $N$ is the total number of nodes in the network).

The random count-down mechanism described in Clubs is quite similar to the CSMA/CA medium access control mechanism widely used in wireless networks. This suggests the possibility of integrating the clustering algorithm directly into MAC layer [35, 41]. Such an approach is efficient as far as the number of control messages is concerned; however, it is very inflexible since its clustering criterion is based solely on channel access.

### Maximal Independent Set

Basagni's DMAC [11] algorithm further generalizes the LCA2 heuristic by allowing the selection of cluster-heads based on a generic *weight* associated with each node (instead of using nodeID or degree), and the resulting cluster-heads form a *maximal weighted independent set.* The dynamically changing weight values are intended to express how suitable a node is for the role of cluster-head. How to calculate the weight is application-dependent, and may include factors such as transmission power, mobility, and remaining battery power, among others [9, 14, 19].

The author of DMAC also tries to generalize the algorithm so that it is suitable for *both* cluster initialization *and* maintenance. This is achieved by augmenting a similar implementation as in [47] so that each node reacts not only to the reception of a message from other nodes, but also to the breakage/formation of a link.

At any time, DMAC guarantees that the following properties are satisfied: (1) Every ordinary node has a cluster-head as its neighbor (*dominance property*); (2) Every ordinary node affiliates with the neighboring cluster-head that has the largest weight; (3) No two cluster-heads can be neighbors (*independence property*).

To enforce the above properties, DMAC requires that when a cluster-head $v$ becomes the neighbor of an ordinary node $u$ whose current cluster-head has weight smaller than $v$, $u$ has to affiliate with $v$. Furthermore, when two or more cluster-heads become neighbors, those with the smaller weights have to resign and affiliate with the now largest-weight neighboring cluster-head. A node $x$ that originally affiliated with the resigning cluster-head tries to affiliate with an existing cluster-head in its neighborhood *with a larger weight.* If such a node does not exist, $x$ becomes a cluster-head itself. This may trigger further violations of the *independence property.* In such a way, resignation of one cluster-head may cause a *rippling effect* such that some nearby cluster-heads may also have to resign. In the worst case, all the clusters in the whole network have to be reformed.

In an attempt to eliminate the global rippling effect exhibited by DMAC, in the Least Cluster Change (LCC) algorithm described in [20], an ordinary node never challenges current cluster-heads even if it has a larger weight. In G-DMAC [12], adjacent cluster-heads are allowed (hence the *independence property* is no longer enforced) and a node does not have to change its cluster even if it moves in the vicinity of a *better* cluster-head. In ARC [17], a cluster-head change occurs only when one cluster becomes a *subset* of another. These solutions greatly improve the cluster stability compared to [11]. However, a central node is still assumed in each cluster, and the dominance property of cluster-heads is always enforced.

### Connected Dominating Set

The straightforward application of CDS as network backbone (*spine*) has motivated a significant amount of research effort aiming to design efficient heuristics to achieve *small CDS.* Some approaches are based on clustering algorithms [2, 9], while others [77, 79] build CDS from scratch. We include both approaches here for completeness.

The algorithm proposed by Alzoubi et al. in [2] consists of two phases to construct a CDS: the first phase is the construction of a MIS; in the second phase, some special nodes (called *connectors*) are selected to connect the MIS nodes together. *The MIS nodes and the connector nodes jointly form the resulting CDS.* The MIS construction algorithm is essentially the same as LCA2. After the MIS construction phase, nodes exchange messages so that a cluster-head knows the nodeIDs of all the cluster-heads that are located in its 3-hop neighborhood. A cluster-head selects a connector node for all the 2-hop and 3-hop cluster-heads with higher nodeID. A selected connector node $c$ further selects a second connector to connect its selector $s$ to cluster-heads 3-hop away from $s$ and with larger nodeID than $s$. The *maintenance* of CDS involves maintaining the MIS first (similar to the maintenance algorithm in LCC), and then maintaining the connection between all MIS nodes within 3-hop distance through connector nodes. Compared to those algorithms that require a separate phase of constructing a global spanning tree as in [1], this maintenance algorithm is strictly localized, hence is more practical for mobile environment. Using the *unit-disk graph* model, [2] shows that the size of CDS maintained is within a constant factor (192) of the size of the MCDS.

In [9], Bao and Garcia-Luna-Aceves present a similar two-phase algorithm to construct a CDS. In the first phase, a *priority*-based heuristic similar to LCA is used, hence the result is a DS instead of a MIS. During the second phase, two types of connector nodes are identified: *doorways* and *gateways*. Accordingly, there are two steps in the second phase: in the *first* step, if two cluster-heads in the DS are 3-hop away and there are no other cluster-heads between them, a node with the highest *priority* on the shortest paths between the two cluster-heads is selected as a *doorway*; in the *second* step, if two cluster-heads *or* one cluster-head and one doorway node are only 2-hop away and there are no other cluster-heads between them, the node between them with the largest nodeID becomes a *gateway* to connect the two cluster-heads or the doorway and the cluster-head. After the two steps, the CDS is formed. Unlike [2] in which cluster-heads are responsible for choosing connector nodes, in [9] each node determines itself whether it becomes a connector. However, since each node only relies on 2-hop neighborhood information to make such a decision, the strictly localized algorithms described in [9] are only approximation of the proposed heuristics for determining connector nodes.

In both of the above algorithms, the approach is to first construct a basic DS, and then to add some nodes to get a CDS. The strictly localized algorithm proposed by Wu and Li [79] takes an opposite approach. The algorithm first finds a CDS and then prunes certain *redundant* nodes from the CDS. The initial CDS $U$ consists of all nodes which have at least two non-adjacent neighbors. Any node in this set is called an *intermediate node*. Two rules are proposed to eliminate redundant nodes: *Rule 1*: An intermediate node $u$ is considered as redundant if it has a neighbor in $U$ with larger ID which dominates all the neighbors of $u$. After eliminating the redundant nodes according to Rule 1, the nodes left in $U$ are called *inter-gateway* nodes. *Rule 2*: Assume that $u$, $v$, and $w$ are three inter-gateway nodes that are mutual neighbors with nodeID satisfying: $u < v$ and $u < w$. If $v$ and $w$ together dominate all the neighbors of $u$, then $u$ is considered as redundant. After eliminating the redundant nodes according to Rule 2, the nodes left in $U$ are called *gateway* nodes. These *gateway* nodes form the resulting CDS. In [77], Stojmenović *et al.* improve the above nodeID-based heuristic by using $(degree, x, y)$ as the key. Detail simulation results comparing different versions of the heuristic, as well as the cluster approach without any optimization for reducing the number of connector nodes (that is, all the nodes that have neighbors in different clusters are considered as *border nodes*) are also discussed in [77] in the context of achieving efficient network broadcasting.

Besides, both CDS and DS/MIS have been studied extensively in CEDAR [75] and its precursor Spine [74] to support QoS routing in MANET. The rationale for preferring DS/MIS to CDS in such a context is that maintaining a good-quality (small) CDS is much more expensive than maintaining a small DS/MIS in MANET [75].

#### Other node-centric schemes

Some other graph theoretic structures are also proposed as virtual infrastructures for MANET, such as *weakly-connected dominating set (WCDS)* [3, 21], *d-hop CDS* with the shortest path property [65], and *k-Tree core* [73]. [13] proposes a virtual infrastructure that imposes more constraints on a generalized MIS, i.e. the network is partitioned into a forest with a small number of trees, and the root of each tree works as cluster-head. These trees also satisfy depth, weight, and some other constraints for QoS guarantees. The algorithms proposed in the above work mainly target static ad hoc networks, hence the question of how to maintain these virtual infrastructures in response to topology changes is left open.

### II.1.2 Cluster-centric schemes

Cluster-centric schemes focus on dividing a large network into manageable sub-networks to form a hierarchical structure over which essential network control functions can be efficiently supported. For example, each cluster can be assigned a unique code to promote spatial reuse of the wireless channel [47]. Each cluster can also naturally act as unit for abstracting and propagating routing state information [6, 8, 17, 27, 37, 42, 49, 62]. In the cluster-centric schemes, there is no special node in a cluster, and each node is capable of assuming the role of logical cluster representative if necessary. Such a more *symmetric* cluster has the potential to form a more *stable* and *robust* virtual infrastructure compared to the node-centric schemes.

### $k$-clustering

$k$-clustering has been suggested by several papers [6, 27, 42]. Fernandess and Malkhi formally define *minimum k-clustering* in [27] as follows: Given $G = (V, E)$ and a positive integer $k$, find the *smallest* value of $l$ such that there is a partition of $V$ into $l$ disjoint subsets, and the diameter of the graph induced by each subset is not larger than $k$. *k-clustering* is NP-hard for general graphs.

A cluster initialization algorithm forming diameter-$k$ clusters is presented in [27]. The algorithm works in two stages: in the first stage, a spanning tree of the network is constructed using the MCDS approximation algorithm in [1] (which works in two stages itself); in the second stage, the spanning tree is partitioned into sub-trees with bounded diameter. How to maintain such a diameter-$k$ cluster in the face of mobility is not discussed in [27]. In [42], forming and maintaining diameter-1 (*clique*) clusters is discussed in the context of MANET routing.

There are also several clustering schemes imposing *implicit* constraints on cluster diameter, such as the ($\alpha$,t)-clustering [49, 50] and *MMWN* [8, 62] discussed below.

### ($\alpha$,t)-clustering

The objective of the ($\alpha$,t)-clustering framework [49] is to maintain an effective virtual infrastructure that adapts to node mobility so that a hybrid routing protocol can be adopted to balance the tradeoff between proactive and reactive routing control overhead according to the temporal and spatial dynamics of the network. Specifically, the ($\alpha$,t)-clustering framework dynamically organizes mobile nodes into clusters in which the probability of path availability ($\alpha$) can be bounded for a period of time ($t$). Since $\alpha$ establishes a lower bound on the probability that a given cluster path will remain available for time $t$, it controls the cluster's inherent stability. For a given $\alpha$ (stability level), the role of $t$ is to manage the cluster size, which controls the balance between routing optimality and efficiency.

However, the definition of ($\alpha$,t)-cluster needs to be refined for working effectively in a general MANET. Note that the $(\alpha, t)$*-reachable* relation is *not* transitive. This, together with the fact that $(\alpha, t)$-clusters do not overlap, implies that two nodes that are relatively stable with each other are not necessarily affiliated within the same cluster, defeating the ultimate goal of $(\alpha, t)$-clustering. Indeed, the values of $\alpha$ and $t$ are crucial for the effectiveness of the protocol, and the optimum values depend on the mobility pattern of nodes in the network. How to determine such values is not discussed in [49].

Besides, implementing the cluster maintenance algorithm described in [49] is not an easy task. Consider the following scenario, when a node $X$ detects that a cluster member $Y$ is connected within the cluster, but not $(\alpha, t)$-reachable, $X$ will voluntarily leave the cluster. However, it is possible that $Y$ detects the same situation simultaneously and also voluntarily leaves the cluster. Obviously, this is not an optimal behavior. Even worse, the leaving of nodes will further trigger the $(\alpha, t)$-unreachability of the other nodes that still stay in the original cluster. Hence a series of leaving events may happen, leading to single-node clusters, which further triggers node joining. This example clearly illustrates the potential convergence problem of an $(\alpha, t)$-cluster, especially when considering the mobile nature of MANET.

McDonald and Znati [50] later propose two major modifications to the original ($\alpha$,t)-clustering framework to address the above problems: (a) The *pairwise* ($\alpha$,t)-reachability in an ($\alpha$,t)-cluster is considered too restrictive, hence the cluster definition is revised so that ($\alpha$,t)-reachability is *only* required between a *potential joining* node and the *parent node* of the cluster; (b) A node does not leave a cluster until the cluster becomes *disconnected*.

### Multimedia Support for Wireless Network System (MMWN)

MMWN [62] presents a hierarchical routing scheme designed for multimedia support in large ad hoc networks. In MMWN, cluster plays a central role in aggregating QoS routing information and limiting the propagation of topology changes.

The centralized cluster initialization algorithm described in [62] uses global link-state information and *recursive* bisection to produce connected clusters within prescribed *size* limit. In each cluster, a single node, the *cluster leader*, performs cluster split and merge to keep clusters within the size bounds as nodes move.

Based on the MMWN framework, [8] proposes a centralized cluster initialization algorithm that can generate clusters with the following desired properties: *(a)* Each cluster is *connected*; *(b)* All clusters should have a min and max *size constraint*; *(c)* A node belongs to a *constant* number of clusters; *(d)* Two clusters should have *low* overlap; *(e)* Clusters should be *stable* across node mobility. The distributed implementation of the centralized algorithm involves creating a *Breadth-First Search (BFS)* tree and traversing the tree in *post order*.

Cluster maintenance is also considered in [8]: (1) New node *joins* can cause the violation of property *(b)* and *(c)*. If *(b)* is violated, the above spanning-tree based clustering algorithm is executed on the current *cluster*; if *(c)* is violated, the clustering algorithm is executed on the *whole network*, hence *not* strictly localized. (2) Existing node *leaves* may cause the violation of *(b)*, hence the nodes in the smaller clusters must *join* some other cluster. (3) A link breakage may split the cluster into disconnected components, hence is equivalent to the scenario where an existing node *leaves*.

### II.1.3 Comparing node-centric and cluster-centric schemes

The previous subsections have shown that there exists a huge variety of clustering schemes in the literature, each with specific properties. Bettstetter and Kraussser [14] propose several general performance metrics that can be used to analyze and compare these significantly different schemes. The major metric proposed is the *stability* of the cluster infrastructure. Indeed, a good clustering algorithm should be designed to maintain its cluster infrastructure as stable as possible while the topology changes [47]. Other proposed metrics include control overhead, level of adaptiveness, convergence time, required neighbor knowledge, etc.

It is important to point out that in the series of CDS algorithms for efficient broadcasting we discussed in Subsection II.1.1, the major performance metric used to compare different algorithms is the *ratio of nodes in the resulting CDS*. Note that this performance metric does not reflect anything about the *stability* of the CDS. Such a discrepancy in the performance evaluation criteria used again reflects the fact that a virtual infrastructure that can be exploited by and optimized for a *specific* purpose does not necessarily mean a good *general-purpose* virtual infrastructure.

Generally speaking, one advantage of the node-centric schemes is that clusterheads (and connectors) naturally form a network backbone that can be exploited for broadcasting and activity scheduling [78]. However, constraining all traffic to traverse such special nodes may reduce the throughput and is likely to impact the robustness of the network since the cluster-heads can easily become *traffic bottlenecks* and *single points of failure* [72]. On the other hand, the *cluster-centric* scheme organizes the network into clusters that need not contain a cluster-head. In this scenario, each node can potentially be the logical representative of the cluster, and different nodes can work as cluster representatives for different applications. In MANET where topology changes occur frequently, this implies a potentially more stable general-purpose infrastructure that can be leveraged by a multitude of applications without introducing traffic bottlenecks and single points of failure. Some of the most important differences between the various virtual infrastructures as well as the corresponding clustering schemes are summarized in Table 1.

## II.2 A REVIEW OF HYBRID ROUTING PROTOCOLS IN MANET

Numerous routing algorithms targeted at small-to-medium MANET have been proposed in the literature, aiming to achieve good performance in terms of high throughput, low control overhead, short delay, low energy consumption, scalability, etc. Traditionally, MANET routing protocols are classified as either *proactive* (such as DSDV [60], OLSR [38], and STAR [28]) or *reactive* (such as DSR [39] and AODV [61]).

Both proactive and reactive routing protocols have their advantages and disadvantages:

1. In terms of *routing table size*, a proactive protocol has to maintain entries for all the nodes in the network, hence cannot scale well to large networks. By contrast, routing information to only active communicating nodes is maintained in a reactive routing protocol.

2. In terms of *delay*, proactive protocols have a route to the destination readily available whenever it is needed, while reactive protocols suffer from longer route acquisition latency due to the on-demand route discovery.

3. In terms of *bandwidth consumption*, reactive routing protocols are generally considered to have lower control overhead. However, when new routes have to be found frequently, the flooding of RREQ (route request) may cause significant overhead. In addition, a path is used as long as it is valid, hence route optimality cannot be achieved in such protocols. This means that the amount

TABLE 1
A Comparison of Virtual Infrastructures and Clustering Schemes

| VI | Properties | Symmetry | Adaptivity | Clustering Scheme |
|---|---|---|---|---|
| LCA [7], LCA2 [26], DMAC [11], LCC [20], ARC [17] | cluster-heads form a DS or MIS | a central node | max number of neighboring cluster-heads | node-centric; strictly localized maintenance only *1-hop* info |
| Max-min *d*-clustering [4, 5] | cluster-heads form a *d*-hop DS | a central node (*d*-hop) in each cluster | *d* | node-centric; a node needs to maintain *2d*-hop info; maintenance by periodical initialization |
| Network backbone [2, 9, 77, 78, 79] | cluster-heads and connectors form a small CDS | with a central node | ratio of CDS nodes | node-centric; strictly localized maintenance requiring *2-hop* info; some need partial 3-hop info for smaller CDS |
| *k*-clustering [6, 27, 42] | upper bound (*k*) on cluster diameter | symmetric | *k* | cluster-centric; centralized initialization (and *non*-strictly localized distributed implementations) |
| *($\alpha$,t)*-clustering [49, 50] | lower bound on intra-cluster path availability | symmetric | $\alpha$ and $t$ | cluster-centric; pairwise intra-cluster *($\alpha$,t)*-reachability difficult to maintain |
| *MMWN*-clustering [8, 62] | lower/upper bounds on cluster size; number of hierarchy levels | symmetric | cluster size | cluster-centric; centralized initialization (*non*-strictly localized implementation; maintenance assumes complete cluster topology |

of bandwidth wasted due to the sub-optimality of routes may become excessive when the call-to-mobility ratio is high. On the other hand, as demonstrated by STAR [28], by relaxing the route optimality and the consistent view constraint, proactive protocols can potentially be designed with the same level of control overhead as reactive protocols. In a sense, this flexibility of balancing the tradeoff between routing control overhead and path optimality is an advantage of proactive approaches over reactive ones.

The emerging consensus [16, 25] is that no single proactive or reactive routing protocol operates efficiently under all network conditions. Considering diverse MANET applications where mobility, traffic, network size, and node density may vary significantly, different choices and tradeoffs have to be made in different situations. An ideal routing protocol should be able to combine the strengths of both proactive and reactive protocols and to adapt its behavior at the appropriate time and for the appropriate scope of the network. This motivates the study of *hybrid* MANET routing protocols.

Ideally, a hybrid routing protocol should have the following properties:

1. *efficient*: the protocol should choose suitable basic components and should integrate them organically to achieve better performance than any single component.

2. *adaptive*: the protocol should be able to dynamically adjust the contribution of each component to achieve different performance goals under different network conditions; such adaptation mechanisms generally require a clear mapping between performance metrics and hybridization parameters.

3. *simple*: the hybridization should be light-weight, avoiding excessive control overhead.

There is a large design space for hybridization between various basic proactive and reactive protocols, and many hybrid MANET routing schemes have been proposed in the literature. These schemes can be classified into *cluster-based* and *zone-based*. In the *cluster-cluster* schemes [37, 51, 72], explicit clusters are formed and maintained as efficient control structures for abstracting and propagating routing information, and the boundary of clusters is the switching point between different routing strategies.

By contrast, in the *node-centric* routing schemes, each node makes use of an implicit control structure that is naturally associated with itself: the area that consists of all the nodes reachable in $k$ hops from it (i.e. its $k$-hop neighborhood). Such a structure is constructed and maintained as a by-product of exchanging regular routing information among nodes, and can be considered to be an *implicit* cluster.

### II.2.1 Cluster-centric hybrid routing protocols

A natural way of implementing hybrid routing is to organize the network into a hierarchy of node groups – clusters – and to adopt different routing strategies for intra- and inter-cluster traffic, respectively. Indeed, hierarchically organizing a network is a well-studied problem in large-scale wireline networks and has been shown to be effective in minimizing the size of routing tables, thus optimizing the use of network resources. In the case of MANET, partitioning nodes into clusters can have other benefits as well: the clustering control structure not only makes a large network seem smaller but, more importantly, can make a highly dynamic network appear less dynamic, essentially hiding mobility.

As discussed in Section II.1, many clustering algorithms are proposed for MANET [37, 49, 51, 72]. Among them, the $(\alpha, t)$ – *cluster* framework of [49, 51] is directly targeted at maintaining an effective cluster topology that adapts to node *mobility* in order to achieve a hybrid routing scheme that balances the tradeoff between proactive and reactive strategies according to the temporal and spatial dynamics of the network. Routing is achieved utilizing a dynamic two-level hierarchical strategy consisting of pure proactive routing (DSDV or OLSR) and least-overhead proactive routing (STAR) operating at each level. Each node maintains two routing tables. The level-1 routing table consists of one entry for each destination node within the same cluster and one entry for each neighboring cluster, indicating the next-hop nodeID along the optimal path to the corresponding destination. The level-2 routing table consists of one entry for each cluster in the network, indicating the next clusterID along the current active path toward the corresponding destination cluster, which can be resolved to a next-hop nodeID using the level-1 routing table. The level-2 protocol requires that one node (the node with the lowest nodeID in each cluster) generate an update on behalf of its cluster. When a level-2 update is generated, it is flooded to all the nodes in each neighboring cluster, but not transmitted beyond neighboring clusters. Based on STAR, every node maintains level-2 topology information.

To forward a packet to the desired destination, a source node must first use a *location management* protocol to discover the current clusterID associated with the destination node. This binding procedure is similar to a reactive route discovery process, and the associated overhead is a common problem to all the cluster-centric routing schemes. In the $(\alpha, t)-cluster$ framework, the level-2 information maintained is used to infer a broadcast tree to forward a request to every cluster only once. Furthermore, each request need only be processed by one node in each intermediate cluster, and if the target is discovered along a given subtree, early termination of the query thread on that subtree is easily achieved. Finally, the request provides binding information directly to the target of the request. Consequently, the response can be sent directly to the source of the request via unicast routing.

The $(\alpha, t) - cluster$ framework clearly demonstrates the benefits and challenges of a cluster-centric hybrid routing protocol. Generally, the hierarchical clustered MANET forces a tight coupling between routing and clustering. It is a very challenging task to determine which combination of routing and clustering algorithms is the most appropriate for a particular network.

### II.2.2 Fisheye routing and FSLS

The *fisheye* [40] routing concept is based on the observation that nodes do not need to have complete topological information in order to make a good next hop decision to reach a far away destination. Given an approximate view of the distant parts of the network, a node can forward a packet in the *proper* direction toward the destination. As the packet makes progress toward the destination, the view of the destination's region becomes more accurate, providing for more precise packet forwarding. This suggests that propagating every LSU (Link State Update) over the entire network may not be necessary. The fisheye technique is used in FSR [30, 37] and DREAM [10] (using location information provided by GPS). This class of approaches is further generalized and analyzed in FSLS [69]. In FSLS, a reduction of control overhead is achieved both in space (by limiting the scope a LSU is transmitted to) and in time (by limiting the interval between successive LSU generations). Specifically, a node wakes up every $t_e$ seconds, and transmits a LSU with TTL set to $s_i$ if the current time is $2^{i-1} * t_e * \{1, 3, 5, 7, 9, ...\}$ ($i$ is positive integer) seconds and there has been a link state change in the last $2^{i-1} * t_e$ seconds.

The choice of a good set of values $\{s_i\}$ is determined by the traffic pattern. Assuming a *uniform* traffic distribution among all nodes in the network, the values of $s_i$ that achieve the best balance between proactive overhead and route sub-optimality is derived in HSLS [69]. In these *fuzzy* proactive protocols there is a higher chance for short-term loops caused by routing inconsistencies due to different local views of the network at different nodes. To the best of our knowledge, there is no mechanism in FSLS to avoid such loops: they are detected and removed by means of TTL expiration [70].

### II.2.3 ZRP

The Zone Routing Protocol (ZRP) [31, 58] provides a hybrid routing framework that is locally proactive and globally reactive. The goal is to minimize the sum of the proactive and reactive control overhead. In ZRP, a node proactively propagates LSUs to all the nodes in its $k$-hop neighborhood, where $k$ is called *Zone Radius(ZR)*. Thus, each node has an up-to-date view of its *routing zone*, that is, all the nodes and links in the node's $k$-hop neighborhood. The routing zone nodes that are at a distance of exactly $k$ hops are called *peripheral nodes*. Each node has its own associated routing zone (hence, its own set of peripheral nodes), and routing zones of neighboring nodes overlap heavily.

ZRP is hybrid not only in that it adopts pure proactive routing for intra-zone traffic and reactive routing for inter-zone traffic but, more importantly, because the zone structure maintained for proactive routing is exploited in the reactive routing procedure through a mechanism called *bordercasting*. Rather than blindly broadcasting a node's RREQ to all its neighbors, bordercasting directs the request to peripheral nodes only.

Using the zone topology maintained, each peripheral node decides whether to reply to the request or to further forward it to its own peripheral nodes. The heavily overlapping neighboring routing zones can lead to query duplication and backward propagation. To alleviate the problem, special query control mechanisms (*Query Detection* and *Early Termination*) [31] are used to identify those peripheral nodes that have been covered by the route query (i.e. that belong to the routing zone of a node that has already bordercast the query) and to prune them from the bordercast tree. This encourages the query to propagate outward, away from its source and away from covered regions of the network.

The latest version of bordercasting [32] works as follows. When a node receives the first copy of a RREQ,

1. if the node is *not* an intended recipient of the RREQ, it is implied that the node's own routing zone has been covered by other bordercasting nodes. Thus, the node marks its entire routing zone as *covered* and discards the RREQ.

2. if the node is an intended recipient of the request, it proceeds to process the RREQ: if the node knows a route to the destination, it forwards the RREQ to the destination; otherwise, the node forwards the RREQ to those 1-hop neighbors that span its *uncovered* peripheral nodes in its bordercast tree. After forwarding the RREQ, the node marks its entire routing zone as *covered.*

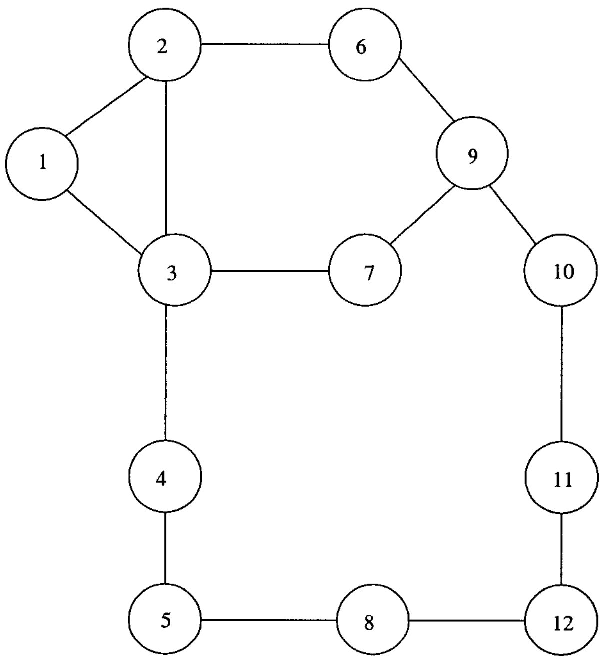


Fig. 1. Bordercasting in ZRP.

The *efficiency* of bordercasting, in terms of the number of forwarding nodes compared to flooding, and its *effectiveness*, in terms of the query success ratio compared to flooding, depend on the traffic pattern and the instantaneous network topology.

Consider the network topology shown in Figure 1, and assume $ZR = 2$. For a query from node 1 to node 9, the RREQ is terminated at nodes 2 and 3 because both 2 and 3 have the destination inside their routing zone. In this case, most of the nodes in the network (to wit, nodes 4–12) are not involved in the propagation of the RREQ. However, when node 2 wants to find a route to node 8, the RREQ propagation involves most of the nodes in the network (except nodes 5, 8, 12) before a route is found.

The optimal zone radius value is dynamically adjusted using *Min Searching* and *Traffic Adaptive Estimation* [58]. For example, if the ratio of proactive overhead to reactive overhead during a certain time interval exceeds a certain threshold, the zone radius is reduced; if the ratio is lower than a certain threshold, the zone radius is increased. By adjusting the *globally-uniform* zone radius, a good balance between proactive and reactive control overhead can be achieved and the total routing control overhead is minimized.

In recent work within the ZRP framework [67], it is argued that using a uniform zone radius throughout the whole network is sub-optimal. Instead, having independent zone radii allows each node to distributively and automatically configure its optimal zone radius, hence performance fine tuning can be achieved. However, in this Independent Zone Routing (IZR) protocol, each node has to know which nodes have a demand for its LSU. In fact, exchanging explicit control messages such as *Zone Building Packet* makes IZR more similar to the cluster-centric approach, and ZRP's simplicity due to the circular-shape zone structure and the implicit zone membership/structure maintenance ability by LSU exchanges is compromised. The tradeoff between the overhead and the benefits of the IZR scheme needs to be further investigated.

### II.2.4 CARD

Contact-based architecture for Resource Discovery (CARD) [34] is proposed as a framework for resource discovery in large-scale MANET. In the context of routing, CARD is targeted at applications in which most of the traffic consists of short flows and small transactions [34]. In such applications, the cost of discovering routes is usually the dominant factor instead of the data transfer as in long flows. As a result, CARD strives to minimize the control overhead during route discovery instead of finding and maintaining shortest paths.

In CARD, each node maintains proactively routing information in its $R$-hop neighborhood (called *vicinity*) and keeps track of all the nodes that are exactly $R$ hops away from itself (called *edge nodes*). The vicinity and edge nodes are similar to ZRP's zone and peripheral nodes, respectively. The key difference is that in addition to the above information, each node maintains paths to a few distant nodes called *contacts*. The underlying motivation is that, based on the *small world* concept, these contacts can help find a route to distant destinations more efficiently.

The selection and maintenance of contacts is the key mechanism of the CARD framework. Theoretically, each node should maintain as few contacts as possible to cover as many nodes outside of the source node's vicinity as possible. This is, to some extent, equivalent to the source-dependent minimum $k$-dominating set problem (a $k$-dominating set is a subset $D$ such that each node is within $k$-hops of a node in $D$). In practice, several heuristics are proposed to provide maximum increase in reachability with the addition of each new contact by minimizing the overlap between contacts.

A source node $s$ selects its contacts one by one. To select a contact, $s$ sends a *Contact Selection Query* (*CSQ*) control packet to one of its edge nodes. The edge node further forwards the packet to a randomly chosen neighbor. The receiving node decides whether to become a contact for $s$ by checking for overlap with $s$'s vicinity, the vicinities of all the already selected contacts and the vicinities of $s$'s edge nodes. If there is no overlap, then the node is selected as a contact. If the node fails to become a contact, it further forwards the control packet. If the packet reaches a node whose distance to $s$ exceeds a predetermined *Maximum Contact Distance* ($r$), the packet is returned to the last sender (*backtracking*). A contact is searched in such a depth-first way until one is found. Note that since searching a contact may involve backtracking, and multiple contacts are searched sequentially, this contact selection phase tends to be time-consuming.

Once a contact $c$ is selected and the route from $s$ to $c$ is established, this route has to be validated periodically. If the route is broken, local recovery is used to try to salvage it. If salvage fails or the length of this route exceeds a certain threshold, the contact is considered lost. If the number of contacts falls below a certain threshold, new contacts are selected.

When the source node *s* needs a path to destination *d*, it first checks whether *d* exists in the vicinity. If not, *s* sends a *Destination Search Query* (*DSQ*) control packet to its contacts. The *Depth of Search* (*D*) field in *DSQ* controls the levels of contacts queried. By performing such a sequential expanding ring search, CARD avoids the complicated query control mechanisms as in ZRP. The tradeoff is a longer route acquisition latency (when the destination is far away) than the already long delay in the reactive route search approach. In addition, there is a tradeoff between the maintenance control overhead and the number of contacts.

CARD provides a wide range of adjustable parameters to achieve fine tuning for various desired performance goals. However, determining and adjusting the optimum values and combinations of vicinity size, number of contacts, maximum contact distance, and maximum depth of search is a challenging problem.

Another hybrid routing protocol based on the *small world* concept is described in [15] in the context of *position-based routing* [76].

### II.2.5 LANMAR, Netmark, and SHARP

ZRP and CARD make no special assumptions about individual nodes. However, in many practical applications some nodes enjoy special properties that happen to be relevant to routing.

LANMAR [59] is designed for MANET that exhibit *group mobility*. A landmark node is selected for a group of nodes that are likely to move together. A *scope* is defined such that each node would typically be within the scope of its landmark node. Each node propagates link state information corresponding only to nodes within its scope, and distance vector information for all landmark nodes. When a node needs to send a packet to a destination within its scope, the local link state routing table is used directly. Otherwise, the packet will be routed toward the landmark nodes of the destination. When the packet arrives within the scope of the destination, it is routed using local link state tables, without necessarily going through the landmark node.

In [66], a node-centric hybrid routing protocol is proposed based on the assumption that some special nodes in a MANET are more *popular* than others. In this protocol, a *hot-spot* node is called a *netmark*. Paths between netmarks and ordinary nodes are maintained proactively, while routes between ordinary nodes are set up on demand. SHARP [64] is also predicated on the existence of *hot-spot* nodes. A *proactive zone* is defined around each hot-spot node. Nodes within the proactive zone maintain routes proactively only to the central node $x$. The nodes that are not in the proactive zone of a given destination use the reactive component (AODV with the optimization mechanisms of route caching and expanding ring search) to establish routes to that node. It is interesting to note that SHARP's proactive zone is far more light-weight than ZRP's routing zone. The proactive component of SHARP is adapted from TORA [56]. The idea is that in the proactive zone centered at node $x$, a directed acyclic graph (DAG) that is rooted at $x$ and is consisted of all the nodes in the proactive zone is built and maintained constantly. The proactive component has two procedures: *DAG construction* and *DAG update.* During *DAG construction*, the center node sends a construction control packet, which is further forwarded by the other nodes in the proactive zone. During this forwarding process, each node is assigned a *height* value. The height is the distance of the node from the center. A data packet arriving at a node is transmitted along the downstream link to the neighbor with the lowest height. During *DAG update*, with link failures, as long as there is a downstream link, a node does not take any specific action since a route to the center is still available (although not necessarily the shortest). Only when all the downstream links at a node have failed, the node reverses the orientation of its upstream links by choosing a new height greater than the height of all its neighbors and broadcasting a new update control packet. Each node receiving this update packet records the new height of this neighbor and, if necessary, adjusts the orientation of its own upstream links and initiates a new update control packet. Compared to DAG construction, the DAG update procedure introduces less control overhead. However, with the movement of nodes, the routes maintained may deviate significantly from the shortest ones, and may have to involve nodes that just moved into the proactive zone, especially in the less dense networks. To deal with these situations, the DAG construction procedure has to be invoked periodically. The more frequently the DAG construction procedure is invoked, the more proactively shortest paths are maintained at the expense of more control overhead.

Each node continually monitors network characteristics including average lifetime of immediate links and average node degree. This information is sent to the destination node periodically. The destination node also locally maintains statistics about the data traffic that it has received. Using this information, each destination independently computes the optimal proactive zone radius to bound routing control overhead, loss rate or delay jitter.

### II.2.6 Comparing different hybrid routing protocols

In this section we have reviewed various hybrid routing schemes proposed in the recent literature, focusing on their motivations, various explicit/implicit structures maintained, choices of basic components, and hybrid routing methodologies.

To recap, in cluster-centric protocols, explicit clusters are formed and maintained as routing units. The clustering constraint includes node locality and group mobility. Creating and maintaining such clusters generally involve significant overhead in the face of mobility. By contrast, node-centric approaches can provide some extent of scalability without involving too much overhead. However the lack of explicit control structures may lead to inefficiency for abstracting and propagating routing states.

We have to point out that there are no fundamental differences between these protocols. For example, CARD can be considered as generalization of LANMAR or Netmark if mobility-group leaders or hot-spot nodes are chosen to be contacts. LANMAR can be considered a special case of either Netmark (in a small network) or $(\alpha, t) - \mathit{clustering}$ (in a large network with group mobility). This similarity suggests the possibility of further hybridizations between these protocols.

## II.3 SUMMARY

To set the stage for discussing our work on clustering and routing in MANET, in this Chapter, we have reviewed a number of clustering schemes and hybrid MANET routing protocols in the literature.

Many clustering schemes have proposed to provide different virtual infrastructures for MANET. Such a diversity in the resultant virtual infrastuctures reflects the plethora of different MANET applications. This in turn calls for a *general-purpose* virtual infrastructure that can be effectively leveraged by a multitude of applications, motivating our tree-based clustering scheme as presented in Chapter III.

In MANET routing, by integrating suitable proactive and reactive components to adapt to changing network conditions, a hybrid protocol can provide better performance than any basic protocol. In Chapter IV, we present TZRP as a hybrid routing framework that can balance the tradeoff among various routing approaches effectively in a wide range of network conditions.

# CHAPTER III

# A TREE-BASED CLUSTERING SCHEME FOR MANET

Since flat networks do not scale, it is a time-honored strategy to overlay a virtual infrastructure on a physical network. There are, essentially, two approaches to doing this. The first approach is protocol-driven and involves crafting a virtual infrastructure in support of whatever protocol happens to be of immediate interest. While the resulting virtual infrastructure is likely to serve the protocol well, more often than not, the infrastructure is not useful for other purposes. This is unfortunate, as its consequence is that a new infrastructure has to be invented and installed from scratch for each individual protocol in a given suite. In bandwidth-constraint MANET, maintaining different virtual infrastructures for different protocols may involve excessive overhead.

The alternate approach is to design the virtual infrastructure with no particular protocol in mind. The challenge, of course, is to design the virtual infrastructure in such a way that it can be leveraged by a *multitude* of different protocols. Such a virtual infrastructure is called *general-purpose* as opposed to *special-purpose* if it is designed in support of just one protocol. The benefits of a general-purpose virtual infrastructure are obvious.

To the best of our knowledge, research studies addressing MANET have, thus far, taken only the first approach. Indeed, an amazing array of special-purpose virtual infrastructures have been proposed in support of various sorts of protocols but only a few of them may have the potential of becoming general-purpose. Our point is that the important problem of identifying general-purpose infrastructures that can be leveraged by a multitude of different protocols has not yet been addressed in MANET.

We view the main contribution of the work in this chapter as the first step in this direction. Specifically, we identify *clustering* as the archetypal candidate for establishing a general-purpose virtual infrastructure for MANET. As shown in the survey of various MANET clustering schemes in Section II.1, most of these schemes are designed for some specific purposes, and the resulting virtual infrastructures may not be reused effectively by the other applications. For example, in clusters predicated on the existence of a centrally-placed cluster-head, such a central node can easily become a communication bottleneck and a single point of failure. Consequently, the resulting virtual infrastructure is not suitable for a number of important network control functions including routing [72] and security.

Motivated by the idea that a virtual infrastructure having a decent chance of becoming truly general-purpose should be able to make a large MANET appear *smaller* and *less dynamic*, we propose a novel clustering scheme based on a number of properties of *diameter-2* graphs. Compared to virtual infrastructures with central nodes, our virtual infrastructure is more symmetric and stable, but still light-weight. In our clustering scheme, cluster initialization naturally blends into cluster maintenance, showing the unity between these two operations. Unlike the cluster maintenance algorithm in [47], our algorithm does not require maintaining complete cluster topology information at each node. We call our algorithm *tree-based* since cluster merge and split operations are performed based on a spanning tree maintained at some specific nodes. Extensive simulation results have shown the effectiveness of our clustering scheme when compared to other schemes proposed in the literature.

The remainder of this chapter is organized as follows: Following the motivation of our work described in Section III.1, Section III.2 presents technicalities that underlie the tree-based clustering scheme; Section III.3 provides the details of the tree-based clustering algorithms; Section III.4 presents our simulation results. Finally, Section III.7 offers concluding remarks and directions for further work.

## III.1 MOTIVATION

Essentially, a cluster is a subset of the nodes of the underlying network that satisfies a certain property $P$. At the network initialization stage, a *cluster initialization* algorithm is invoked and the network is partitioned into individual clusters each satisfying property $P$. Due to node mobility, new links may form and old ones may break, leading to changes in the network topology and, thus, to possible violations of property $P$. When property $P$ is violated, a *cluster maintenance* algorithm must be invoked. It is intuitively clear that the less stringent property $P$, the less frequently is cluster maintenance necessary.

As discussed in Section II.1, the precise definition of the desirable property $P$ of a cluster varies in different contexts. However, there are some general guidelines suggesting instances of $P$ that are desirable in all contexts. One of them is that a *consensus* must be reached quickly in a cluster in order for a cluster to work efficiently. Since the time complexity of the task of reaching a distributed consensus increases with the diameter of the underlying graph [48], *small-diameter* clusters are generally preferred in MANET [8]. As an illustration, some authors define property $P$ such that every node in the cluster is *1-hop* away from every other node, that is, each cluster is a diameter-1 graph [42]. A less restrictive widely-adopted definition of $P$ is the *dominance property* [7, 11, 29] which insists on the existence of a central cluster-head adjacent to all the remaining nodes in the cluster. In the presence of a central node, consensus is reached trivially: indeed, the cluster-head dictates the consensus.

Motivated by the fact that a cluster-head may easily become a traffic bottleneck and a single point of failure in the cluster, and inspired by the instability of the virtual infrastructures maintained by the node-centric clustering schemes, in the clustering scheme proposed by Lin and Gerla in [47], although the cluster initialization algorithm used is node-centric with the clusters featuring a central cluster-head, once clusters are constructed, [47] eliminates the requirement for a central node, defining the cluster simply as a diameter-2 graph. Only when the cluster is no longer a diameter-2 graph will a cluster change occur. This definition imposes fewer constraints on a cluster and hence may result in significant improvement on the stability of the resulting virtual infrastructure. In addition, Nakano and Olariu [53] have shown that a distributed consensus can be reached fast in a diameter-2 cluster. In the light of these observations, in this work we adopt the *diameter-2 property* as the *defining property of a cluster.*

The basic idea of the *degree*-based *cluster maintenance algorithm* of [47] is the following: when a violation of the diameter-2 property is detected, the highest degree node and its 1-hop neighbors remain in the original cluster and all the other nodes leave the cluster. It is expected that a leaving node will join another cluster or form a new cluster by itself. Unfortunately, the description of the algorithm in [47] is very succinct and many important details are glossed over.

In fact, there are several problems with the above degree-based cluster maintenance algorithm as discussed in [47]. To illustrate consider the cluster topology in Figure 2(a). When the link (3,4) is broken due to mobility, the diameter-2 property is violated. One problem is that various nodes have a different local view, precluding them from reaching a global consensus as to which node(s) should leave the cluster.

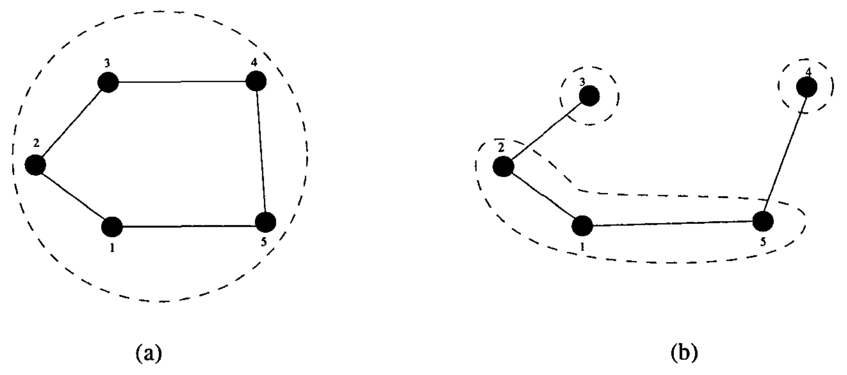


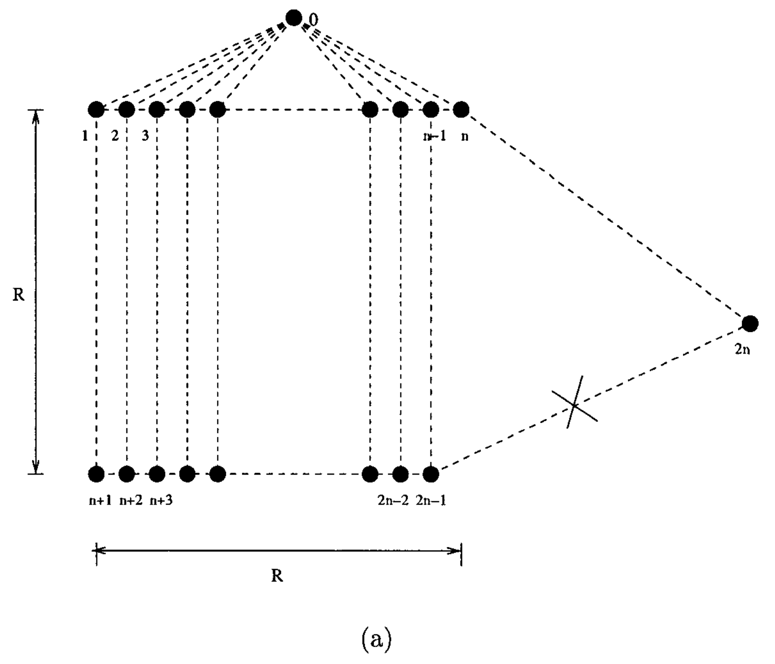
Fig. 2. An example of the degree-based cluster maintenance algorithm.

To wit, even if the highest degree of nodes in Figure 2(b) is propagated throughout the entire topology, the nodes still do not have sufficient information to decide whether or not they should leave the cluster. For example, node 3 is adjacent to node 2 which has degree two, thus being a highest-degree node. Consequently, node 3 decides that it should not leave the cluster. Likewise, node 5 is adjacent to node 4 which also has the highest degree and decides that it should not leave the cluster. The net effect is that no node will leave, invalidating the correctness of the cluster maintenance algorithm.

Notice that the insecurity we just outlined stems, in part, from the fact that in Figure 2(b) there are three highest-degree nodes: nodes 1, 2, and 5. The above problem can be helped somewhat by using the lowest nodeID criterion to break ties. Under this criterion, node 1 and its 1-hop neighbors, nodes 2 and 5, stay in the original cluster, and nodes 3 and 4 leave. Thus, in this case, the original cluster is partitioned into three clusters: {1,2,5}, {3}, and {4}.

Furthermore, if the cluster maintenance algorithm of [47] is to be fully distributed, each node must maintain the whole topology of the cluster; otherwise, the nodes cannot reach a consensus as to which is the *unique* node with the highest degree. Note that maintaining the complete topology of the cluster at each member node requires flooding the formation and breakage of *every* link to *all* the other nodes in the cluster, involving a large overhead.

The cluster maintenance algorithm of [47] tries to minimize the *number of node*



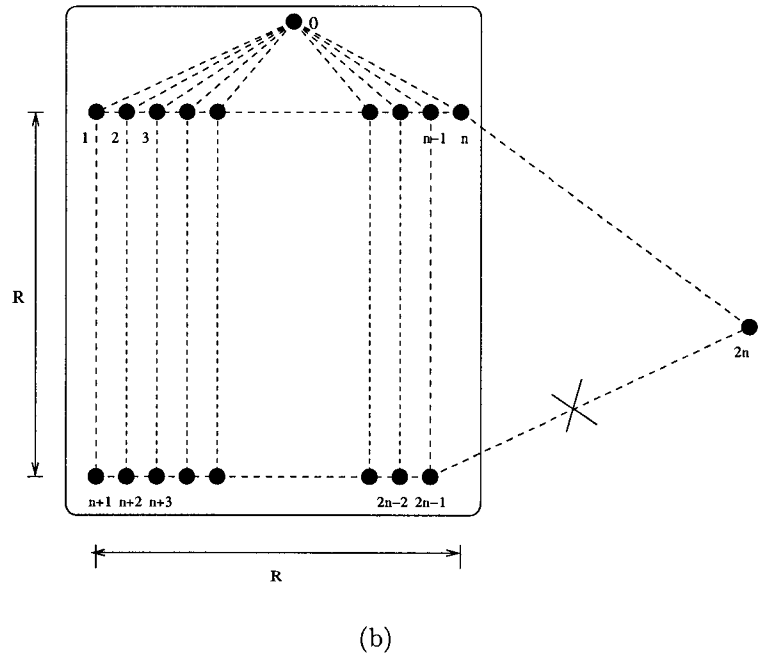


Fig. 3. An example in which the degree-based algorithm generates many leaving nodes.

*transitions* between clusters and this number is used to evaluate the *stability* of the cluster infrastructure. However, there is no guarantee that this algorithm will minimize node transitions. In the example shown in Figure 3(a), there are $2n+1$ nodes in the cluster, numbered from 1 to $2n+1$. Nodes $1, 2, \ldots, n$ are within transmission range ($R$) from each other; similarly, the nodes $n+1, n+2, \ldots, 2n-1$ are within transmission range from each other. With the breakage of link between nodes $2n-1$ and $2n$, the cluster is no longer diameter-2. Nodes $1, 2, \ldots, n$ have degree $n+2$ and are the highest-degree nodes. Assume that node 1 is chosen as the maintenance leader. In this case, according to the degree-based algorithm, $n-1$ nodes (namely, nodes $n+2, n+3, \ldots, 2n$) leave the cluster while, in fact, the minimum number of nodes that have to leave the cluster is just one as shown in Figure 3(b).

Moreover, using the number of node transitions as the *sole* criterion to assess the goodness of a cluster maintenance algorithm is misleading since: (a) It implicitly assumes that the highest-degree node is the same as the logical cluster representative. This assumption is not attractive since during normal operation of a cluster, the highest-degree node may change frequently due to link changes. If every highest-degree node change results in a migration of the logical cluster representative, a significant amount of overhead will be involved. (b) It assumes that only *leaving* nodes are responsible for the overhead in the cluster maintenance procedure. In reality, during the maintenance procedure, all nodes in the involved clusters participate in computation and message passing for determining the new cluster membership. Consider an example simulation for two clustering schemes 1 and 2. During the simulation, in Scheme 1, a cluster with 100 nodes are split once into two clusters, each with 50 nodes; in Scheme 2, a cluster with 100 nodes decreases its size by one node for 30 times. It is not clear that Scheme 2 is definitely more stable than Scheme 1; (c) In many cases, the degree-based algorithm generates *single-node* clusters. Such a cluster is of little use and must merge with some other existing cluster. This operation should be considered part of the overhead introduced by the cluster maintenance algorithm. Consider the following cluster infrastructure: each node is a single-node cluster and cluster merge never occurs. In such an infrastructure, the number of node transitions is 0. However, this is a very poor cluster infrastructure and the benefits of clustering are lost. This example clearly illustrates the tradeoff between cluster *stability* and *quality*. We must consider both metrics when evaluating the performance of a cluster maintenance algorithm.

## III.2 TECHNICALITIES

The main goal of this subsection is to develop the graph-theoretic machinery that will be used by our clustering algorithms. As customary, we model a multi-hop ad hoc network by an undirected graph $G = (V, E)$ in which $V$ is the set of nodes and $E$ is the set of links between nodes. The edge $(u, v) \in E$ exists whenever nodes $u$ and $v$ are 1-hop neighbors. Each node in the network is assigned a unique identifier (nodeID). The distance between two nodes is the length of the *shortest* path between them. The diameter of a graph is the largest distance between any pair of nodes. Our cluster maintenance algorithm relies on the following theorems of diameter-$d$ graphs.

**Theorem III.2.1** *Consider a diameter-d graph $G$ and an arbitrary edge $e$ of $G$. Let $G' = G - e$ be the graph obtained from $G$ by removing edge $e$. If $G'$ is connected, then there must exist a node in $G'$ whose distance to every other node is at most $d$. Moreover, the diameter of $G'$ is at most $2d$.*

Proof. Assume that the edge $e = (u, v)$ is removed. Since $G'$ is connected, there must exist a shortest path $P'(u, v) : u = x_1, x_2, \ldots, x_k = v$ joining $u$ and $v$ in $G'$. Consider node $x_{\lceil \frac{k}{2} \rceil}$. Clearly, the distance from $x_{\lceil \frac{k}{2} \rceil}$ to both $u$ and $v$ is unaffected by the removal of the edge $e = (u, v)$. We claim that the distance in $G'$ from $x_{\lceil \frac{k}{2} \rceil}$ to all the remaining nodes is bounded by $d$. To see this, consider an arbitrary node $y$ in $G$ and let $\Pi$ be the shortest path in $G$ joining $x_{\lceil \frac{k}{2} \rceil}$ to $y$. If $\Pi$ does not use the edge $e$, then the removal of $e$ does not affect $\Pi$. Assume, therefore, that $\Pi$ involves the edge $e$. Assume, without loss of generality, that in $\Pi$ node $v$ is closer to $y$ than $u$. However, our choice of $x_{\lceil \frac{k}{2} \rceil}$ guarantees that the path consisting of the nodes $x_{\lceil \frac{k}{2} \rceil}, x_{\lceil \frac{k}{2} \rceil + 1}, \ldots, x_{k-1}, v, \ldots y$ cannot be longer than $\Pi$, completing the first part of the claim.

Consider a BFS tree of $G'$ rooted at $x_{\lceil \frac{k}{2} \rceil}$. We just proved that the depth of this tree is bounded by $d$, confirming that the diameter of $G'$ is, indeed, bounded by $2d$.
□

Theorem III.2.1 has the following important consequence that lies at the heart of our cluster maintenance algorithm.

**Corollary III.2.2** *Consider a diameter-2 graph $G$ and an edge $e$ of $G$. Let $G' = G - e$ be the graph obtained from $G$ by removing edge $e$. If $G'$ is connected, then there must exist at least one node in $G'$ whose distance to every other node is at most two. Furthermore, the diameter of $G'$ is at most four.*

**Theorem III.2.3** *Let $G$ be a diameter-$d$ graph, and let $x$ and $y$ be a pair of nodes that achieve the diameter of $G$. Then the graph $G' = G - \{x\}$ is connected. Furthermore, in $G'$, any BFS tree rooted at $y$ has depth at most $d$.*

Proof. In $G$, $x$ is a level-$d$ (leaf) node of any BFS tree rooted at $y$. Hence removing node $x$ does not affect the distance from $y$ to any other node. Thus, $G'$ must be connected, and in $G'$, any BFS tree rooted at $y$ has depth at most $d$. □
Theorem III.2.3 has the following important consequence that will be used in our cluster maintenance algorithm.

**Corollary III.2.4** *Let $G$ be a diameter-2 graph, and let $x$ be a node in $G$ such that there exists at least one node $y$ in $G$ that is not adjacent to $x$. In the graph $G' = G - \{x\}$, any BFS tree rooted at $y$ has depth at most two.*

**Theorem III.2.5** *Consider a graph $G = (V, E)$, disjoint subsets $V_1, V_2$ of $V$, and let $G'$ be the subgraph of $G$ induced by $V_1 \cup V_2$.*

*(1) If the subgraphs of $G$ induced by $V_1$ and $V_2$ are diameter-$d$ graphs, and*

*(2) if for every node $x$ of $V_1$, the BFS tree of $G'$ rooted at $x$ has depth at most $d$*

*then $G'$ is a diameter-$d$ graph.*

Proof. Consider an arbitrary pair of nodes $u, v$ in $G$. We need to show that $u$ and $v$ are at distance at most $d$ in $G'$. Indeed, if $u, v \in V_1$ (resp. $V_2$), the conclusion is implied by assumption (1). Consequently, we may assume, without loss of generality, that $u \in V_1$ and that $v \in V_2$. By assumption (2), the BFS tree of $G'$ rooted at $u$ has depth at most $d$, implying that the distance between $u$ and $v$ is bounded by $d$. This completes the proof of Theorem III.2.5. □

## III.3 OUR TREE-BASED CLUSTERING ALGORITHM

In MANET link failures caused by node mobility can be predicted by the gradual weakening of the radio signal strength. In addition, since mechanical mobility and radio transmission occur at vastly different time scales, multiple link failures can be treated as a series of single-link failures. With this in mind, in this work we adopt the *single-link failure* and *single-node failure* models where either one *link* or one *node* fails at any one time. We also note that the single-node failure model can be used to account for the scenarios where link breakages occur unpredictably.

We make the following two assumptions: (1) a message sent by a node is received correctly by all its neighbors within a finite time, called a *message round*; (2) the cluster split algorithm is atomic in the sense that no new link/node failure occurs during its execution.

### III.3.1 The tree-based cluster split algorithm: single-link failure

In this subsection, we discuss the details of our cluster split algorithm in the case where a single-link failure occurs.

When a node detects the formation/breakage of one of its immediate links, it broadcasts a HELLO beaconing message containing its nodeID, clusterID, cluster size, the nodeIDs and clusterIDs of its 1-hop neighbors, as well as the signal strength of each link to its 1-hop neighbors. By receiving such beaconing messages, each node $u$ maintains a depth-2 BFS tree $T(u)$ rooted at $u$ itself and containing only the nodes belonging to the same cluster as $u$. Clearly, as long as the diameter-2 property holds, the distance between each pair of nodes is at most two, and the tree $T(u)$ contains *all* the nodes in the cluster. Thus, each node knows the number $n$ of nodes in its own cluster.

In our model, each node monitors the signal strength of the links joining it with its 1-hop neighbors. When a generic node $u$ detects that the signal strength of one of its links weakens below a threshold value, it reconstructs $T(u)$. By comparing the size $|T(u)|$ of $T(u)$ with $n$, node $u$ determines whether all the cluster members are still at most two hops away. If it finds that some member cannot be reached in two hops, it broadcasts a VIOLATION message to all of its 1-hop neighbors, identifying the single-link failure causing the violation of diameter-2 property. Each node $v$ receiving a VIOLATION message reconstructs its own tree $T(v)$ and checks whether $|T(v)|$

matches $n$. If there is a mismatch, the node forwards the VIOLATION message to all its neighbors; otherwise, it declares itself a maintenance leader. In other words, a maintenance leader is a node which can reach every other node in at most two hops. By Corollary III.2.2, after being forwarded at most once, the VIOLATION message will reach a maintenance leader. Note that there might be multiple maintenance leaders: each of them runs an instance of the cluster split algorithm independently. Finally, *the instance which yields the best quality new clusters is adopted.*

For a generic maintenance leader $x$, the tree $T(x)$ is composed of: (1) node $x$ itself – the root of the tree; (2) level-1 nodes, that is, $x$'s 1-hop neighbors in the original cluster; (3) level-2 nodes, all the remaining nodes in the original cluster.

During the split procedure, there can be several different considerations as to how to split the original cluster. Our motivation is to minimize the number of newly generated clusters when splitting. In addition, by considering link stability during a split, the newly generated clusters tend to be more stable.

Specifically, a generic maintenance leader $x$ performs the following steps:

Step 1. Node $x$ tries to find the minimum number of level-1 nodes to cover all the level-2 nodes. A level-1 node $y$ can cover a level-2 node $z$ if and only if $x$ can reach $z$ through $y$. This is an instance of the well-known minimum set covering problem and can be solved using the following greedy heuristic [23]:

Initially, all level-2 nodes are marked *uncovered*, and all the level-1 nodes constitute the *total level-1 set*. For each node $y$ in the total level-1 set, $x$ calculates the number $N_y$ of uncovered level-2 nodes that can be covered by $y$. The node $y$ with the largest $N_y$ is deleted from the total level-1 set, added to the *critical level-1 set* and marked as *new leader*. All the $N_y$ level-2 nodes covered by $y$ are marked *covered.* Node $x$ continues the above process until all the level-2 nodes are marked covered. We call the current total level-1 set as *redundant level-1 set.* For each level-2 node $z$ marked covered, $x$ calculates the stability (i.e. signal strength) of the link $STA_{zw}$ between $z$ and every critical level-1 node $w$. Denote the node $w$ that has the largest $STA_{zw}$ as $p$. Node $x$ marks $w$ as new member of $p$.

Step 2. Next, $x$ tries to use the nodes in the critical level-1 set to cover the nodes which are left in the redundant level-1 set. For each node $r$ in the redundant level-1 set, $x$ determines the stability of the link between $r$ and each of the critical level-1 nodes adjacent to $r$. Denote the one that has the most stable link as $w$; $x$ marks $r$

as a new member of $w$.

Step 3. $x$ checks whether there exist nodes in the redundant level-1 set. If so, $x$ marks itself new leader and all the uncovered nodes in the redundant level-1 set as new members of $x$. Otherwise, $x$ finds a new leader $q$ which has the largest link stability value in the critical level-1 set and marks itself as new member of $q$.

At this point, $x$ has reached its cluster split decision. It broadcasts the result through a MAINTENANCE-RESULT message to all its 1-hop neighbors. A node finding itself chosen as a new leader further broadcasts a MEMBER-ENLIST message containing its new cluster membership list. Upon receiving such a message, each node in the original cluster knows its new membership. This completes the split procedure in the case of a single-link failure.

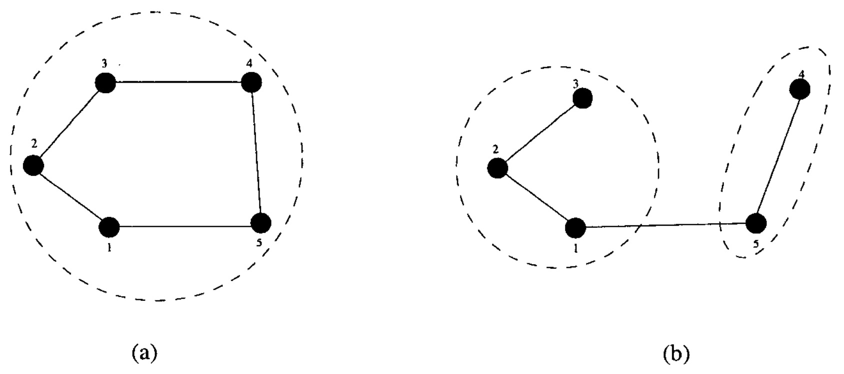


Fig. 4. An example of the tree-based cluster split algorithm.

We now illustrate the tree-based split algorithm using an example. There are five nodes in the cluster shown in Figure 4(a). When the link (3,4) is broken, nodes 3 and 4 detect that the diameter-2 property is violated. Each of them broadcasts a VIOLATION message. Upon receiving the VIOLATION message, nodes 2 and 5 reconstruct their respective BFS trees. Since neither of them can work as maintenance leader, they forward the VIOLATION message. When node 1 receives the VIOLATION message, it reconstructs $T(1)$ and finds that $|T(1)| = 5$. At this point, node 1 knows that it is a maintenance leader. In $T(1)$, node 2 covers node 3, and node 5 covers node 4. Hence nodes 2 and 5 are chosen as critical level-1 nodes. Assuming that link (1,2) is more stable than link (1,5), node 1 chooses to be covered by node 2. The result of this split procedure is two new clusters: {1,2,3} and {4,5}, as shown in Figure 4(b).

### III.3.2 The tree-based cluster split algorithm: single-node failure

Our cluster split algorithm for the case when a single-node failure occurs relies, in part, on Corollary III.2.4. Indeed, by Corollary III.2.4, when a single-node failure occurs in a cluster and the tree maintained by the failed node (just before its failure) has depth two, then the resulting graph is still connected (although it need not be diameter-2) and there must be some node that still maintains a BFS tree with depth at most two. This means that a maintenance leader still exists, and that we can still use our tree-based cluster split algorithm. Specifically, when a node detects the *sudden* breakage of a link to/from a 1-hop neighbor, it assumes a node failure, deletes the failed node from its cluster membership list, and reconstructs the BFS tree. A VIOLATION message is sent out when necessary, identifying the single-node failure causing the violation of diameter-2 property. The remaining steps are the same as those described in Subsection III.3.1.

However, if the failed node maintains a depth-1 (as opposed to depth-2) tree before its failure, it is possible that none of the remaining nodes can play the role of maintenance leader. To solve this problem, during the cluster's normal operation phase, when a node finds that it is the only node maintaining a depth-1 tree in the cluster, it periodically runs a minimum dominating set (MDS) algorithm (using a greedy algorithm similar to that described in Subsection III.3.1) on its 1-hop neighbors, and notifies the nodes in the MDS to become candidate maintenance leaders. When the node fails, each candidate maintenance leader detects this failure and immediately broadcasts a MEMBER-ENLIST message containing its new cluster membership list. Upon receiving such a message, each node in the original cluster knows its new membership. This completes the split procedure in the case of a single-node failure.

### III.3.3 Merging clusters

The previous discussion focused on one aspect of cluster maintenance: the cluster split procedure. Clearly, cluster maintenance cannot rely on cluster splitting only, for otherwise the size of the clusters will continually decrease, and we would end up with many one-node clusters, defeating the purpose of clustering. To prevent this phenomenon from occurring, the other necessary component is a mechanism for merging two clusters. The main goal of this subsection is to discuss a simple tree-based cluster merge procedure.

When the members of two clusters move close so that they can reach each other in two hops, the two clusters may be merged. To better control the cluster merge procedure and to prevent it from being invoked too frequently, we introduce the concept of *desirable size of a cluster*. Specifically, given system parameters – desirable cluster size $k$ and tolerances $\alpha$, $\beta$ , we insist that clusters should have size in the range $[k-\alpha, k+\beta]$. Clusters of size less than $k-\alpha$ are said to be *deficient*. Only deficient clusters are seeking neighboring clusters with which to merge.

For definiteness, consider a deficient cluster $A$ of size $|A| < k - \alpha$. By receiving HELLO beaconing messages described in Subsection III.3.1, the nodes in $A$ maintain a list of feasible clusters for merging. Among these, the one, say, $B$ such that $|A| \leq |B|$ and $|A| + |B|$ is as close as possible to $k$ but not exceeding $k + \beta$ is selected. *ClusterID* is used to break ties. Upon selecting $B$ as a candidate, the nodes of $A$ that have a 1-hop neighbor in $B$ broadcast a MERGE-REQUEST message. If $B$ is not involved in a merge operation, the nodes of $B$ that have received the MERGE-REQUEST message send back a MERGE-ACK message. At this point, every node in cluster $A$ computes its BFS tree involving nodes in $A \cup B$. A node in $A$ for which the size of the corresponding tree differs from $|A| + |B|$ sends a VIOLATION message to the other nodes in $A$. By virtue of Property III.2.5, if no VIOLATION message is received, $A \cup B$ is a diameter-2 graph. In this case, the nodes in cluster $A$ broadcast a MERGE-CONFIRMATION message to cluster $B$ indicating the new cluster membership and the merge procedure terminates. If, however, a VIOLATION message was received, the merge operation is aborted, a MERGE-ABORT message is sent to the nodes of cluster $B$, and a new candidate for merging is examined.

We note that the merge operation takes precedence over split. To explain the intuition behind this design decision refer to Figure 5. Here cluster $X$ consisting of nodes $e$ and $f$ attempts to merge with cluster $Y$ consisting of nodes $a$, $b$, $c$, and $d$. Assume that either while the request to merge is issued or just prior to that the edge $(a, d)$ broke, invalidating $Y$ as a cluster. Rather than proceeding with the split operation, as would normally be the case, the merge operation is given priority. As illustrated in the figure, all nodes in $X$ and $Y$ detect that $X \cup Y$ has diameter two and is, therefore, a valid cluster. We note, however, that had $X \cup Y$ had diameter larger than two, the merge operation would have failed and the nodes in $Y$ would have proceeded with the split operation.

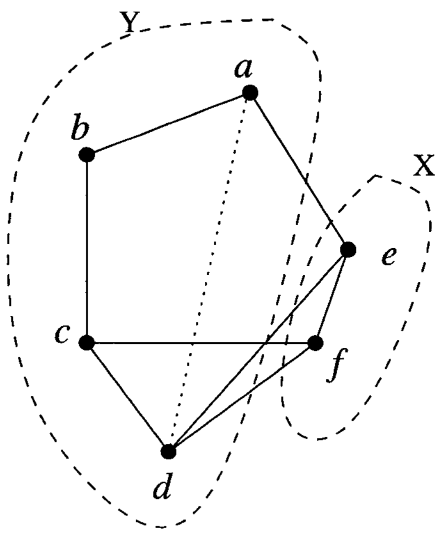


Fig. 5. An example illustrating the priority of merge over split.

## III.3.4 Cluster initialization

The cluster merge algorithm described in Subsection III.3.3 is perfectly general and can, in fact, be used for the purpose of cluster initialization. Initially, each *node* is in a *cluster* by itself. The cluster merge algorithm is started as described above. The initialization algorithm naturally blends into cluster maintenance as more and more clusters reach desirable size.

It is worth noting that our cluster initialization algorithm has a number of advantages over the nodeID-based algorithms. First, our algorithm is cluster-centric, as opposed to node-centric. Second, the natural blend of cluster initialization and cluster maintenance shows the unity between these two operations. This is certainly not the case in the vast majority of clustering papers in the literature. Third, our cluster initialization algorithm (just as the cluster merge) can be performed in the presence of node mobility.

Last, our initialization algorithm results in better quality clusters than the

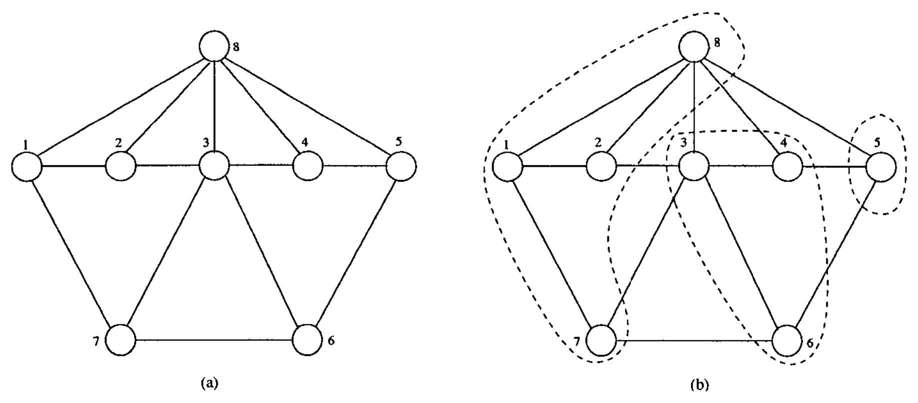


Fig. 6. An example of the cluster initialization algorithm.

nodeID-based algorithms. To see this, consider the sub-network in Figure 6(a) and assume that the desirable cluster size ($k$) is seven with tolerances $\alpha = \beta = 2$. It is not hard to see that our initialization algorithm actually returns the entire sub-network as a single cluster – for this graph is diameter-2. On the other hand, the nodeID-based algorithm results in many deficient clusters, as illustrated in Figure 6(b).

## III.4 PERFORMANCE ANALYSIS AND SIMULATION RESULTS

In this subsection, we use simulation to demonstrate the effectiveness of our tree-based clustering scheme compared to other clustering schemes in the literature. We choose LCC [20] as a representative of the node-centric clustering schemes since it avoids the global rippling effect and greatly reduces cluster changes compared to the other nodeID-based algorithms. In addition, it is shown in [36] that in the *unit-disk graph* model, LCC is asymptotically optimal with respect to the number of clusters maintained in the system.

### III.4.1 Performance metrics

As discussed in Subsection III.1, we need to consider both cluster *quality* and cluster *stability* in our comparison. The number of clusters in the system is generally considered as a good indication of the quality of a cluster infrastructure [4, 36]. A clustering scheme that generates and maintains fewer clusters is potentially able to accommodate more nodes in a cluster, hence providing better load balancing among clusters. In our simulation, we count the number of clusters in the system once every second of simulation time. We calculate the sum of these numbers divided by the total simulation time, and we use this *average number of clusters* maintained in the system to characterize cluster *quality*. We note, however, that the number of clusters maintained does not tell the whole story. Given two clustering algorithms that maintain, essentially, the same number of clusters, we prefer the one that generates clusters of roughly equal size to the one that generates a mix of very large and very small clusters. Indeed, clustering schemes that generate very small clusters have to rely on frequent cluster merges to keep cluster quality, clearly an undesirable situation.

To evaluate cluster *stability*, we assume that each cluster chooses one of its member as cluster leader and takes its nodeID as the clusterID. When a node is no longer in the same cluster as its latest cluster leader, this node is considered as a node *changing cluster*. Note that the cluster leader defined here serves only as a reference point that allows us to count and compare the number of *node transitions* in different clustering schemes. In LCC, the central node of a cluster is always the cluster leader. In the diameter-2 schemes, each node initially chooses its nodeID as the clusterID of the single-node cluster. When two clusters merge, the clusterID of the cluster with larger size is used as the new clusterID. When a cluster split happens, among the new clusters, the one which contains the original cluster leader still keeps the original clusterID, and all the other clusters choose the minimum nodeID of its members as the new clusterID. Further, we need to clearly identify the events that can cause cluster changes. In *LCC*, there are two types of events that can cause nodes to change clusters:

- a non-leader node is no longer adjacent to its leader; in this case, the node joins another leader, or becomes itself a new leader;
- when two cluster leaders become neighbors, the one with larger nodeID gives up its role, and all the nodes in its cluster either join a new cluster, or become new leaders by themselves.

In the *diameter-2* schemes, the two types of events that can cause nodes to change clusters are:

- a cluster is no longer diameter-2, and is split to several sub-clusters;

- a cluster merges with another cluster.

With the above assumptions in mind, we define two measurements to evaluate cluster *stability*: (1) *total number of nodes changing clusters*; (2) *average cluster lifetime*. Specifically, we compare the snapshots of the system taken exactly before and after the execution of the maintenance algorithm triggered by either of the above events. If node $x$'s clusterID after the event is different from its clusterID before the event, then it is counted as a *node changing its cluster*. If a node $x$ is a cluster leader before the event, but no longer a leader after the event, then the cluster is considered as disappearing and we stop increasing its lifetime. If a node $x$ is not a cluster leader before the event, but becomes one after the event, then a new cluster is considered generated, and we start increasing its lifetime. The *average cluster lifetime* is calculated as the sum of all the cluster lifetimes divided by the number of clusters generated in the simulation.

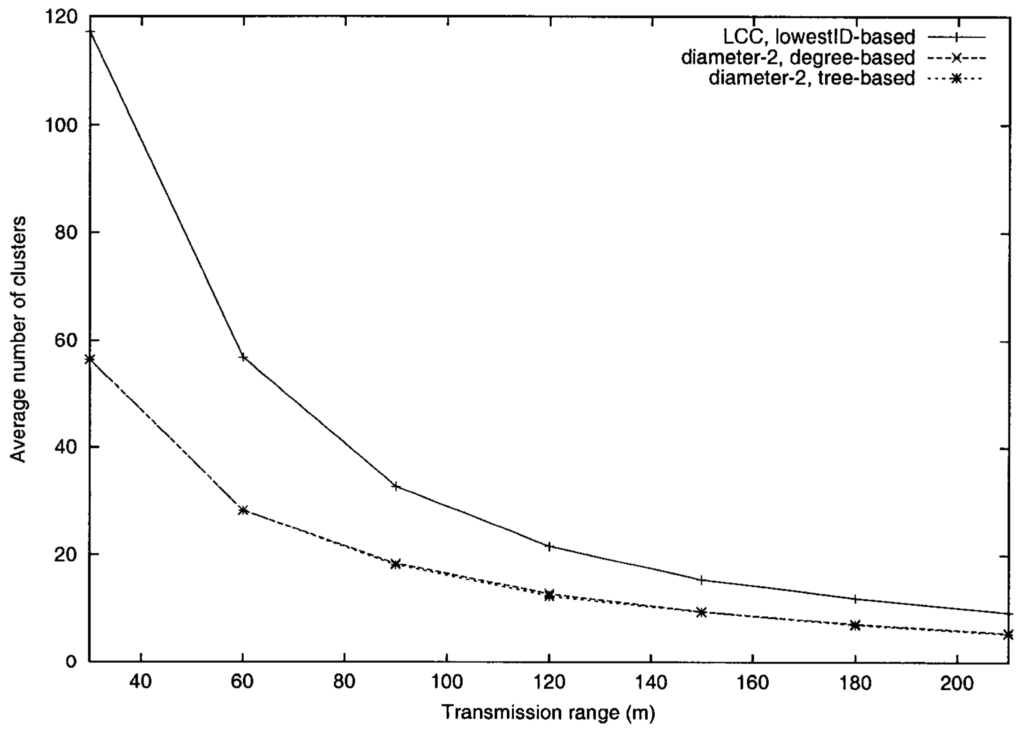


Fig. 7. Comparing the performance of different clustering schemes: average number of clusters.

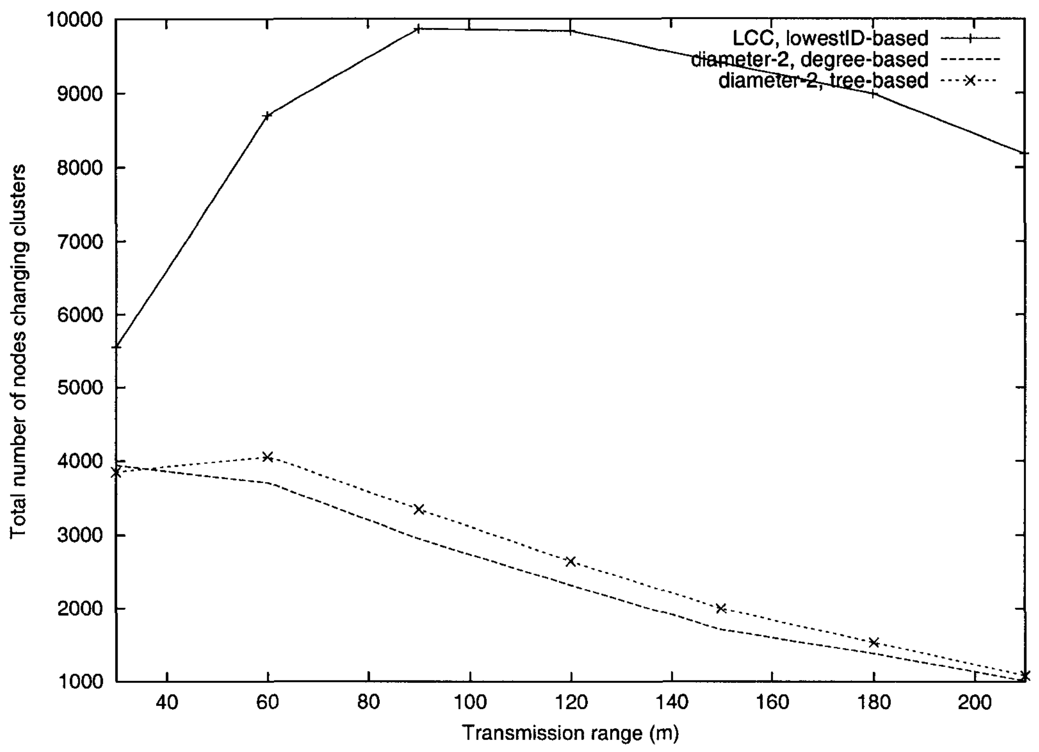


Fig. 8. Comparing the performance of different clustering schemes: total number of nodes changing clusters.

## III.4.2 Simulation results

We simulate a MANET by placing $N$ nodes within a bounded region of area $A$. The nodes move according to the random way-point model [18] with zero pause time and constant node speed $V$. All the nodes have uniform transmission range, which varies from 30m to 210m in different simulations. For each simulation, we examine the first 300 seconds of simulation time. All the simulation results presented here are an average of 10 different simulation runs. We also plot 95% confidence intervals for the means. The small confidence intervals show that our simulation results precisely represent the unknown means.

A set of representative simulation results ($N$=100, $A$=500m × 500m, $V$=5m/s) are shown in Figures 7 – 10. For the tree-based algorithm, we implement a *baseline* version which does not consider link stability during cluster split. Also, since the tree-based algorithm allows for controlling cluster merging frequency and LCC and

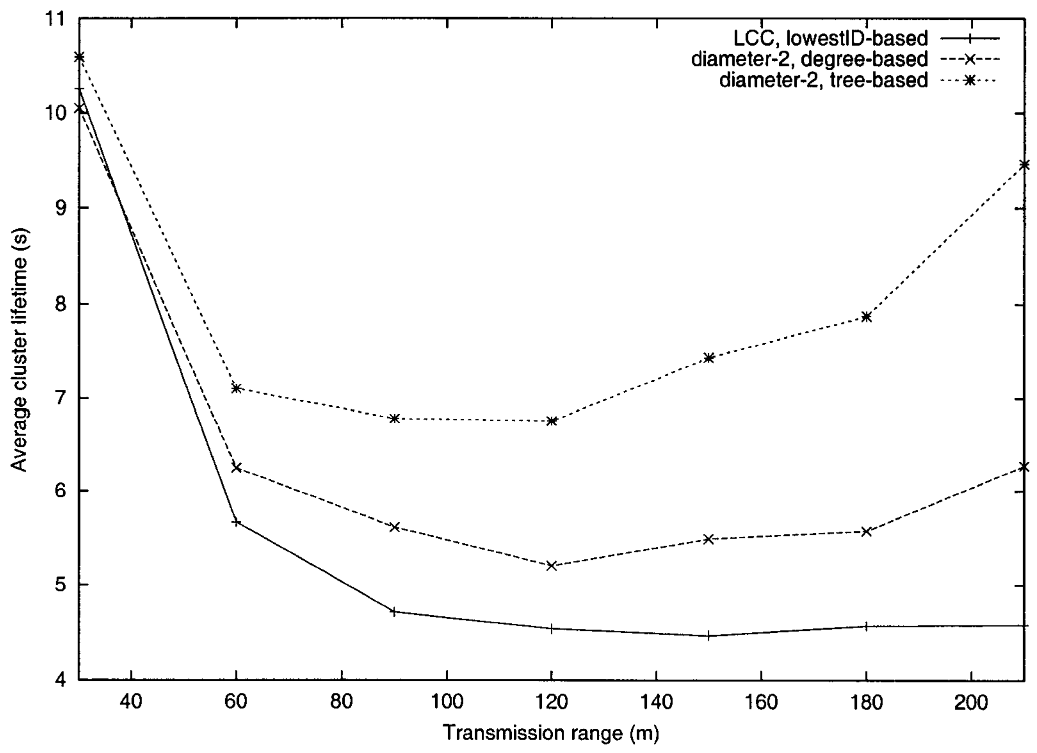


Fig. 9. Comparing the performance of different clustering schemes: average cluster lifetime.

the degree-based algorithm do not, we have set the *desirable size of a cluster* to $\infty$.

*(A) Comparing the node-centric LCC and the cluster-centric diameter-2 schemes* Figure 7 indicates that the average number of clusters in the system maintained by the diameter-2 clustering schemes is about *half* of that maintained by LCC. Figure 8 shows that the number of nodes changing clusters in LCC is significantly larger than in either of the diameter-2 schemes. This is hardly a surprise since LCC is node-centric and it is obvious that clusters predicated on the existence of a central node (the cluster-head) are more brittle than regular diameter-2 clusters. This is further confirmed by Figure 9 that illustrates that the average lifetime of clusters generated by LCC is shorter than the lifetime of clusters generated by either of the diameter-2 schemes. These results demonstrate that by removing the central-node constraint, the diameter-2 cluster is a much more *stable* structure and can provide *better quality* clusters, especially in MANET applications where central node is not necessary, such as [6, 47, 49, 62].

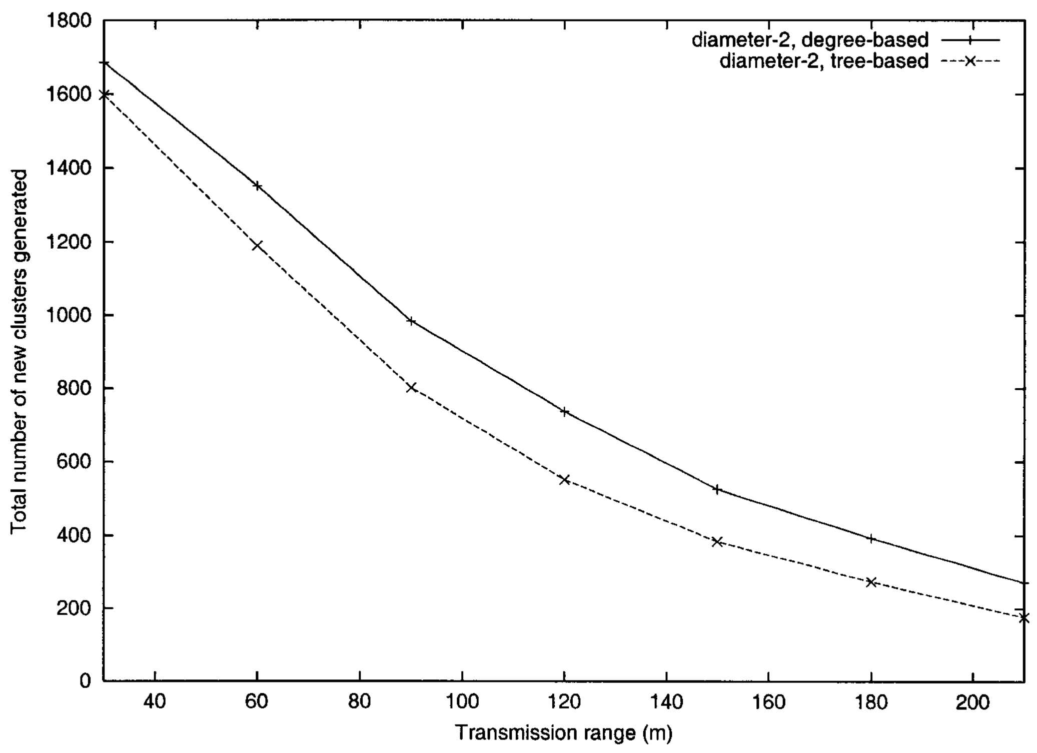


Fig. 10. Comparing the performance of different clustering schemes: total number of clusters generated during simulation.

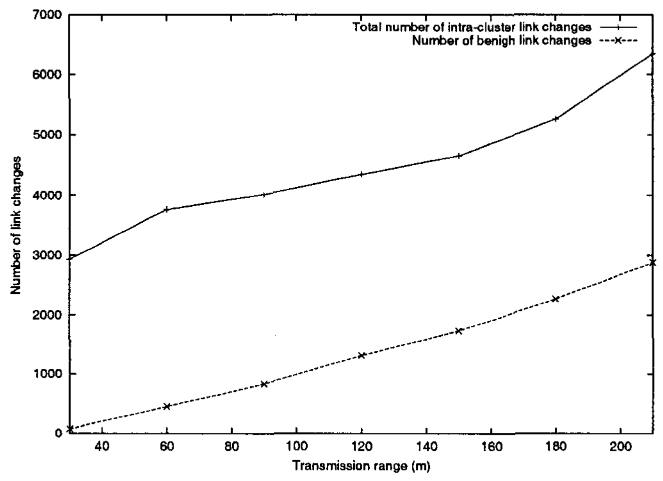
*(B) Comparing the tree-based algorithm and the degree-based algorithm*

In terms of the average number of clusters maintained in the system, the tree-based algorithm is slightly better than the degree-based algorithm as shown in Figure 7. Figure 9 shows that the average cluster lifetime in the tree-based algorithm is longer than in the degree-based algorithm. From Figure 10, we can see that the degree-based algorithm generates many more new clusters than the tree-based algorithm. On the other hand, Figure 8 shows that the total number of nodes changing clusters is significantly larger in the tree-based algorithm than in the degree-based algorithm. The explanation is simple: the degree-based algorithm tends to generate single-node clusters during cluster split, while the clusters generated by the tree-based algorithm are much more *balanced*. The net effect is that when a cluster split/merge happens, a larger number of nodes change clusters in the tree-based algorithm than in the degree-based algorithm. This result shows that the number of nodes changing clusters is not always indicative of the quality of the cluster maintenance algorithm. Note that the single-node clusters generated in the degree-based algorithm are *short-living* and will be merged with other clusters soon, hence they do not significantly influence the *average number of clusters* maintained in the system shown in Figure 7.

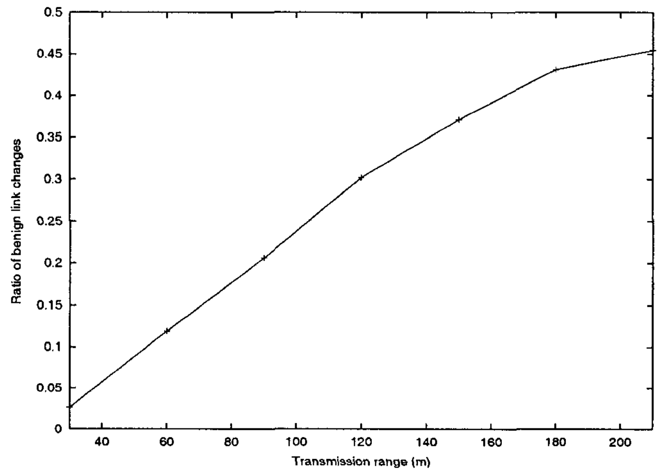
It is important to realize that what really distinguishes the tree-based algorithm and the degree-based algorithm is the *cluster maintenance overhead.* Since the degree of a node is a rather unstable parameter, in the degree-based algorithm, *every* link change (formation and breakage) has to be forwarded to *all* the cluster members. This is certainly not the case in the tree-based algorithm where, as long as the cluster is still diameter-2, link formation and link breakage are propagated in the HELLO beaconing message as described in Subsection III.3 and will not be forwarded by the other nodes.

To take this point one step further, we count the total number of intra-cluster link changes during the simulation. We call a link change between nodes $A$ and $B$ in the same cluster *benign* if after the change nodes $A$ and $B$ remain in the original diameter-2 cluster, and $A$ and $B$ have a common 2-hop neighbor. For example, in the cluster shown in Figure 6(a), the breakage of link (6,7) is benign since the resultant graph is still diameter-2, and nodes 6 and 7 have a common 2-hop neighbor (node 8). However, the breakage of link (3,8) is not benign since nodes 3 and 8 do not have a common 2-hop neighbor. We note that, trivially, the tree-based algorithm saves *at least* one message forwarding per benign link change over the degree-based algorithm. We count the number and ratio of benign link changes, and the corresponding simulation results are shown in Figure 11. As the simulation result shows, the ratio of benign link changes is quite significant, and as the node density becomes higher, the savings become more and more significant.

Our simulation results have revealed an interesting piece of evidence that speaks for the robustness of our tree-based algorithm: even when multiple link failures occur in a cluster, the probability of the existence of a maintenance leader is still very high. Theoretically, when multiple edges are removed from a diameter-2 graph, there may no longer exist a maintenance leader in the resulting graph. There are two approaches that can be employed by the tree-based algorithm to deal with this situation. The first approach is to predict link failure ahead of time whenever possible. Thus, when



(a) Number of benign intra-cluster link changes



(b) Ratio of benign link changes

Fig. 11. Comparing the maintenance overhead of tree-based and degree-based algorithms.

multiple link failures occur at the same time, all these links are actually still there, and the maintenance leader can arbitrarily choose one link as the only broken link. Essentially, this prevents real link failures from occurring in the first place. The second approach is to simply let multiple link failures occur. By Corollary III.2.2, if a maintenance leader exists, each node will know the maintenance result in at most four message rounds. A node sets a 4-message round long timer when violation is detected. Upon time-out, each node uses the cluster initialization algorithm described in Subsection III.3.4 as the last resort for cluster maintenance.

## III.5 APPLICATION

In this section, we discuss *topology control* [44] in mobile ad hoc networks as a sample application of the cluster-based general-purpose infrastructure we have proposed.

Cluster-based infrastructure provides a natural framework for designing topology control algorithms. In such a framework, no node maintains the global topology. Instead, the framework relies on clustering where nodes autonomously form groups. In each cluster, a centralized topology control algorithm is executed by a clusterhead, or a distributed topology control algorithm is executed by all the nodes, so that a some desirable topology properties are achieved in the cluster. The desired topology properties between clusters are achieved by exchanging information between adjacent clusters.

Motivated by the above idea, we propose a cluster-based algorithm to construct an approximate Minimum Spanning Tree (MST). The algorithm has three phases: (1) Phase 1: Cluster formation. A distributed clustering algorithm is used to generate and maintain clusters in the network. In this work, our focus is the diameter-2 clustering we have proposed; (2) Phase 2: Forming intra-cluster MST. In our infrastructure, since each cluster is diameter-2, a distributed MST algorithm exists that finishes very quickly [48]. Alternatively, a leader for topology control can be elected in each cluster, which is responsible for running a centralized MST algorithm (such as Kruskal's algorithm [23]). Note that this leader is a *logical* leader for the topology control application only; (3) Phase 3: Connecting clusters. In this phase, connectivity between adjacent clusters is considered. Each cluster runs the following algorithm: by exchanging information with neighboring clusters only, a cluster knows its *shortest* link to each of its adjacent clusters, as well as the shortest links between each pair of its adjacent clusters. Based on this information, a cluster constructs a Localized Minimum Spanning Tree (LMST) [43]. Note that each node in the LMST is a cluster, and each edge is the LMST is an actual link between two nodes. When running the LMST to establish connections between two adjacent clusters, the power assigned to the involved nodes is increased only. The collections of all edges in the LMSTs constructed by all nodes, as well as the links selected in Phase 2, form the resulting structure.

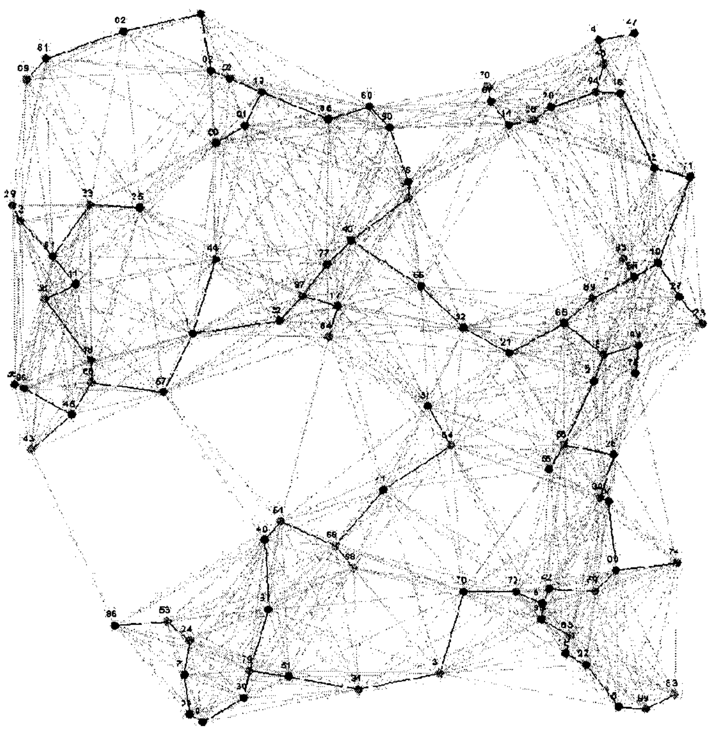


Fig. 12. Resulting structure formed by the centralized Kruskal's algorithm.

We have conducted a simulation study to determine the effectiveness of our cluster-based MST algorithm. In this study, 100–500 nodes were distributed uniformly at random in an area of $1000 * 1000m^2$. When operating at full transmission power, each node has a transmission range of 250m. In the simulation, for a specific number of nodes, we generate 50 different topologies. And the result is the average of these 50 simulation runs. Also, in this simulation, we consider static topology only.

We consider the following metrics in the simulation: (1) The two most important metrics, average link length and number of links in the resulting topology, consider only the bi-directional links in the resulting connected structures. For a connected network with $N$ nodes, its MST has $N-1$ links. The average link length is calculated as the sum of the length of each link divided by the number of links; (2) The degree of the node is counted in the following way: for a node $u$ with transmission power $P_u$, and a node $v$ with transmission power $P_v$, if the distance between $u$ and $v$ is not larger than $P_v$, then node $v$ is considered as a neighbor of $u$. Note that this relationship is not symmetric; (3) Average node power is calculated as the sum of the powers assigned to each node divided by the total number of nodes in the network; (4) Max

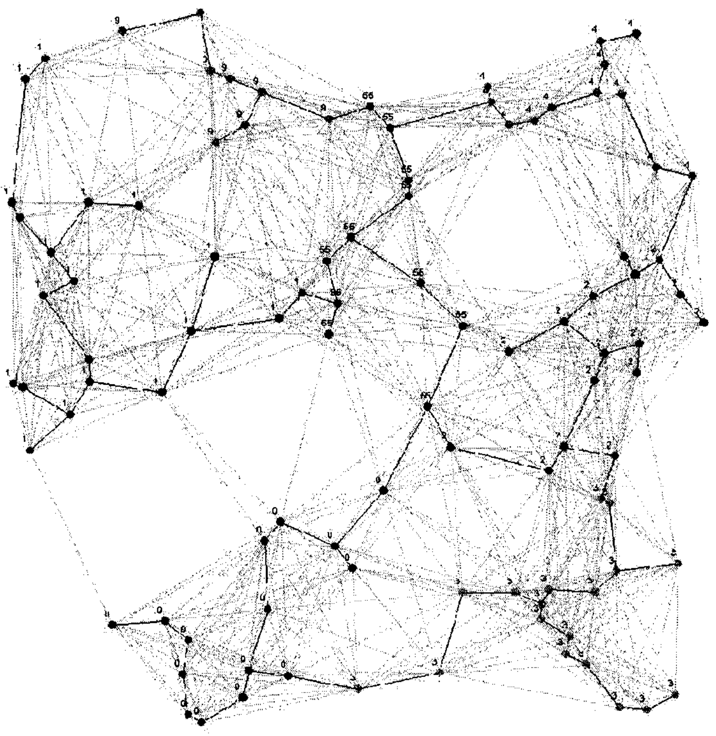


Fig. 13. Resulting structure formed by the cluster-based MST algorithm (diameter-2).

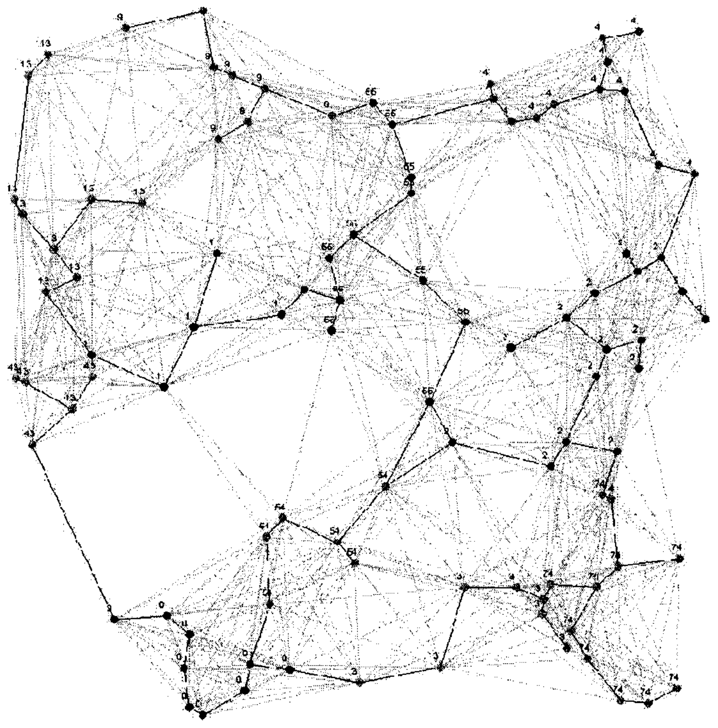


Fig. 14. Resulting structure formed by the cluster-based MST algorithm (with central node).

node power: it is the maximum value among the powers assigned to the nodes in the network.

A sample topology and the resulting structures generated by the three different topology control algorithms are shown in Figure 12, 13, and 14. From Figure 13 and 14, we can see that for this specific topology, seven clusters are generated by the diameter-2 clustering scheme (clusters $0, 1, 2, 3, 4, 9, 55$), while eleven clusters are generated by the lowestID clustering scheme (clusters $0, 1, 2, 3, 4, 9, 13, 43, 54, 55, 74$). More generally, for any given topology, the diameter-2 clustering scheme can potentially generate/maintain fewer (or equal) number of clusters than any central-node based clustering scheme, hence there are more topology information available for making *intra*-cluster decisions (since there are *more* nodes in a cluster) and for making *inter*-cluster decisions (since there are *fewer* clusters in the network), leading to a better-quality global structure.

More detail simulation results are shown in Table 2. In the table, *MST* is the result using a centralized Kruskal's algorithm, *Diameter-2* and *LowestID* are the results of diameter-2 clustering and the lowestID clustering, respectively.

From the simulation result, it is evident that resulting topology constructed by our cluster-based MST algorithm approximates the MST effectively in terms of all the four performance metrics used. Specifically, (1) The average link length of the resulting structure is very close to the optimal value (the approximation ratio is 1.06, 1.06, 1.05, 1.08, 1.05 as the number of nodes increases from 100 to 500); the number of links in the resulting structure is about three more than the optimal value, regardless of the number of nodes in the networks (the approximation ratio is 1.03, 1.02, 1.01, 1.01, 1.01 as the number of nodes increases from 100 to 500); (2) The average node degree keeps stable when the number of nodes increases; (the approximation ratio is 1.15, 1.16, 1.14, 1.12, 1.16 as the number of nodes increases from 100 to 500); (3) The average node power is very close to the optimal value (as the number of nodes increases from 100 to 500; the approximation ratio of the average node power is 1.09, 1.08, 1.07, 1.07, 1.07); (4) the approximation ratio of the max node power is a little high (1.16, 1.27, 1.25, 1.27, 1.27 as the number of nodes increases from 100 to 500). This is expected since max node power is determined by the critical part of a network. In fact, it is proved in [44] that it is impossible for any localized algorithm to construct a connected structure such that the max node power based

TABLE 2
Performance Comparison of the Three Topology Control Algorithms

| Number of nodes | Algorithm | MST | Diameter-2 | LowestID |
|---|---|---|---|---|
| 100 | Max node power | 164.44 | 190.32 | 192.23 |
| 100 | Average node power | 82.19 | 89.89 | 90.89 |
| 100 | Average node degree | 2.51 | 2.89 | 2.93 |
| 100 | Average link length | 68.06 | 72.14 | 72.77 |
| 100 | Number of links | 99 | 102.42 | 103.20 |
| 200 | Max node power | 116.74 | 147.80 | 149.71 |
| 200 | Average node power | 57.73 | 62.51 | 62.88 |
| 200 | Average node degree | 2.51 | 2.90 | 2.92 |
| 200 | Average link length | 47.42 | 50.32 | 50.50 |
| 200 | Number of links | 199 | 202.58 | 202.94 |
| 300 | Max node power | 99.44 | 124.19 | 125.79 |
| 300 | Average node power | 46.97 | 50.41 | 50.53 |
| 300 | Average node degree | 2.50 | 2.85 | 2.86 |
| 300 | Average link length | 38.66 | 40.70 | 40.79 |
| 300 | Number of links | 299 | 302.78 | 303.06 |
| 400 | Max node power | 86.70 | 110.51 | 113.82 |
| 400 | Average node power | 40.28 | 42.94 | 43.15 |
| 400 | Average node degree | 2.51 | 2.81 | 2.83 |
| 400 | Average link length | 33.19 | 35.74 | 34.85 |
| 400 | Number of links | 399 | 402.92 | 403.52 |
| 500 | Max node power | 78.44 | 99.75 | 100.42 |
| 500 | Average node power | 36.00 | 38.36 | 38.48 |
| 500 | Average node degree | 2.51 | 2.80 | 2.81 |
| 500 | Average link length | 29.67 | 31.09 | 31.15 |
| 500 | Number of links | 499 | 503.58 | 504.04 |

on this structure is within a constant factor of that based on MST.

In the simulation result, the diameter-2 clustering consistently generates better-quality structures than the lowestID clustering in terms of all the performance metrics used; however the difference between the two is small. The reason is that the simulation is conducted on static topologies only, and under static topologies, the difference between these two clustering schemes is not as dramatic as the difference in face of mobility (see Figure 7). The advantage of diameter-2 clustering scheme is expected to be more obvious in face of node mobility.

Finally, it is worth emphasizing that in this section we use MST construction as an illustration of the application of our proposed general-purpose infrastructure; but in fact, the cluster-based infrastructure provides a powerful general framework, and similar approaches can be used to establish many other global structures such as *strongly-connected* graphs [71].

## III.6 PROOF OF SOME PROPERTIES OF DIAMETER-2 GRAPHS

In this section, we prove some properties of diameter-2 graphs.

Let $T$ be a set of nodes (with transmission range $D$) on the plane with the following property *P1*:

**P1 (diameter-2 property)**: For every two nodes $p, q \in T$, there exists a node $r \in T$ such that $|pr| \leq D$ and $|qr| \leq D$. If $|pq| \leq D$, we can take $r$ to be either $p$ or $q$.

Let $T_d \subseteq T$ be a subset of $T$ with the following property *P2*.

**P2 (dominating property)**: For every point $x \in T$, there exists a point $y \in T_d$ such that $|xy| \leq D$.

**Lemma III.6.1** *Let $V$ be a circle. The chord $\overline{pq}$ divides $V$ into two parts $V = V_+ \cup V_-$. Let $U_r$ be the circle of centered at $r$. If $U_r$ covers both point $p$ and point $q$, then $U_r$ covers either $V_+$ or $V_-$.*

Proof. We prove by contradiction. In the following, we assume that chord $\overline{pq}$ divides $V$ into a *left* part and a *right* part.

Case 1: Assume that both $p$ and $q$ are on the boundary of the circle $U_r$. Since $U_r$ can cover neither $V_+$ nor $V_-$, then if there is a third intersecting point between circle $U_r$ and circle $V$, $U_r$ is same as $V$. So, $p$, $q$ must be the only two intersecting points between $U_r$ and $V$. Consider the arc of $U_r$ on the right side of $\overline{pq}$, the center of $U_r$ must be to the left of the center of $V$. On the other hand, consider the arc of $U_r$ on the left side of $\overline{pq}$, the center of $U_r$ must be to the right of the center of $V$. This is a contradiction.

Case 2: Assume that only $p$ is on the boundary of circle $U_r$. If $U_r$ cannot cover either $V_+$ or $V_-$, circle $U_r$ and circle $V$ must have at least an intersecting point on the left side of $\overline{pq}$ and at least an intersecting point on the right side of $\overline{pq}$. This contradicts the fact that three points determine a circle.

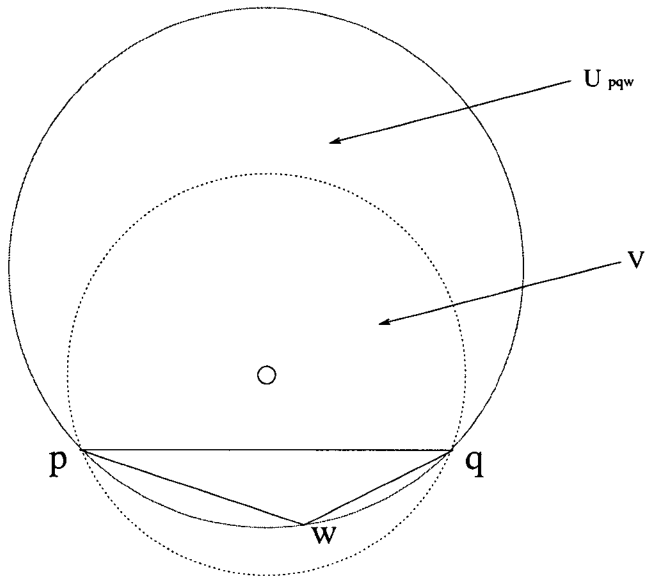


Fig. 15. Proof of Lemma III.6.2: $w \in S$ achieves the minimum angle , and $U_{pwq}$ is the circle passing through $p, q$, and $w$.

Case 3: Assume that neither $p$ nor $q$ is on the boundary of circle $U_r$. If $U_r$ can cover neither $V_+$ nor $V_-$, circle $U_r$ and circle $V$ must have two intersecting points on the left side of $\overline{pq}$ and two intersecting point on the right side of $\overline{pq}$. This contradicts the fact three points determine a circle. □

**Lemma III.6.2** *Let $V$ be a circle that contains $T$. There are two nodes $p, q \in T$ lying on the boundary of $V$. The chord $\overline{pq}$ divides $V$ into two parts $V = V_+ \cup V_-$, where $V_+$ is the larger part in terms of area. Let $r$ be a node in $T$ such that $|pr| \leq D$ and $|qr| \leq D$. Let $U_r$ be the circle of radius $D$ centered at $r$. Suppose that there is a node $r \in T$ such that $V_+ \subset U_r$, then there exists a subset $T_d$ of $T$ with property $P2$ and $|T_d| \leq 3$ .*

Proof. Let $S$ be the set of nodes of $T$ that lie in $V_-$, but not on the chord $\overline{pq}$. If $S$ is empty, then we are done. Otherwise, we prove by **induction on the number of nodes in $S$.**

Let $w \in S$ be the node such that the angle $\angle pwq$ achieves the *minimum* for any $w \in S$. (see Figure 15)

Let $U_{pwq}$ be the circle that passes through $p, q$, and $w$. Our choice of $w$ implies that $T \subset U_{pwq}$.

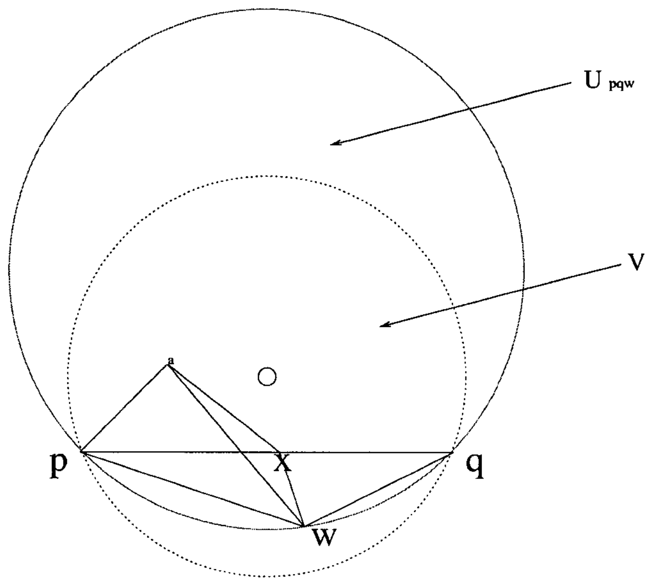


Fig. 16. Proof of Lemma III.6.2 case (1): $x$ is located outside $\triangle paw$.

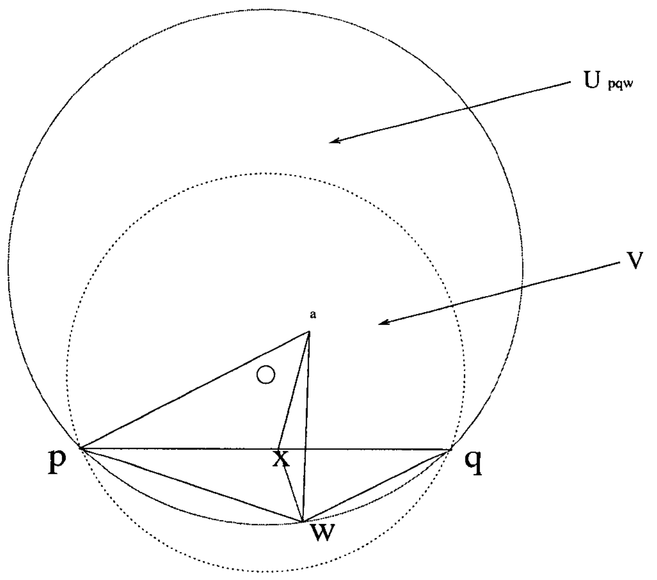


Fig. 17. Proof of Lemma III.6.2 case (2): $x$ is located inside $\triangle paw$.

Let $a \in T$ be the node such that $|pa| \leq D$ and $|wa| \leq D$. Also let $U_a$ be the circle of radius $D$ centered at $a$. Let $\widehat{pw}$ and $\widehat{wqp}$ be two arcs of the boundary of $U_{pwq}$ divided by the chord $\overline{pw}$. According to Lemma III.6.1, we have: either (a) $\widehat{pw} \subset U_a$, or (b) $\widehat{wqp} \subset U_a$.

If (b) is true, then we can replace $V$ with $U_{pwq}$, replace node $r$ with node $a$, replace $p, q$ with $p, w$. We have reduced the number of nodes in $S$ by *one* (point $w$) and hence it follows from induction hypothesis that Lemma III.6.2 holds.

Similarly, let $b \in T$ be the node such that $|wb| \leq D$ and $|qb| \leq D$. Then either $\widehat{wq} \subset U_b$ or $\widehat{qpw} \subset U_b$. Again, if $\widehat{qpw} \subset U_b$ is true, it follows from induction hypothesis and we are done.

So, the remaining case is that: both $\widehat{pw} \subset U_a$ and $\widehat{wq} \subset U_b$. In the remaining of this proof, we are going to prove that in this case, $T \subset U_a \cup U_b \cup U_r$.

Let $V_{pwq} \subset U_{pwq}$ be the region bounded by $\widehat{pwq}$ and chord $\widehat{pq}$. Then $T \subset V_{pwq} \cup V_+$. Since we assume that $V_+ \subset U_r$, it is enough to show that $V_{pwq} \subset U_a \cup U_b$. Let $x$ be the midpoint of the chord $\overline{pq}$. If we can show that $x \in U_a$ and $x \in U_b$, then the convex region bounded by $\widehat{pw}$, $\overline{xp}$, and $\overline{wx}$ lies in $U_a$, and the convex region bounded by $\widehat{wq}$, $\overline{qx}$, and $\overline{xw}$ lies in $U_b$; hence $V_{pwq} \subset U_a \cup U_b$. So it is enough to show that $|ax| \leq 2$ and $|bx| \leq 2$.

We first prove that $|ax| \leq 2$. Note that $a$ can be chosen to be any point in $T$ satisfying $|pa| \leq 2$ and $|wa| \leq 2$. So if $|pw| \leq 2$, we may choose $a = p$. Similarly, if $|wq| \leq 2$, we choose $b = q$.

Since $|pq| \leq 4$, so for every point $y \in V_-$, we have $|xy| \leq 2$. So if $a \in V_-$, then $|ax| \leq 2$, and we are done. Also, if $|pw| \leq 2$, then $a = p$ and $|ax| \leq 2$, and we are done.

The remaining case is that $a \in V_+$ and $|pw| > 2$. There are two sub-cases here based on whether $x$ is located inside $\triangle paw$.

Case (1): $x$ is located outside $\triangle paw$ (see Figure 16). Assume $|ax| > 2$. Then we have $|ax| + |pw| > 4$. Consider the quadrilateral $apwx$, we have $|px| + |aw| > |ax| + |pw|$. Hence $|px| + |aw| > 4$. However this is impossible since $|px| \leq 2$ and $|aw| \leq 2$.

Case (2): $x$ is located inside $\triangle paw$ (see Figure 17). Assume $|ax| > 2$. In $\triangle pax$, we have $|px| \leq 2, |ap| \leq 2$, so $\angle apx > \pi/3$. In $\triangle wax$, we have $|wx| \leq 2, |aw| \leq 2$, so $\angle awx > \pi/3$. So, In $\triangle paw$, $\angle paw < \pi/3$. However, since $|pw| > 2, |ap| \leq 2, |aw| \leq 2$, we have $\angle paw > \pi/3$. This is a contradiction.

Hence $|ax| \leq 2$. Similarly we can prove that $|bx| \leq 2$.

□

**Lemma III.6.3** *Let $V$ be the smallest circle that contains $T$, and there are three nodes $p, q, r \in T$ lying on the boundary of $V$. Among the chord $|pq|$, $|qr|$, and $|pr|$, if at least two are $\leq D$, then $V$ is covered by at most three nodes in $T$.*

Proof. Without loss of generality, we assume that $|pr| \leq D$, and $|qr| \leq D$. Also we assume that $|pr| \leq |qr|$. Now we draw a circle with $U_r$ with $r$ as center and $|qr|$ as radius. The area covered by $U_r$ includes the following three parts: (1) the area bounded by $\widehat{pr}$ and $\overline{pr}$; (2) the area bounded by $\widehat{qr}$ and $\overline{qr}$; (3) $\triangle pqr$.

According to Jung's Theorem [33], the center of circle $V$ must be located inside $\triangle pqr$. This means that $U_r$ covers the *larger* part of $V$. Based on Lemma III.6.2, $V$ can be covered by at most three nodes in $T$.

□

**Theorem III.6.4** *Let $T$ be set of nodes with property $P1$, then there exists a subset $T_d$ of $T$ with property $P2$ and $|T_d| \leq 3$.*

Proof. Let $V$ be the *smallest* circle that contains $T$. By Jung's Theorem [33], we know that one of the following holds:

**Case (1)**: There are *two* nodes $p, q \in T$ lying on the boundary of $V$, and $|pq|$ is the diameter of $V$;

**Case (2)**: There are *three* nodes $p, q, r \in T$ lying on the boundary of $V$, and the center of $V$ lies inside the triangle $\triangle pqr$.

Let $R$ be the radius of $V$. In case (1), it is obvious that $R = d(T)/2 \leq 2$. In case (2), it can be shown that $R \leq d(T)/\sqrt{3} \leq 4/\sqrt{3}$. (The equality holds when $|pq| = |qr| = |rp| = d(T) = 4$.)

For the case (1) in Theorem III.6.4, it follows directly from the Lemma III.6.2 since node $r$ always exists.

For the case (2) in Theorem III.6.4, there are three nodes $p, q, r \in T$ lying on the boundary of $V$. Let $x, y, z \in T$ be the nodes such that $|pz|, |qz|, |qx|, |rx|, |ry|, |py| \leq D$. Let $U_x, U_y, U_z$ be the circles of radius $D$ centered at $x, y, z$, respectively. (see Figure 18)

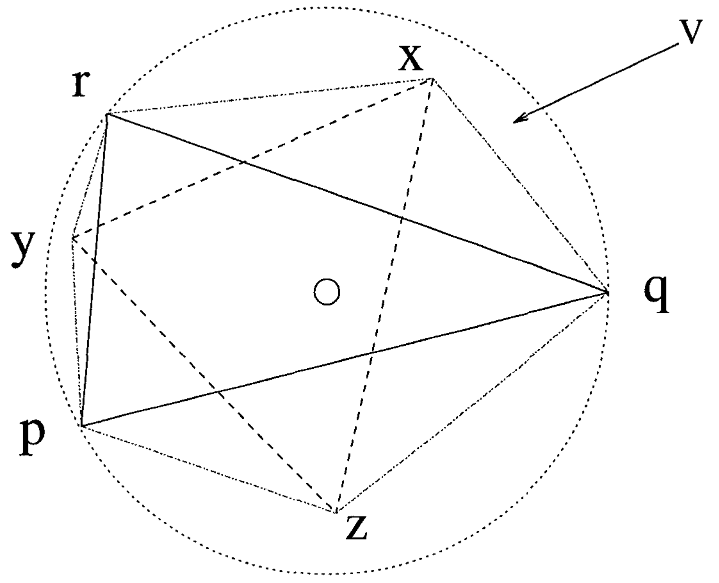


Fig. 18. Proof of Theorem III.6.4: $x, y, z \in T$ are three points such that $|pz|, |qz|, |qx|, |rx|, |ry|, |py| \leq 2$.

Let $\widehat{pq}, \widehat{qr}, \widehat{rp}$ be the arcs of the boundary of $V$ such that $\widehat{pq} \cup \widehat{qrp} = \widehat{qr} \cup \widehat{rpq} = \widehat{rp} \cup \widehat{pqr} = boundary(V)$.

Since $q, r \in U_x$, we have either (a) $\widehat{qr} \subset U_x$ or (b) $\widehat{rpq} \subset U_x$; similarly, either (a) $\widehat{rp} \subset U_y$ or (b) $\widehat{pqr} \subset U_y$ ; either (a) $\widehat{pq} \subset U_z$ or (b) $\widehat{qrp} \subset U_z$ .

If any one of the above three (b)s is true, Theorem III.6.4 follows directly from Lemma III.6.2.

So, in the following, we assume $\widehat{qr} \subset U_x$ , $\widehat{rp} \subset U_y$, and $\widehat{pq} \subset U_z$.

Further, if $x$ and $y$ are the same node, then $x$ covers $\triangle pqr$. Since the center of $V$ is located inside $\triangle pqr$, $x$ covers the *larger* part of $V$. According to Lemma III.6.2, we are done.

Hence in the following, we assume that $x, y, z$ are three *unique* nodes.

It remains to prove that $\triangle pqr \subset U_x \cup U_y \cup U_z$.

To show that $\triangle pqr \subset U_x \cup U_y \cup U_z$, it suffices to show that there does *not* exists a point $w$ inside $\triangle pqr$ such that $|wx| > 2, |wy| > 2$, and $|wz| > 2$.

**We prove by contradiction. Assume that there exists a point $w$ inside $\triangle pqr$ such that $|wx| > 2, |wy| > 2$, and $|wz| > 2$.**

We have three cases based on the length of edges of $\triangle pqr$: (1) at least two of them $\leq 2$; (2) exactly two edges $> 2$; (3) all three edges $> 2$.

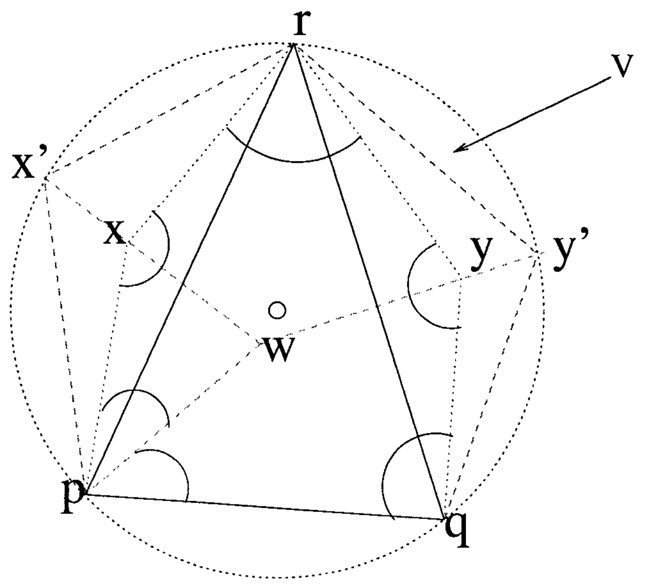


Fig. 19. Proof of Theorem III.6.4 case (2): among the three edges of $\triangle pqr$, exactly two edges ($|pr|, |qr|$) are longer than two.

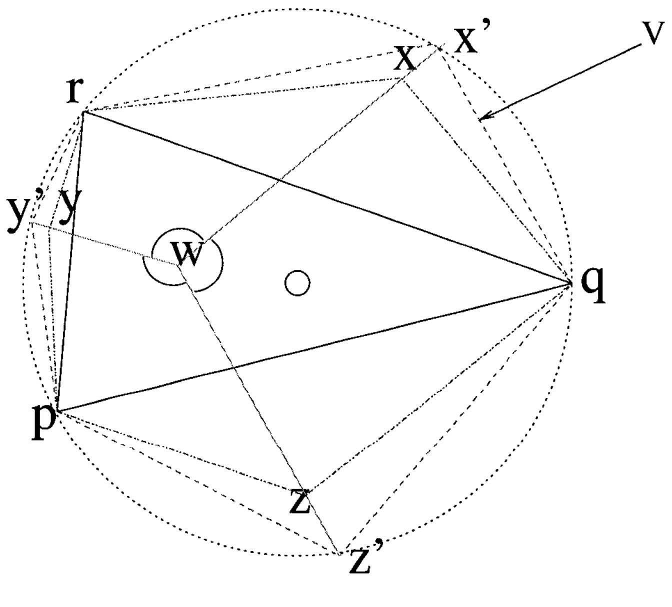


Fig. 20. Proof of Theorem III.6.4 case (3): all three edges of $\triangle pqr$ are longer than two.

For case (1), by Lemma III.6.3, we are done.

For case (2), assume $|pr| > 2, |qr| > 2, |pq| \leq 2$. (see Figure 19). We choose $p$ to be $z$. We have: $\angle pqy > \angle pwy$, $\angle xry > \angle xwy$, $\angle xpw > \angle xwp$. So, $\angle pqy + \angle xry + \angle xpw > \angle pwy + \angle xwy + \angle xwp = 2 * \Pi$. In pentagon $pqyrx$, we have $\angle pqy + \angle xry + \angle xpw + \angle pxy + \angle qyr + \angle wpq = 3 * \Pi$. So $\angle pxy + \angle qyr + \angle wpq < \Pi$.

On the other hand, on the boundary of $V$, we have $\angle pxy \geq \angle px'r = \widehat{pqr}/2 \geq \Pi/2$, $\angle qyr \geq \angle qy'r = \widehat{rpq}/2 \geq \Pi/2$. So $\angle pxy + \angle qyr \geq \Pi$. We have a contradiction.

For case (3), see Figure 20. We have $\angle ypz + \angle xqz + \angle xyr > 2 * \Pi$. In hexagon $pzqxyr$, we have $\angle ypz + \angle xqz + \angle xyr + \angle pzq + \angle qxr + \angle ryp = 4 * \Pi$. So, $\angle pzq + \angle qxr + \angle ryp < 2 * \Pi$. However, $\angle pzq + \angle qxr + \angle ryp \geq \angle pz'q + \angle qx'r + \angle ry'p = 2 * \Pi$. This is a contradiction.

□

**Theorem III.6.5** *Consider a graph $G$ and an arbitrary edge $e$ of $G$. Let $G' = G - e$ be the graph obtained from $G$ by removing edge $e$. If $G'$ is connected, then $|MDS(G')| = |MDS(G)|$ or $|MDS(G')| = |MDS(G)| + 1$.*

Proof. Assume that the edge $e = (u, v)$ is removed.

First, if neither $u$ nor $v$ is in the $MDS$, then the removal of $e$ does not affect the dominance property of the $MDS$, and any $MDS$ in $G$ is still a $MDS$ in $G'$, so we have $|MDS(G')| = |MDS(G)|$.

Second, if both $u$ and $v$ are in the $MDS$, then it is obvious that $|MDS(G')| = |MDS(G)|$.

Third, if exactly one of $u$ and $v$ is in the $MDS$ of $G$. Let assume that $u$ is in $MDS$ of $G$. Then, if $v$ is not dominated by $u$ in $G$, it is obvious that $|MDS(G')| = |MDS(G)|$. If $v$ is dominated by $u$ in $G$, and if $v$ can also be dominated by another node in $MDS$, then we have $|MDS(G')| = |MDS(G)|$. If $v$ is dominated by $u$ in $G$, but it cannot be dominated by another node in $MDS$, then we need to add $v$ into the $MDS$, and $MDS \cup v$ is a dominating set in $G'$. So, $|MDS(G')| = |MDS(G)|$ or $|MDS(G')| = |MDS(G)| + 1$. □

**Theorem III.6.6** *Consider any diameter-2 graph $G$ and an arbitrary edge $e$ of $G$. Let $G' = G - e$ be the graph obtained from $G$ by removing edge $e$. If $G'$ is connected, then $|MDS(G')| \leq 4$.*

Proof. This theorem follows immediately from Theorem III.6.4 and Theorem III.6.5.

□

**Theorem III.6.7** *There exists a unit-disk diameter-2 graph $G$ such that* $|MDS(G)| = 3$.

Proof. We have written a Java program to generate random unit-disk diameter-2 graphs, and in the one million instances of graphs that was generated by the program, all can be dominated by two nodes. This suggests that the probability that a unit-disk diameter-2 graph is dominated by two nodes is very high.

On the other hand, we have been able to construct a counter example that cannot be dominated by two nodes. Consider the unit-disk diameter-2 graph shown in Figure 21. In the figure, $D = 10000$ unit, and the coordinates of nodes are shown in Table 3. This counterexample is inspired by Figure 11 in [33]. It can be verified by hand or program that this *unit-disk* graph is *diameter-2* but cannot be dominated by *two* nodes. □

TABLE 3
Coordinates of the Nodes in Figure 21

| Node | Coordinates (x, y) |
|---|---|
| 0 | (0,0) |
| 1 | (-10000, 0) |
| 2 | (10000,0) |
| 3 | (-8710, -4943) |
| 4 | (8710, -4943) |
| 5 | (-5080, 1360) |
| 6 | (5080, 1360) |
| 7 | (0, -17320) |
| 8 | (-3630, -13737) |
| 9 | (3630, 13737) |
| 10 | (-5000, -8660) |
| 11 | (5000, -8660) |
| 12 | (-2540, -7652) |
| 13 | (2540, -7652) |
| 14 | (-360, -2629) |
| 15 | (360, -2629) |
| 16 | (-2900, -7034) |
| 17 | (2900, -7034) |

From Theorem III.6.4 and Theorem III.6.7, we know that any *unit-disk* diameter-2 graph can be dominated by at most *three* nodes. Note that this is not true for

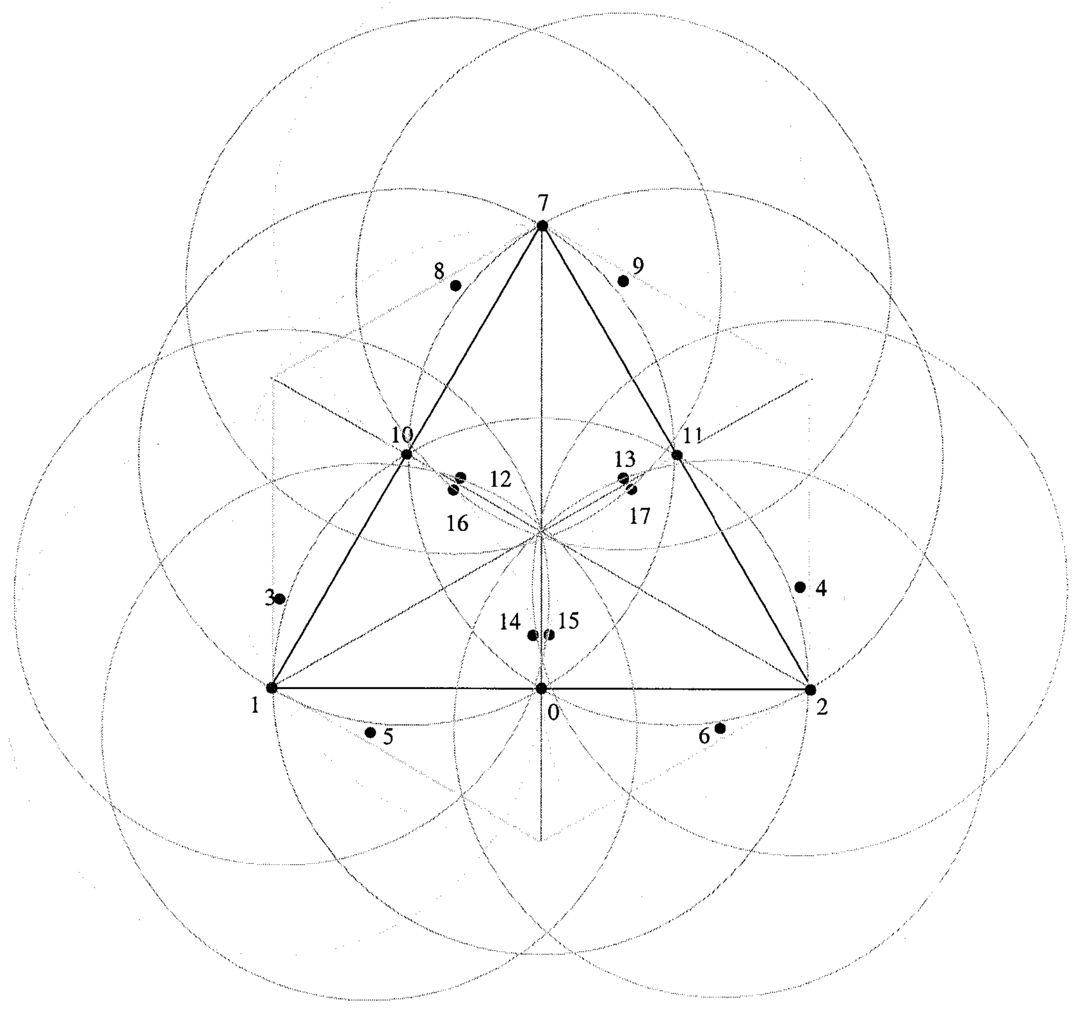


Fig. 21. A unit-disk diameter-2 graph that cannot be dominated by two nodes.

a *general* diameter-2 graph. For a *general* diameter-2 graph, there is no proved constant upper bound on the size of its $MDS$. In the examples shown in [46], there is a *general* diameter-2 graph with 198 nodes and max node degree 16, so its $|MDS| \geq 12$.

## III.7 SUMMARY

A large number of clustering schemes for MANET have been proposed in the recent literature. In general, we believe that a clustering scheme that can generate a more *stable* and *symmetric* virtual infrastructure is especially suitable for MANET, and such a virtual infrastructure can be leveraged by a number of MANET applications without introducing traffic bottlenecks and single points of failure.

To illustrate the feasibility of this concept we have proposed a tree-based cluster initialization/maintenance algorithm for MANET based on a number of properties of diameter-2 graphs. The resulting algorithm is cluster-centric and works in the presence of node mobility. Simulation results demonstrated the effectiveness of our algorithm when compared to other clustering schemes in the literature.

# CHAPTER IV

# A TWO-ZONE HYBRID ROUTING PROTOCOL

In this chapter, we propose Two-Zone Routing Protocol (TZRP) as a general hybrid routing framework that can balance the tradeoffs between pure proactive, fuzzy proactive, and reactive routing approaches more effectively in a wide range of network conditions. In contrast with the original Zone Routing Protocol (ZRP)[31, 58] where a single zone serves a dual purpose, TZRP aims to decouple the framework's ability to adapt to traffic pattern from the ability to adapt to mobility. In TZRP, each node maintains two zones: a Crisp Zone for proactive routing and efficient bordercasting, and a Fuzzy Zone for heuristic routing using imprecise locality information. The perimeter of the Crisp Zone is the boundary between pure proactive routing and fuzzy proactive routing, and the perimeter of the Fuzzy Zone is the boundary between proactive routing and reactive routing. By adjusting the sizes of these two zones, a reduced total routing control overhead can be achieved. Further, TZRP can be considered to be a general MANET routing framework that can balance the tradeoffs between pure proactive, fuzzy proactive, and reactive routing approaches more effectively in a wide range of network conditions.

The remainder of this chapter is organized as follows. Section IV.1 begins by motivating the need to decouple concerns about traffic characteristics and mobility in ZRP. Section IV.2 presents a high-level overview of TZRP. The details of TZRP are discussed in Section IV.3. Section IV.4 presents our simulation results showing that TZRP outperforms ZRP. Finally, Section IV.6 offers concluding remarks and maps out directions for further investigations.

## IV.1 MOTIVATION

Although the Zone Routing Protocol (ZRP) provides an elegant and powerful hybrid routing framework, the choice of the specific proactive or reactive protocols used therein is of key importance. In fact, the *bordercasting* mechanism — the key component of ZRP — has some very important implications on ZRP's IARP (IntrA-zone Routing Protocol) component: the IARP must be able to provide *up-to-date* topology information of the routing zone.

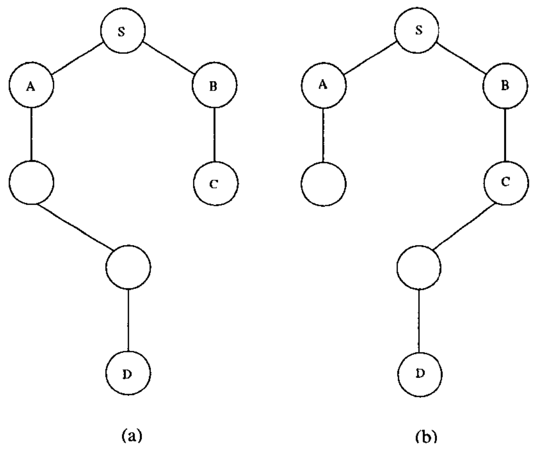


Fig. 22. Inaccurate zone topology information can lead to query failure.

Consider the scenario illustrated in Figure 22, and assume $ZR = 3$. The actual topology is shown in Figure 22(a). However, due to an IARP that fails to provide accurate zone topology information, the topology perceived by nodes $S$, $A$, and $B$ is that of Figure 22(b). When $S$ wants to find a route to $D$, it constructs its bordercast tree. S prunes A from its RREQ receiving set and sends the RREQ only to $B$. $B$ further forwards it to $C$, which has no choice but to terminate the query thread. Hence the query procedure fails. Since the source $S$ has to wait for an amount of time that is proportional to the expected network diameter before realizing the query failure and trying again, such a query failure can cause significantly longer route acquisition latency. This example illustrates the importance of the freshness and consistency of the IARP information maintained at each node. Indeed, bordercasting in ZRP requires an IARP that converges very fast, implying that the distance vector variants and the long-timer-based link state variants are generally not suitable to work as IARP. By contrast, the event-driven link-state approach is the ideal choice. However, in the bandwidth-limited MANET, frequent topology changes make a pure event-driven implementation infeasible. Thus, most link-state approaches are implemented in a timer-based fashion [37, 69]. That is, a LSU is sent out only at some specific intervals. The smaller the interval, the shorter the convergence time, and the better approximation of an event-driven link-state routing can be achieved.

As discussed in [58], both mobility and traffic pattern influence the routing control overhead and, hence, the optimum configuration of the zone radius. In a high-mobility scenario, the event-driven IARP incurs a very large proactive control overhead. This drives toward a smaller zone radius. However, reducing the zone radius also reduces the initial query hit ratio since more nodes will be outside of the node's immediate knowledge. For example, assume that node $x$'s zone (with a radius of five) is divided into four areas: $A$, $B$, $C$, and $D$. Using an IARP that can approximate event-driven link-state routing reasonably well, some nodes in area $A$ and $C$ are moving so fast that many LSUs need to be generated. These LSUs are received by $x$, and when such proactive traffic is too large, $x$ will decrease its zone radius to, say, four. The result is that $x$ no longer proactively maintains routing information to its 5-hop neighbors, even though these 5-hop neighbors are quite stable with respect to $x$. When $x$ needs to find a route to one of these nodes, a global bordercasting is required. Note that, although ZRP has several mechanisms to terminate a query thread as early as possible [31], asymptotically, once the query goes out of the initial zone, at least half of the network will be flooded [68]. Consequently, bordercasting is still an expensive procedure compared to an immediate available route, hence should be avoided as much as possible.

Basically, the single zone structure of the original ZRP framework is intended to serve a *dual* purpose simultaneously as far as reducing routing control overhead is concerned: ($a$) it maintains routes to nearby nodes proactively so that local traffic can be routed immediately; in scenarios featuring *traffic locality*, this can result in a significant reduction in reactive control overhead since it avoids global search to a great extent; ($b$) it provides a structure that can be exploited to achieve efficient flooding (bordercasting) when a global search is necessary. The key problem with this framework is that although *accurate* topology information of the circular shape (instead of any other shape) zone is necessary for purpose ($b$), such information is not necessary for purpose ($a$).

In fact, bordercasting is used to find a route to a destination whose location is *unknown* to the source. This implies that bordercasting serves a *global* purpose and the protocol must ensure that a query passes through even the *weakest* part of the network and reaches the destination's zone. Hence, inaccurate topology information used by bordercasting nodes to prune their bordercast trees may terminate a query prematurely, causing a bad global effect. On the other hand, taking advantage of traffic characteristics to reduce routing overhead serves a *local* purpose. As demonstrated by FSLS[69], FSR, and GSR[37], reduced frequency and accuracy in LSU generation and propagation work well in making a local decision on the next hop to a distant node, and history routing information to a distant node provides a good approximation for the current route to that node.

Understanding the requirements for information accuracy of different components of a hybrid MANET routing protocol like ZRP is important since accurate topology is inherently expensive to maintain in MANET and hence should be limited to small scope. The high sensitivity to mobility renders the zone structure of ZRP less useful as a means of adapting to changing traffic patterns when mobility becomes high. This motivate us to find a companion structure that works well to achieve fine tuning of the total routing control overhead when high mobility forces the zone radius to be small.

## IV.2 BASIC IDEA OF TZRP

In outline, the basic idea of TZRP is as follows: each node $x$ maintains two zones, both with $x$ as center. One is the *Crisp Zone*, with radius $ZR_c$, the other is the *Fuzzy Zone*, with radius $ZR_f$. We always have $ZR_c \leq ZR_f$. Node $x$ maintains proactively the up-to-date topology of its Crisp Zone; however $x$ does not have to know the exact topology of its Fuzzy Zone. Instead, a *fuzzy-sighted*-like proactive routing protocol [69] is employed as the IARP in node $x$'s Fuzzy Zone.

In a low-mobility scenario where topology changes occur infrequently, a large Crisp Zone can be maintained with little proactive overhead. In such a case, we have $ZR_c = ZR_f$, which is the same as the original ZRP. In a high-mobility scenario where it is too costly to maintain a large Crisp Zone, $ZR_c$ is reduced to a smaller size. However, since the control overhead involved in maintaining the Fuzzy Zone is long-timer based and, thus, largely independent of the node's mobility pattern, a large $ZR_f$ can be maintained. This implies that the traffic locality benefit is still preserved to a great extent due to fuzzy proactive routing. Essentially, TZRP aims to decouple the framework's ability to adapt to traffic pattern from its ability to adapt to mobility. The Crisp Zone is used to balance the influence of mobility on the routing control overhead, while the Fuzzy Zone is used to balance the influence of traffic pattern on the routing control overhead. By adjusting these two radii, a lower total routing control overhead can be achieved.

Although TZRP is proposed as an extension of ZRP, it bears some resemblance to FSLS/FSR. However, there is a major difference: TZRP has a reactive component, while FSLS does not. This difference implies that in FSLS, each node maintains a routing table with $\Theta(N)$ entries, while in TZRP, each node maintains a routing table with $\Theta(n + e)$ entries ($N$ is the number of nodes in the network, $n$ is the number of nodes in the Fuzzy Zone, and $e$ is the number of active nodes that are out of the Fuzzy Zone). In FSLS, as time evolves and nodes move, a node will learn of the failure of previously computed routes due to links going down; however, the node will *not* learn in a *timely* manner of new routes formed due to the long update interval of the information about the far-away nodes. In a less dense network where there are fewer alternative routes between nodes, this can lead to unnecessary data packet droppings even when there exists a route to the destination. In fact, this problem is common to every protocol in the FSLS family [70]. TZRP effectively solves this problem by using a reactive component.

It is important to realize that what really differentiates ZRP from FSLS is the underlying assumption about the traffic pattern. *Traffic locality* is a key assumption of ZRP, while a *uniform traffic* pattern across the entire network is assumed by FSLS(HSLS). By including a fuzzy proactive component to ZRP, and by including a Crisp Zone-based reactive component to FSLS, TZRP effectively takes advantage of the benefits of both protocols under the guideline of *making the common case fast, and making the rare case correct* [57]. When traffic locality holds, TZRP works similarly to ZRP but is more adaptive to high mobility, and when traffic locality cannot be assumed, TZRP works similarly to FSLS but reduces the chances of loops and data packet droppings by an efficient reactive component. The Crisp Zone perimeter is the boundary between pure proactive routing and fuzzy proactive routing, and the Fuzzy Zone perimeter is the boundary between proactive routing and reactive routing. In general, ZRP is a special case of TZRP where $ZR_f = ZR_c$; and FSLS is a special case of TZRP where $ZR_f = \infty$ without a reactive component. Thus, TZRP can be considered to be a general MANET routing framework that can balance the tradeoffs between pure proactive, fuzzy proactive, and reactive routing approaches more effectively in a wide range of network conditions.

## IV.3 DETAILS OF TZRP

The main goal of this subsection is to provide the details of TZRP. The key difference between TZRP and the original ZRP is that our IARP component explicitly distinguishes between event-driven IARP and timer-based IARP. Specifically, we use a short-timer-based implementation to approximate Crisp IARP, and we use a HSLS-based implementation as Fuzzy IARP. However, we want to emphasize that the Fuzzy IARP can be implemented using any protocol in the FSLS family [70] with different number of *scopes* and *update intervals.*

### IV.3.1 Generation and propagation of LSU

A node $x$ counts the current time $T$ (in seconds), starting from $T = 0s$. It *wakes up* every $t_s$ second, and finds the largest positive integer $i$ such that $T\ mod\ (2^{i-1} * t_e) = 0$. (1) If such a positive $i$ exists, then $x$ checks whether there was a link state change during the last $(2^{i-1} * t_e)$ seconds. If so, then $x$ is in *SendingFuzzyLSU* mode, and $L$ is set to $2^{i-1}$. Further, if $L < ZR_c - 1$, then $L$ is set to $ZR_c - 1$; if $L > ZR_f - 1$, then $L$ is set to $ZR_f - 1$. Finally, a LSU with $TTL = L$ is generated; (2) Otherwise (i.e., such a positive integer $i$ does not exist), $x$ checks whether there was a link change during the last $t_s$ seconds. If so, $x$ is in *SendingCrispLSU* mode, and a LSU with $TTL = ZR_c - 1$ is generated.

The Crisp Zone structure is exploited during the propagation of a LSU. When $ZR_c \geq 2$, each node $x$ maintains its shortest paths to every 2-hop neighbor by exchanging LSUs with TTL=1. Node $x$ uses the minimum number of 1-hop neighbors to cover all its 2-hop neighbors by applying one of the well-known greedy heuristics [77]. The selected 1-hop neighbors form a *forwarding set* for the LSUs received from node $x$. Node $x$ includes this forwarding set information in each LSU it generates or forwards. When a node $y$ receives a LSU for the first time, it integrates this LSU into its link state table. Next, $y$ checks whether it itself appears in the forwarding set of the LSU; if so, and the TTL of the LSU is larger than one, then $y$ decrements the TTL, calculates the forwarding set, appends this forwarding set to the LSU, and forwards it; otherwise, $y$ discards the LSU.

### IV.3.2 Computing Crisp/Fuzzy IARP route

When a node receives a more recent LSU generated by node $s$, it deletes all the existing entries with $s$ as the source or destination in the current link state table, and then inserts the link state entries contained in the LSU just received. When there is a route to be resolved, the intra-zone routing table is recomputed based on the latest link state table. Specifically, node $x$ uses Dijkstra's algorithm to compute a shortest path from $x$ to any other node of which it is aware. All the shortest paths with length of exactly $ZR_c$ hops constitute $x$'s *bordercast tree*, which is used in bordercasting as described in Subsection IV.3.3.

### IV.3.3 Bordercasting

If the destination node is unreachable from node $x$ through either a Crisp IARP route or a Fuzzy IARP route, then a reactive *bordercasting* procedure is invoked. We follow the latest version of BRP described in [32]. Specifically, the nodes that are direct children of node $x$ in the bordercast tree constructed above form the forwarding set for the RREQs received from node $x$. Node $x$ appends the forwarding set to the RREQ and broadcasts it to all its 1-hop neighbors. Upon receiving the first copy of a RREQ, a node determines whether it is a forwarding node by checking the forwarding set information piggy-backed in the RREQ. If a node finds that it is not in the forwarding set, it simply discards the RREQ.

A node $y$ in the forwarding set proceeds to process the RREQ. If there is an IARP route from $y$ to the query destination with length not longer than $ZR_c$ (hence a Crisp IARP route), $y$ unicasts the RREQ to the destination, which then sends a RREP back to the query source, indicating that a route to the destination has been found. Otherwise, node $y$ constructs its bordercast tree in the following way: First, it computes the shortest path tree with $x$ as the root. All the nodes that are $ZR_c$ or fewer hops from $x$ are marked as *covered*. Second, $y$ computes the shortest path tree with $y$ as the root, and all the uncovered nodes as leaves. The paths of length exactly $ZR_c$ hops constitute $y$'s bordercast tree. Then $y$ appends the forwarding set in the RREQ and further forwards it. Finally, $y$ marks all nodes that are $ZR_c$ or fewer hops from $y$ as *covered*.

During the bordercasting procedure, routing information is created and maintained at the involved nodes. We adapt AODV as the IERP (IntEr-zone Routing

Protocol). That is, during the propagation of RREQ, a backward routing entry toward the query source is established at each forwarding node; during the propagation of RREP, a forward routing entry toward the query destination is established at each forwarding node. The destination sequence number is used in the similar way as in AODV to prevent routing loop. Different from AODV, zone information is used for route maintenance. When a link breakage on an active route is detected, the upstream node checks whether the destination node can be reached through any alternative Crisp/Fuzzy IARP route. If so, the route is locally-repaired successfully; otherwise, a RERR is sent back to the source as in AODV.

## IV.4 SIMULATION RESULTS AND DISCUSSION

We have simulated the TZRP protocol using the *ns-2* simulator [55]. In our simulations, we have $N$=200 nodes, each of which has a radio transmission range of $T_r$=250m and transmission rate of 2Mbps. Initially, the nodes are distributed uniformly at random in an area $A$ which is either a square or a rectangle. The nodes move according to the *random way-point model*; in all our simulations, we set the pause time to zero and each node always moves at the fastest speed $V$. The values of $A$ and $V$ vary in different scenarios, as illustrated in Table 4. The node density, $D$, in Table 4 is calculated as $D = \frac{N*\pi*T_r^2}{A}$ and corresponds to the expected degree of a node in the underlying graph. Each simulation begins at time 0 and ends at time 190s. We collect statistics data on various control packets starting at $t$=10s until the end of the simulation.

TABLE 4
TZRP Simulation Scenarios

| Scenario | A (m×m) | V (m/s) | D |
|---|---|---|---|
| 1 | 2000×2000 | 10/20 | 9.81 |
| 2 | 2500×2500 | 10/20 | 6.28 |
| 3 | 4000×1000 | 10/20 | 9.81 |
| 4 | 5000×1250 | 10/20 | 6.28 |

In our simulation, we use IEEE 802.11 DCF MAC (with RTS/CTS) as well as an ideal MAC layer. The only difference between the two is that we assume in the ideal MAC, *broadcast* is reliable and is not impaired by collisions. We choose to present our simulation results mainly using the ideal MAC since this allows us to focus on understanding and analyzing the behavior of ZRP and TZRP without being distracted by cross-layer interactions. In fact, collision-free broadcast is a common assumption in the existing ZRP simulations reported in the literature [58]. In Subsection IV.4.3, we will show our simulation results under IEEE 802.11 MAC. As it turns out, using suitable broadcasting optimizations [80], TZRP's advantage over ZRP demonstrated in the ideal MAC case still holds in the IEEE 802.11 MAC case.

Our protocol relies on periodic HELLO beacons to detect link formations and breakages. The HELLO beacons are sent every 0.1s, and the number of tolerable missed HELLOs is two. In addition, MAC-layer link breakage detection and packet salvage is enabled. We extend the scenario generation tool in *ns-2* to generate traffic based on a given flow distance distribution. By controlling the flow distance, we can clearly identify whether a flow is *intra*-zone or *inter*-zone for a specific scenario. This enables us to determine as the zone radius increases, whether the reduction in total routing control overhead is more attributable to traffic locality or to efficient bordercasting. In addition, the flows generated in this way have a better chance to involve connected nodes.

### IV.4.1 Sensitivity of bordercasting to IARP timer

The goal of this set of simulations is to demonstrate the influence of the IARP timer on the effectiveness of bordercasting. In our implementation, if a RREP is not received within $0.4s$ after the first RREQ for a query is issued, the source node resends the RREQ and doubles its waiting time. After *three* failed attempts, the query is dropped. We calculate the query success ratio at each attempt. Combining this number with the *route acquisition latency* provides a sufficiently good indication of the effectiveness of bordercasting. We note that when calculating the route acquisition latency, only successful queries are considered.

For this set of simulations, we generate 2000 queries during a three-minute simulation for each scenario, and examine those flows whose distance between the source and destination is at least five hops at the instant when the flow is generated at the source. The average route length for each scenario is shown in Table 5. We make the duration time of each flow short and each flow has only one packet to send. The

TABLE 5
Traffic Pattern: Average Route Length of Queries

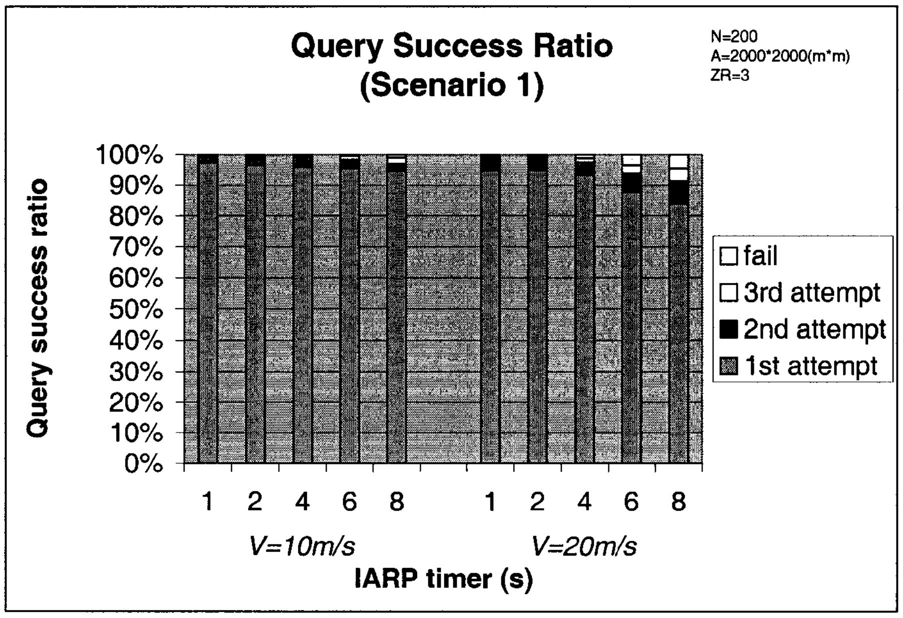
| A (m×m) | V (m/s) | Average Route Length (hops) |
|---|---|---|
| 2000×2000 | 10 | 6.3 |
| 2000×2000 | 20 | 6.3 |
| 2500×2500 | 10 | 7.2 |
| 2500×2500 | 20 | 7.5 |
| 4000×1000 | 10 | 8.2 |
| 4000×1000 | 20 | 8.3 |
| 5000×1250 | 10 | 10.2 |
| 5000×1250 | 20 | 9.9 |

intention is to isolate the effects of various possible route maintenance optimizations and focus on the route discovery procedure only. Also, in this set of simulations, we use pure timer-based IARP (i.e. without the optimization of propagating LSU using the forwarding set) since such an optimization requires accurate 2-hop topology information, which is not the case when the IARP timer is increased. We study the behavior of bordercasting for various values of zone radius, and the results featured in Figures 23, 24, 25, 26 correspond to a value $ZR = 3$.

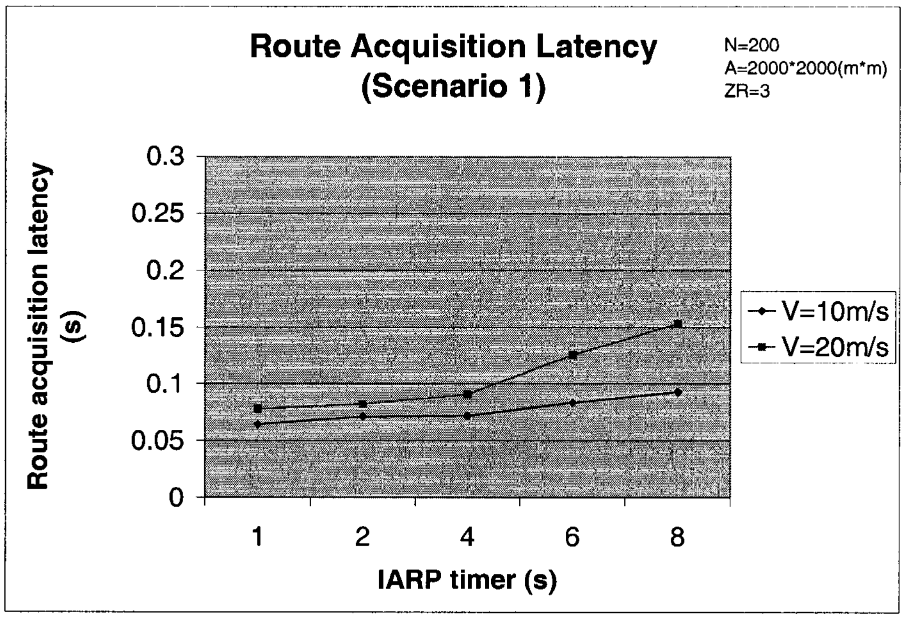
Figures 23 – 26 indicate clearly that *node density*, *node mobility*, and the *Crisp-Zone IARP timer* are the key factors that have a significant influence on the effectiveness of bordercasting.

Notice that when node density is high (see Figure 23), node mobility has relatively little influence on route acquisition latency and on query success ratio of bordercasting. This is because a large number of threads are generated for a single query, and although some of them lose their directions and are terminated prematurely due to the inaccuracy of topology information when mobility is high and/or IARP timer is long, the probability that at least one thread survives and reaches the destination is still high. As a result, the query success ratio is relatively stable, and the route acquisition latency only increases slightly.

However, as the node density decreases (see Figure 24), fewer threads are generated for each query, and the number of alternative routes to a destination decreases as well. In this case, the influence of the accuracy of the zone topology information on the route acquisition latency becomes more and more obvious as mobility increases.

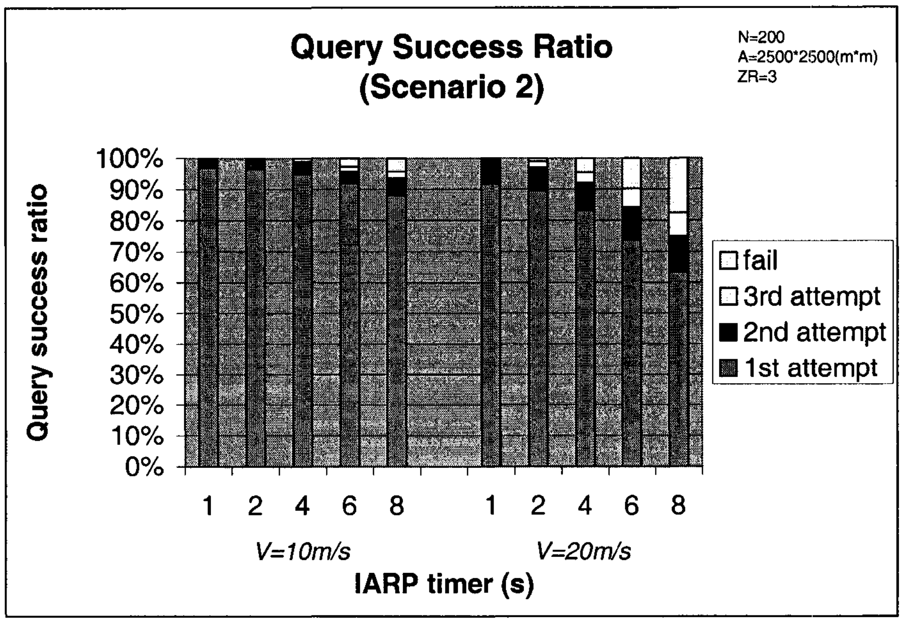


(a) Scenario 1, Query success ratio

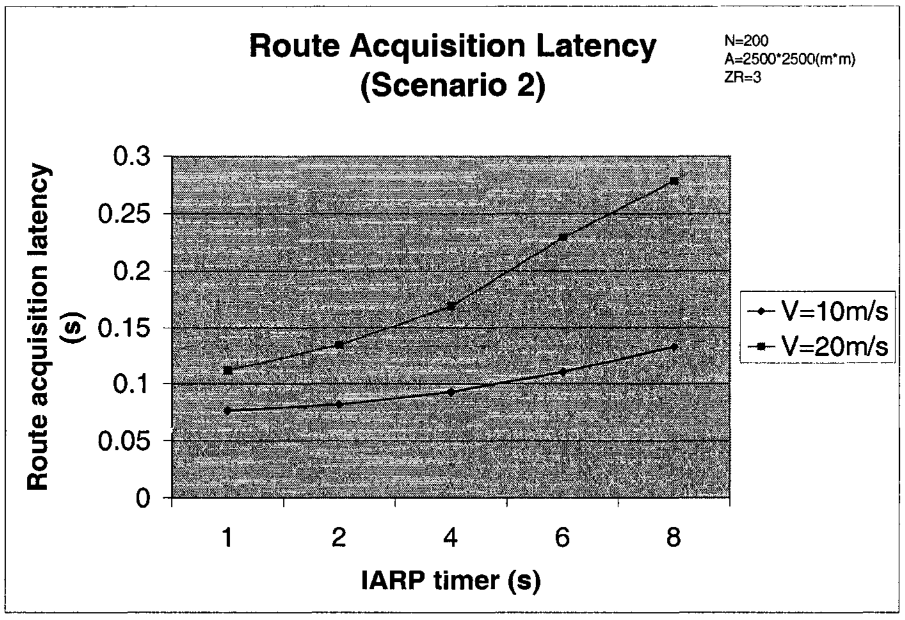


(b) Scenario 1, Route acquisition latency

Fig. 23. The influence of IARP timer on bordercasting: scenario 1.

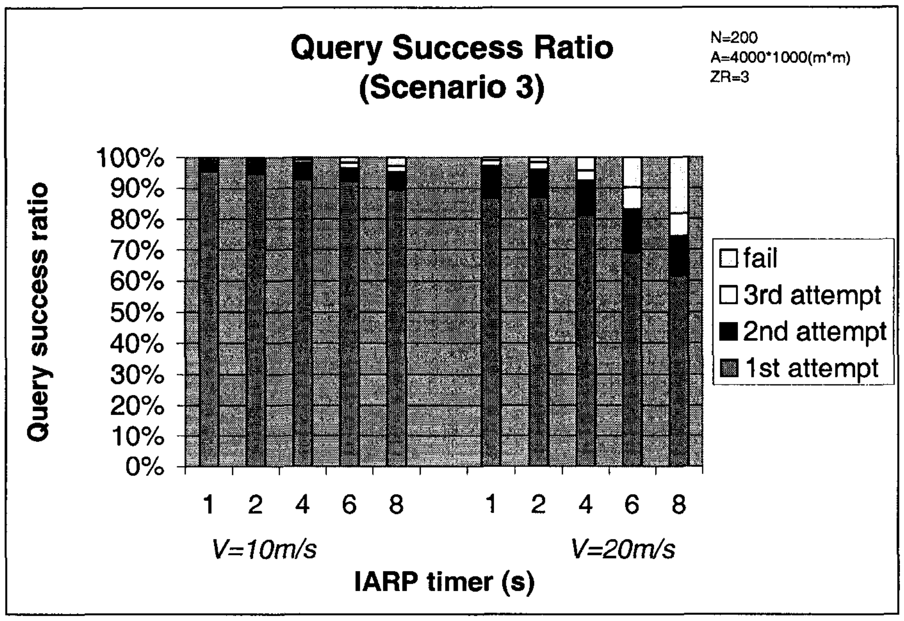


(a) Scenario 2, Query success ratio

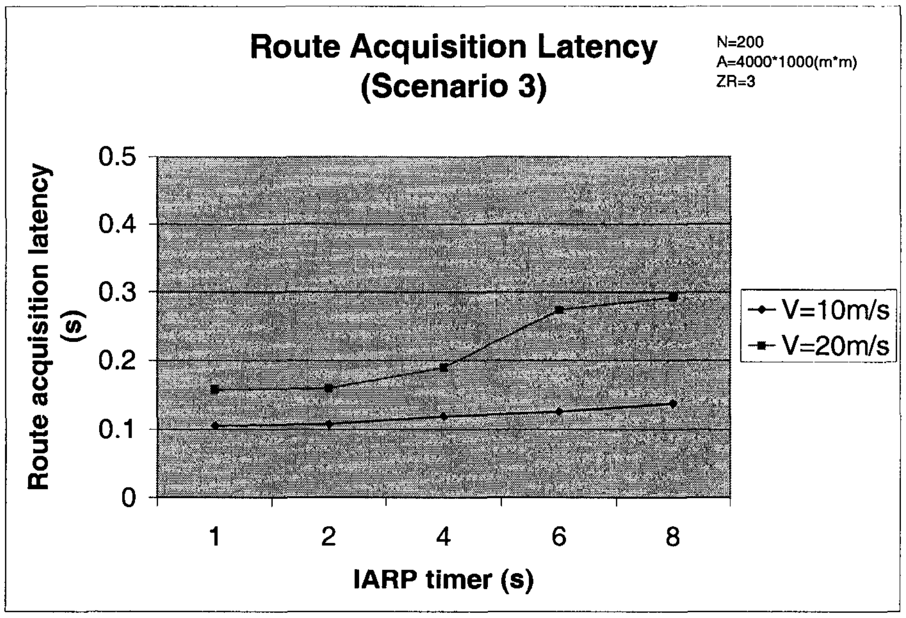


(b) Scenario 2, Route acquisition latency

Fig. 24. The influence of IARP timer on bordercasting: scenario 2.

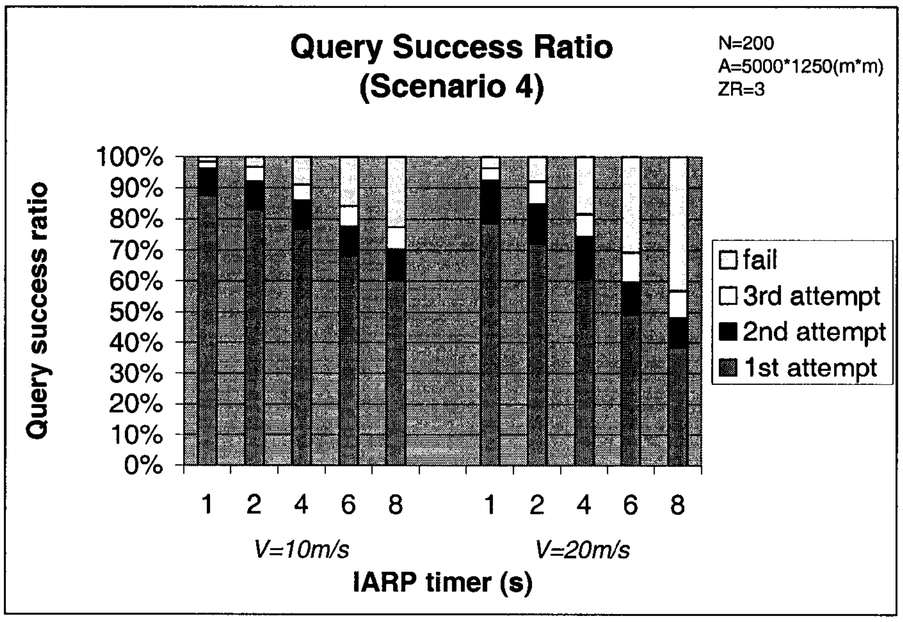


(a) Scenario 3, Query success ratio

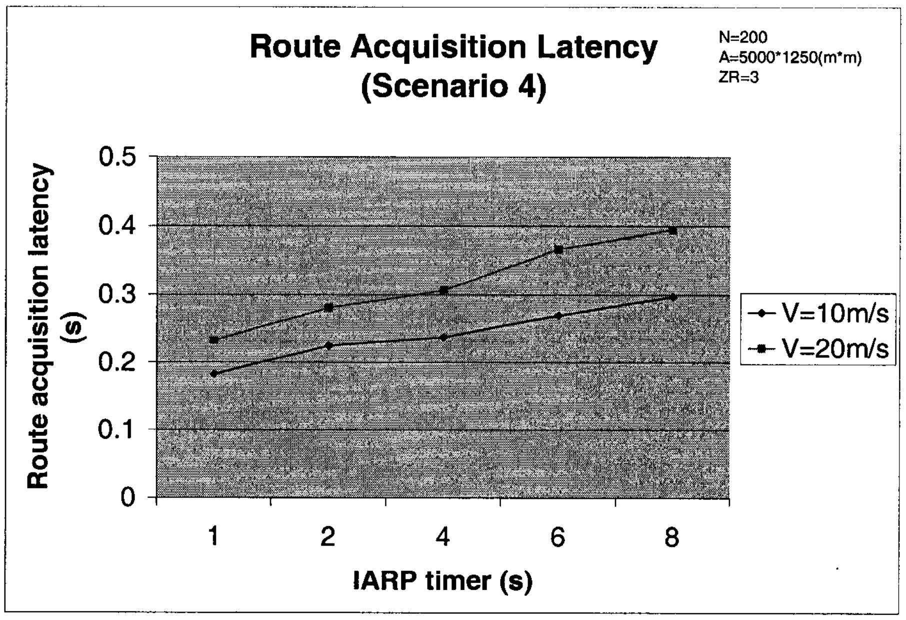


(b) Scenario 3, Route acquisition latency

Fig. 25. The influence of IARP timer on bordercasting: scenario 3.



(a) Scenario 4, Query success ratio



(b) Scenario 4, Route acquisition latency

Fig. 26. The influence of IARP timer on bordercasting: scenario 4.

When mobility is too high, queries may fail even after several retries under a long IARP timer, leading to a significant decrease in the query success ratio.

Simulation shows that bordercasting in rectangular scenarios (see Figures 25 and 26) is more sensitive to IARP timer than in the case of square scenarios. This is because in rectangular scenarios (1) more queries involve further-away nodes, and (2) the number of a node's peripheral nodes is smaller.

These simulation results clearly demonstrate that bordercasting requires accurate zone topology and hence an event-driven IARP (or a short-timer-based IARP), not only for theoretical correctness, but also for practical effectiveness, especially in less dense and/or high mobility scenarios.

### IV.4.2 Performance evaluation of TZRP

In this set of simulations, we demonstrate the performance of TZRP compared to the original ZRP. We use the *total routing control overhead* and *query success ratio* as the representative performance metrics. Within the total control overhead, the number of transmissions (including generation and forwarding) of those LSUs with initial TTL=$ZR_c$-1 is considered *pure proactive overhead*, the number of transmissions of those LSUs with initial TTL $>$ $ZR_c$-1 is considered *fuzzy proactive overhead*, and the *reactive overhead* is the sum of RREQ, RREP, and RERR transmissions. As to *query success ratio*, in our simulation, a data packet is dropped in one of the following three cases: (1) the next hop node is the node from which the packet was received, (2) TTL has expired, or (3) the packet arrives at a node that cannot find a route to the destination after three bordercastings.

Note that in the thrid case, the intermediate node can choose to drop the datat packet immediately as long as there is not an known route and the broen route cannot be locally reparied, without doing bordercasting for this data packet that is not originated by itself. We do not do this in our simulation since our goal is a *reliable* routing protocol in the sense that it does not give up until a packet reaches destination, at the cost of more reactive control overhead. So the *data packet delivery ratio* reported here can be much higher than those implementations that drop data packets more aggressively.

We use Scenario 4, with $V = 20m/s$, $t_s = 1s$, and $t_e = 2s$ (see Subsection IV.3 for the definitions of $t_s$ and $t_e$). Here, $Q$ queries with distance in the range of $H$ are

TABLE 6
Traffic Patterns Used to Demonstrate the Performance of TZRP.

| Traffic | $H$(hops) | Average Route Length (hops) | Number of Queries ($Q$) |
|---|---|---|---|
| T1 | 1-5 | 3.300 | 6000 |
| T2 | 1-5 | 3.293 | 3000 |
| T3 | 1-32 | 7.637 | 2500 |

generated between 10s and 190s simulation time, and each query is 64 bytes. The values of $H$ and the corresponding average route lengths of the three traffic patterns used in the simulation are shown in Table 6. The simulation results for these three traffic patterns are shown in Figures 27, 28, 29, 30, and 31. We also summarize the total control overhead and query success ratio of different scenarios in Table 7.

Since this protocol is an extension of ZRP, we first illustrate a representative scenario that differentiates TZRP from the original ZRP. Under traffic T1, high mobility makes it too costly to maintain a large Crisp Zone, which is reflected in Figure 27 as a significant increase in the pure proactive overhead when ZR increases by one. By comparison, we can notice from Figure 28(a)(b) that the increase of the fuzzy proactive overhead is much less drastic as the Fuzzy Zone radius increases. Hence, by reducing the $ZR_c$ and by keeping a large $ZR_f$, TZRP significantly reduces the reactive routing control overhead, and achieves a better balance between proactive and reactive control overhead than the original ZRP under many $(ZR_c, ZR_f)$ settings, as shown in Figure 29.

The traffic *intensity* of T2 is smaller than T1. In this scenario, the advantage of TZRP over ZRP is less obvious as shown in Figure 30. This is expected since the reactive overhead when $ZR_c = 2$ is already very small, hence the maintenance of a larger Fuzzy Zone does not have much effect on reducing the reactive control overhead.

Compared to T1 and T2, traffic T3 has more *diversity* in terms of the flow distance. From Figure 31 we can see that the total routing control overhead achieved by TZRP is smaller than ZRP even when the Fuzzy Zone radius is very large ($ZR_f = 32$). In fact, under this Fuzzy Zone radius setting, TZRP works similar to HSLS. To see the difference between TZRP and HSLS under this scenario, we implement HSLS with three different $t_e$ values (1s, 2s, 4s). For fairness, we introduce the forwarding

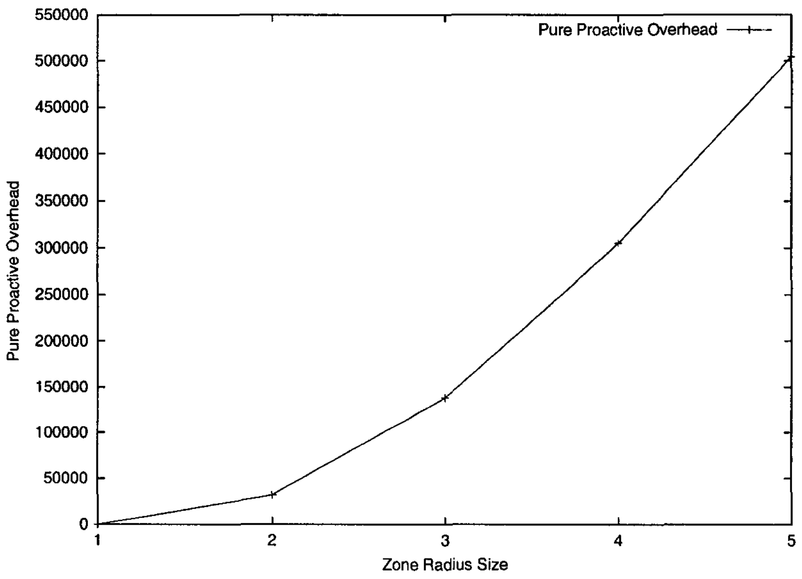
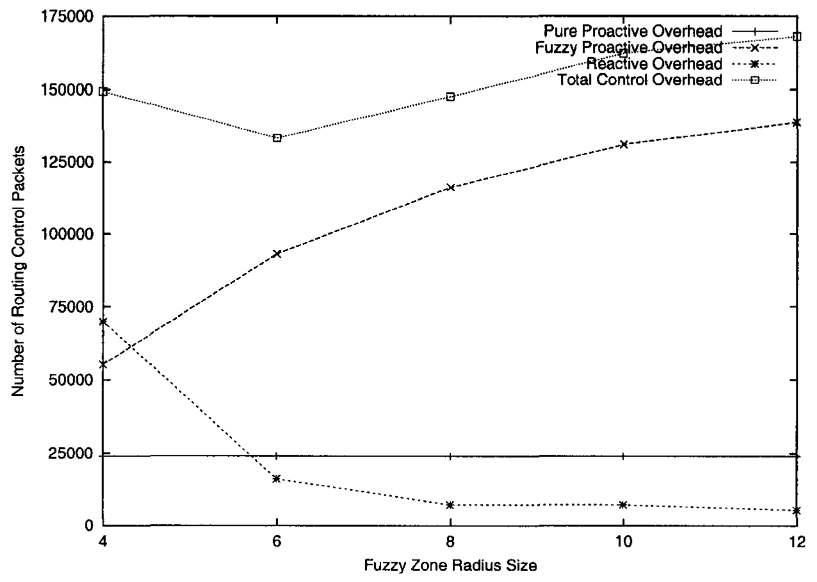


Fig. 27. TZRP performance (traffic T1): pure proactive routing control overhead.

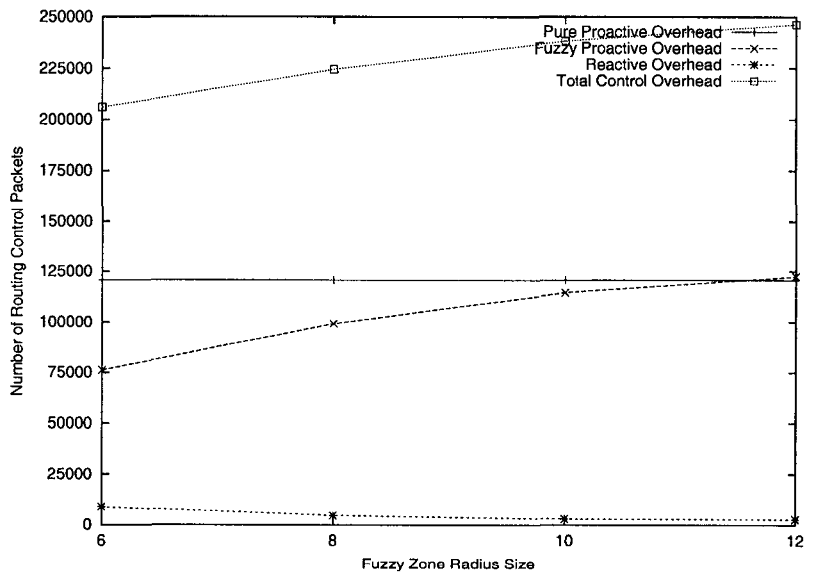
set-based LSU propagation to HSLS, so that the routing control overhead of HSLS is greatly reduced. Accordingly, each node checks whether to send a LSU with TTL = 1 every second, regardless of $t_e$ settings. The performance of HSLS is shown in Table 7. (Note that HSLS requires all the nodes flood LSUs globally once when the simulation just begins. This part of control overhead is not included in the simulation result shown here.) Among the three versions of HSLS, the smaller the update interval, the higher the query success ratio. However, even when $t_e = 1s$, the query success ratio is still significantly smaller than that can be achieved by TZRP, especially in traffic T3 where there are more long-distance flows. This justifies the necessity of the reactive component in TZRP for protocol correctness. Of course, we can further reduce the HSLS update interval to achieve higher query success ratio, but the total routing control overhead of HSLS will also increase significantly, which is not necessary when the traffic demonstrates locality.

### IV.4.3 The influence of MAC on the performance of TZRP

The simulation results presented thus far were based on an ideal MAC. In TZRP (as well as ZRP), the reliability and efficiency of MAC layer broadcast impact both the proactive component and the reactive component of the protocol. This suggests



(a) $ZR_c$=2



(b) $ZR_c$=3

Fig. 28. TZRP performance (traffic T1): fuzzy proactive routing control overhead.

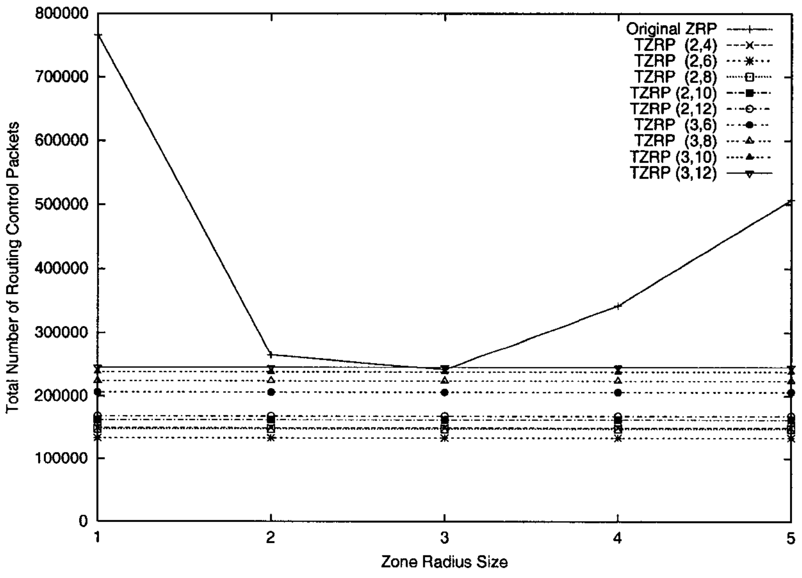


Fig. 29. TZRP performance (traffic T1): total routing control overhead.

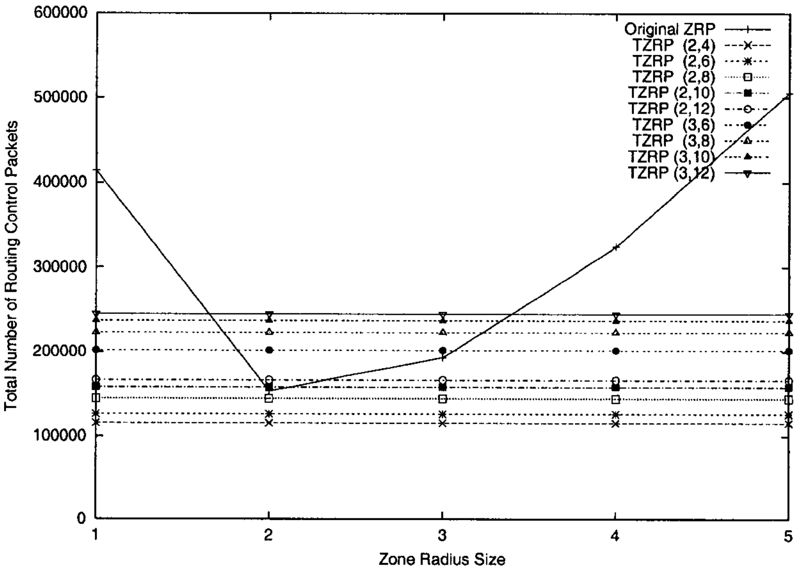


Fig. 30. TZRP performance (traffic T2): total routing control overhead.

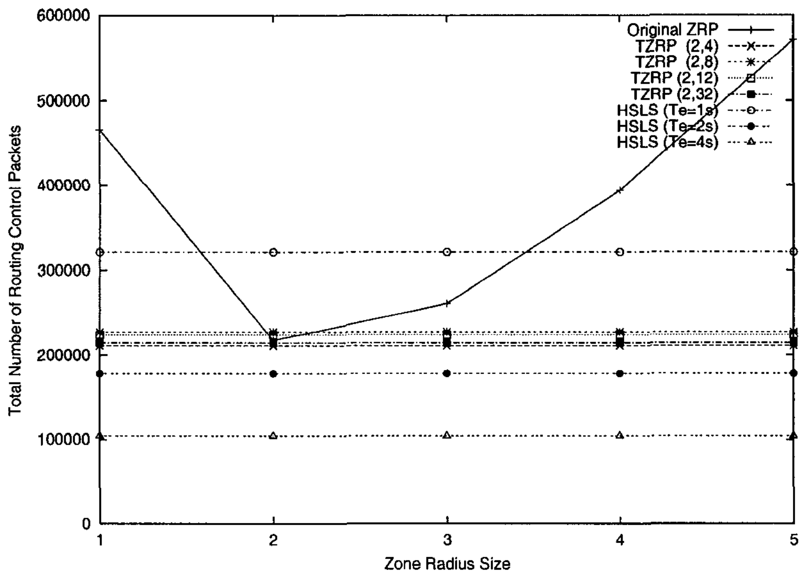


Fig. 31. TZRP performance (traffic T3): total routing control overhead.

TABLE 7

TZRP Performance: Total Routing Control Overhead/Query Success Ratio

| $ZR/(ZR_c, ZR_f)$ | T1 | T2 | T3 |
|---|---|---|---|
| 1 | 766367 / 99.98% | 413857 / 99.93% | 465616 / 99.75% |
| 2 | 265053 / 99.91% | 153695 / 99.77% | 217428 / 98.44% |
| 3 | 242510 / 99.75% | 193023 / 99.87% | 260500 / 98.18% |
| 4 | 343076 / 99.77% | 324332 / 99.78% | 393763 / 98.01% |
| 5 | 507319 / 99.73% | 505624 / 99.70% | 570662 / 97.06% |
| (2,4) | 149370 / 99.21% | 115909 / 99.40% | 211218 / 97.02% |
| (2,6) | 133254 / 99.39% | 126237 / 99.27% | 226383 / 98.03% |
| (2,8) | 147584 / 99.29% | 144358 / 99.20% | 227307 / 97.51% |
| (2,10) | 162371 / 99.24% | 158457 / 99.50% | 228269 / 97.17% |
| (2,12) | 168137 / 99.43% | 166509 / 99.27% | 224054 / 96.76% |
| (2,32) | 182646 / 99.36% | 180342 / 99.20% | 214390 / 96.00% |
| (3,6) | 206158 / 99.57% | 201740 / 99.60% | 273942 / 97.55% |
| (3,8) | 224435 / 99.48% | 222862 / 99.60% | 282557 / 97.53% |
| (3,10) | 238224 / 99.48% | 236685 / 99.54% | 288463 / 96.89% |
| (3,12) | 246195 / 99.55% | 244889 / 99.30% | 289724 / 97.00% |
| HSLS ($t_e = 1s$) | 320357 / 99.15% | 320897 / 98.97% | 321254 / 88.92% |
| HSLS ($t_e = 2s$) | 177877 / 98.29% | 178188 / 97.78% | 177996 / 82.83% |
| HSLS ($t_e = 4s$) | 104028 / 96.16% | 103865 / 95.82% | 104359 / 74.46% |

the existence of complicated interactions between ZRP/TZRP and the underlying MAC layer when an IEEE 802.11-like MAC is used. For example, if a LSU or RREQ transmission is translated into multiple reliable MAC-layer unicasts, then the number of control overhead introduced by bordercasting can be overwhelming; if a LSU or RREQ transmission is translated into a single unreliable MAC-layer broadcast, then ZRP/TZRP's behavior can become more unpredictable since either of these messages can be lost.

On the other hand, many solutions have been proposed to alleviate such a *broadcast storm* problem [80]. Studying and comparing the performance of those solutions is not the focus of this work. Hence we choose to keep most simulation parameters as the same values used in the ideal MAC case, only increasing the HELLO interval to $0.5s$, and the *jitter* value for LSU and RREQ transmissions to $0.02s$. The simulation results under IEEE 802.11 MAC are shown in Table 8. Comparing Table 8 and Table 7, although LSU and RREQ collisions make the simulation results somewhat different from those corresponding to an ideal MAC, the basic trend reflecting the flexibility of TZRP in balancing control overhead and maintaining high query success ratio over ZRP and HSLS is still obvious.

## IV.5 THEORETICAL ANALYSIS OF THE CONTROL OVERHEAD OF HYBRID ROUTING PROTOCOLS

Most existing work on ZRP is based on simulations trying to verify the intuitions that motivate ZRP. In this section, we present a theoretical analysis of ZRP and TZRP. We derive expressions for total control overhead induced by these routing frameworks. These expressions provide a deeper insight into the performance of hybrid routing protocols in general. In Subsection IV.5.1, we list the assumptions and notations we use in our analysis. In Subsection IV.5.2, we derive the lower bound expressions for the control overhead of *broadcasting* and *bordercasting*. Based on these expressions, we analyze the total control overhead of ZRP and TZRP in Subsection IV.5.3 and IV.5.4.

### IV.5.1 Assumptions and notations

We assume that nodes are uniformly distributed. A generic node $S$ only communicates with the nodes that are no more than $R$ hops away, and $R$ is called *world*

TABLE 8
TZRP Performance: Total Routing Control Overhead/Query Success Ratio, IEEE 802.11 MAC

| $ZR/(ZR_c, ZR_f)$ | T1 | T2 | T3 |
|---|---|---|---|
| 1 | 736873 / 94.60% | 412587 / 99.47% | 545569 / 90.38% |
| 2 | 220998 / 99.25% | 136184 / 99.50% | 201665 / 90.66% |
| 3 | 207926 / 98.76% | 171609 / 98.87% | 230607 / 85.90% |
| 4 | 281075 / 98.52% | 269553 / 98.31% | 316330 / 79.20% |
| 5 | 389504 / 98.60% | 388627 / 98.81% | 420905 / 72.22% |
| (2,4) | 138001 / 98.31% | 107278 / 98.21% | 201062 / 90.48% |
| (2,6) | 120524 / 98.55% | 111742 / 98.41% | 205572 / 91.80% |
| (2,8) | 126909 / 98.37% | 122827 / 98.21% | 201596 / 91.18% |
| (2,10) | 134745 / 98.43% | 130029 / 98.14% | 198550 / 90.34% |
| (2,12) | 137424 / 98.47% | 135791 / 98.37% | 194901 / 91.23% |
| (2,32) | 142745 / 98.52% | 141874 / 98.37% | 184416 / 89.68% |
| (3,6) | 181391 / 98.72% | 176973 / 98.44% | 241674 / 88.94% |
| (3,8) | 191336 / 98.69% | 189187 / 98.61% | 243377 / 88.58% |
| (3,10) | 199035 / 98.55% | 198291 / 98.77% | 245060 / 88.80% |
| (3,12) | 203685 / 98.44% | 201341 / 98.71% | 243879 / 88.69% |
| HSLS ($t_e = 1s$) | 241122 / 97.85% | 242796 / 98.21% | 241069 / 81.92% |
| HSLS ($t_e = 2s$) | 137468 / 96.96% | 138190 / 96.25% | 138820 / 74.84% |
| HSLS ($t_e = 4s$) | 86614 / 92.96% | 86270 / 92.80% | 86260 / 64.14% |

*radius.* We list the notations used in our theoretical analysis and their corresponding meanings in Table 9.

Note that (1) $\lambda_m$ is proportional to number of changes in the neighbor list / second; (2) $\lambda_t$ is proportional to the number of queries/second, including both *intra*-zone and *inter*-zone query.

The function $T(i)$ describes the probability that a flow has a distance $\leq i$ hops. In this Section, we use the following three definitions of $T(i)$:

$$T_1(i) = \frac{i^2}{R^2} \tag{1}$$

$$T_2(i) = \frac{i}{R} \tag{2}$$

$$T_3(i) = 1 - \frac{1}{2^i} \tag{3}$$

TABLE 9
Notations Used in the TZRP Overhead Analysis

| Notation | Meaning |
|---|---|
| $\lambda_m$ | mobility rate |
| $\lambda_t$ | traffic rate |
| $R$ | world radius of a generic node |
| $x$ | Crisp Zone radius of a generic node |
| $y$ | Fuzzy Zone radius of a generic node |
| $T(i)$ | traffic locality function |
| $T_r$ | transmission range of a node |
| $\rho$ | node density (number of nodes/ $m^2$) |
| $D$ | node degree ( $\rho = D/(\Pi * T_r^2)$) |
| $a$ | call to mobility ratio ($\lambda_t/\lambda_m$) |
| $PO$ | proactive routing control overhead (per node, per second) |
| $RO$ | reactive routing control overhead (per node, per second) |
| $TO$ | total routing control overhead (per node, per second) |

Among the above three traffic patterns, $T_1$ is the *uniform* traffic pattern. In terms of traffic locality, $T_3$ is more local than $T_2$, which is in turn more local than $T_1$.

### IV.5.2 Overhead lower bounds for broadcasting and bordercasting

In this subsection, we derive lower bounds for two important tasks in ZRP: *broadcasting* and *bordercasting*. In a *broadcasting* task, the message sent by the source node is sent to *all* the node in the network [77]. In comparison, the goal of a *bordercasting* task is to send the query to *all* levels of the *peripheral* nodes. Although various efficient broadcasting schemes [77] can be used to finish a bordercasting task, bordercasting itself does not require every node in the network should receive the query. Understanding the difference between the goals of these two tasks is crucial for understanding the lower bounds derived below.

**Property IV.5.1** *The lower bound of the overhead of a* broadcasting *task* $= \Theta(R^2)$

Proof. First, the total number of nodes in the world $= \rho * \Pi * (R * T_r)^2) = D * R^2$
To dominate $D * R^2$ nodes, we need at least $\Theta(R^2)$ transmissions.

Next, we construct a broadcasting scheme that achieves the $\Theta(R^2)$ lower bound. Consider the source node $S$ of the broadcasting task. The number of nodes that are in the distance of exactly 1 hop $= D$, the number of nodes that are in the distance of exactly 2 hops $= D*4-D*1 = 3D$, the number of nodes that are in the distance of exactly 3 hops $= D*9 - D*4 = 5D$, and so on. Similarly, the number of nodes that are in the distance of exactly $R$ hops $= D*R^2 - D*(R-1)^2 = (2R-1)D$.

In order for all these nodes to receive a copy of the message, we can have the following nodes to take part in transmitting and forwarding the message: the source node, 3 of the 1-hop nodes, 5 of the 2-hop nodes, 7 of the 3-hop nodes,..., $2R-1$ of the $(R-1)$-hop nodes. So the minimum number of message transmissions/forwardings is $1+3+5+7+...+(2R-1) = \Theta(R^2)$. □

**Property IV.5.2** *Consider node $S$ and its peripheral node set $P$. The lower bound of the overhead of paging $P = \Theta(x)$.*

Proof. First, to dominate $|P| = D*x^2 - D*(x-1)^2 = D*(2*x-1)$ nodes, we need at least $\Theta(x)$ transmissions.

Next, we construct a bordercasting scheme that achieves the $\Theta(R^2)$ lower bound. let's construct a $D$-ary tree with the nodes in $|P|$ as leaves. Assume its depth is $h$. Then $D^h = D*(2*x-1)$. So we have $h \approx lg(2*x)$. Hence the number of nodes in the tree $= 1+D+D^2+D^3+D^{h-1} = \frac{1-D^h}{1-D} = \frac{D*(2*x-1)-1}{D-1} \approx 2*x$. The total overhead $= x-h+2*x \approx 3*x = \Theta(x)$ .

□

**Property IV.5.3** *The lower bound of the overhead of a* bordercasting *task* $= \Theta(\frac{(R-x)^2}{x})$

Proof. In an *ideal* bordercasting scheme, a node that is $x$-hop away from the boundary of the world terminates the query and stops forwarding it any further away from the source node. Since each bordercasting node covers $\Theta(x^2)$ nodes, we need at least $Theta(\frac{(R-x)^2}{x^2})$ bordercasting nodes to cover the whole network. According to Property IV.5.2, the minimum overhead of each bordercasting tree is $\Theta(x)$. So the minimum overhead of a bordercasting task $= \Theta(\frac{(R-x)^2}{x^2} * x) = \Theta(\frac{(R-x)^2}{x})$.

□

Note that we did not consider the back propagation of queries when deriving the above lower bound for bordercasting. Although the two versions of bordercasting schemes proposed for ZRP [32] significantly reduce the back propagation, it is not completely eliminated. Hence the lower bound here demonstrates the best behavior of bordercasting in terms of control overhead, and by using this lower bound in our theoretical analysis, we derive the lowest possible control overhead that can be achieved by ZRP.

### IV.5.3 Overhead analysis of ZRP

In this subsection, we analyze the total control overhead of ZRP. We use the lower bounds derived above to express the proactive and reactive overhead. For simplicity, we omit the $\Theta$ notations in the following properties, and we also assume that $|LSU| = |RREQ| = 1$ in the following discussions. Note that this simplification may influence the absolute value of the total routing control overhead, but does not influence the trend showing the effect of radius changes on the total routing control overhead, which is the focus of our discussion here.

**Property IV.5.4** *In ZRP,* $PO = \lambda_m * |LSU| * x^2$.

Proof. This property follows immediately from Property IV.5.1. □

**Property IV.5.5** *In ZRP,* $RO = \lambda_t * |RREQ| * (1 - T(x)) * \frac{(R-x)^2}{x}$.

Proof. This property follows immediately from Property IV.5.3. □

**Property IV.5.6** *In ZRP,* $TO = \lambda_m * |LSU| * x^2 + \lambda_t * |RREQ| * (1 - T(x)) * \frac{(R-x)^2}{x}$.

Proof. This property follows immediately from Property IV.5.4 and IV.5.5. □

When the traffic locality function is $T_1$, we have: $TO = \lambda_m * x^2 + \lambda_t * (1 - \frac{x^2}{R^2}) * \frac{(R-x)^2}{x}$. We draw the curve for $TO$ under different $a = \frac{\lambda_t}{\lambda_m}$ values, and the results are shown in Figure 32, 33, and 34. From Figure 32 and 33, we can observe the basic trend of zone radius' influence on the total control overhead. This curve is very similar to the simulation result of ZRP shown in the literature [31, 58] as well as in Section IV.4, justifying the correctness of our theoretical model of ZRP. From Figure 34, we see that: when $a = 1$, the optimal zone radius = 15; when $a = 5$, the optimal zone radius = 26; when $a = 10$, the optimal zone radius = 35. This confirms the intuition and simulation results which suggest that in ZRP *high call to mobility ratio favors larger zone radius, while low call to mobility ratio favors smaller zone radius* [31, 58, 67].

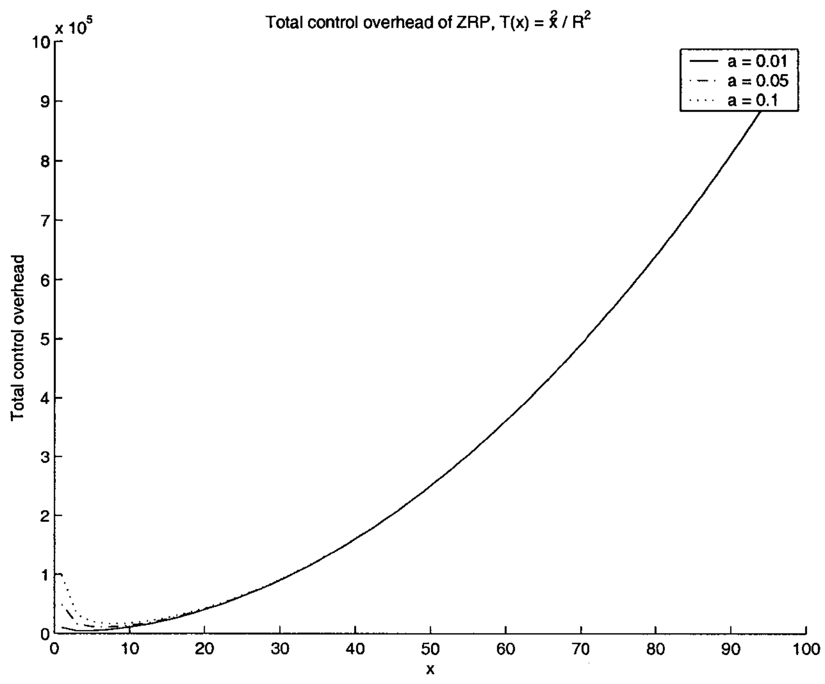


Fig. 32. ZRP control overhead (traffic T1): low call-to-mobility ratio.

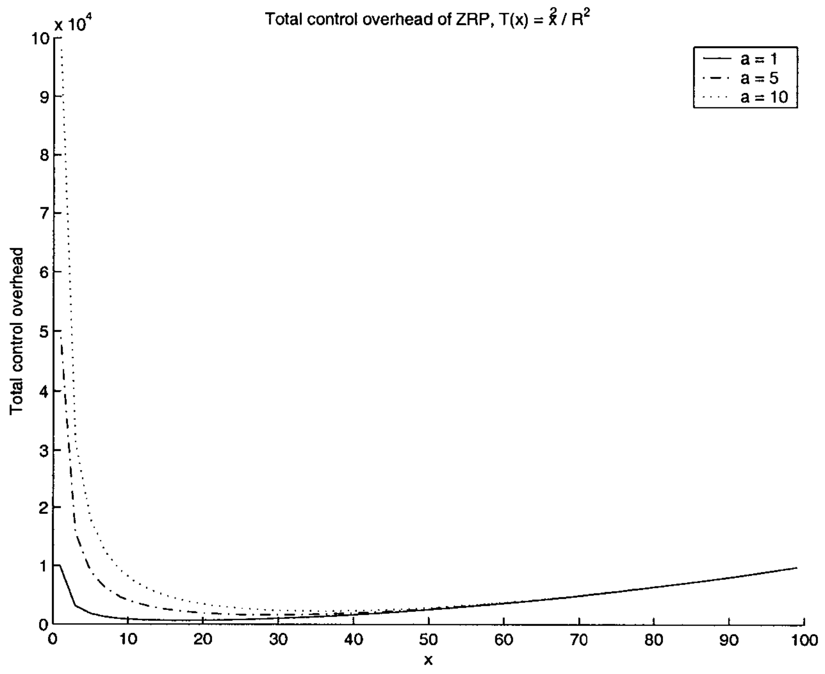


Fig. 33. ZRP control overhead (traffic T1): high call-to-mobility ratio.

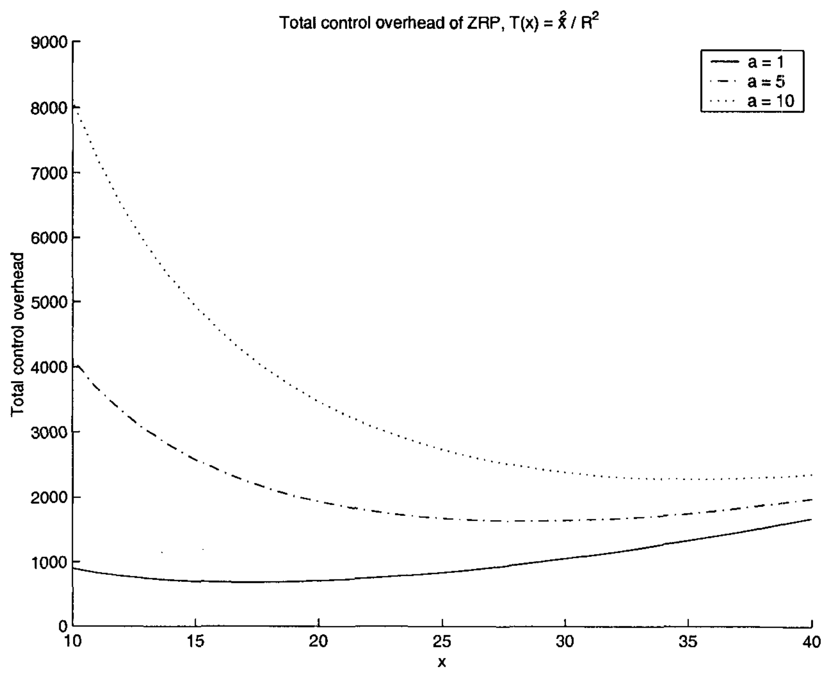


Fig. 34. ZRP control overhead (traffic T1): high call-to-mobility-ratio, zoomed in.

When the traffic locality function is $T_2$, we have: $TO = \lambda_m * x^2 + \lambda_t * (1 - \frac{x}{R}) * \frac{(R-x)^2}{x}$. We draw the curve for $TO$ under different $a$ values, and the results are shown in Figure 35, 36, and 37. Comparing with the total overhead under $T1$, little changes in the overall behavior are observed.

When the traffic locality function is $T_3$, we have: $TO = \lambda_m * x^2 + \lambda_t * (\frac{1}{2^x}) * \frac{(R-x)^2}{x}$. We draw the curve for $TO$ under different $a$ values, and the results are shown in Figure 38, 39, and 40. Under this traffic pattern, strong traffic locality is demonstrated. This drives the optimal zone radius to a much smaller value when compared with the above two traffic patterns (6, 8, 9 in Figure 40 *vs.* 15, 26, 35 in Figure 34).

It is worth pointing out that ZRP's ability in balancing the proactive overhead and reactive overhead by changing zone radius heavily depends on *how the implementation of each component actually approximates the overhead lower bound.* For example, when efficient broadcasting [77] is used to do bordercasting, then $RO = R^2$. Hence we have:

$$TO = \lambda_m * x^2 + \lambda_t * T(x) * R^2 \quad (4)$$

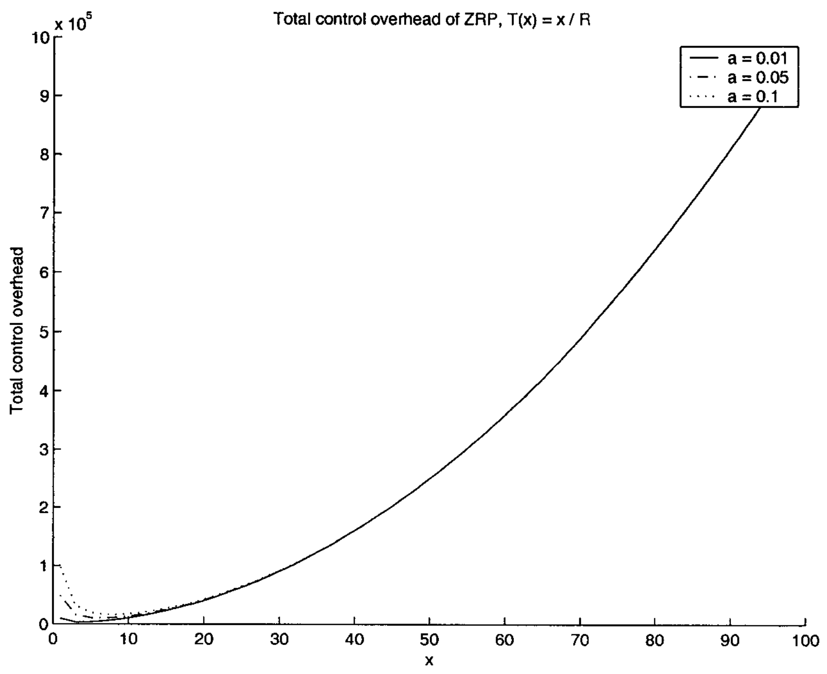


Fig. 35. ZRP control overhead (traffic T2): low call-to-mobility ratio.

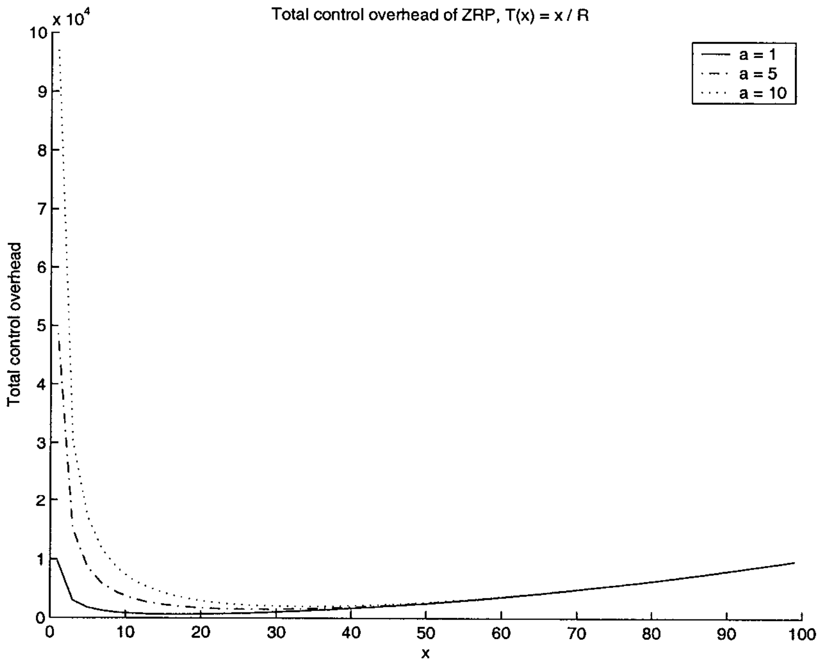


Fig. 36. ZRP control overhead (traffic T2): high call-to-mobility ratio.

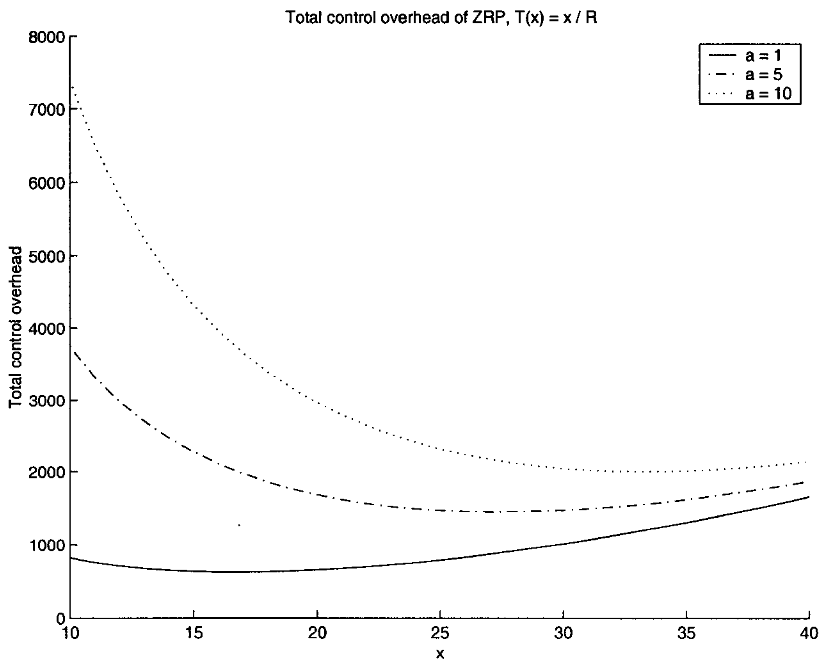


Fig. 37. ZRP control overhead (traffic T2): high call-to-mobility ratio, zoomed in.

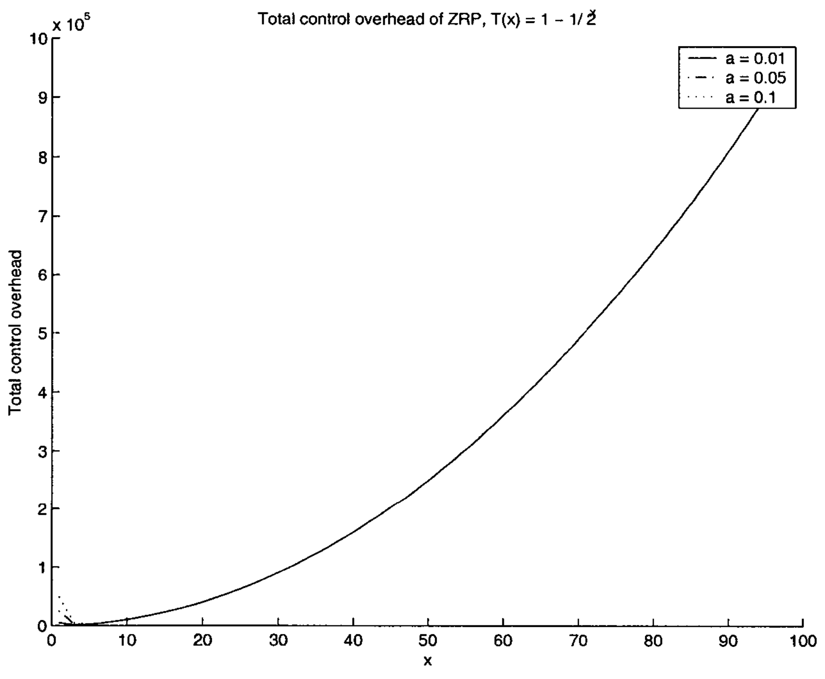


Fig. 38. ZRP control overhead (traffic T3): low call-to-mobility ratio.

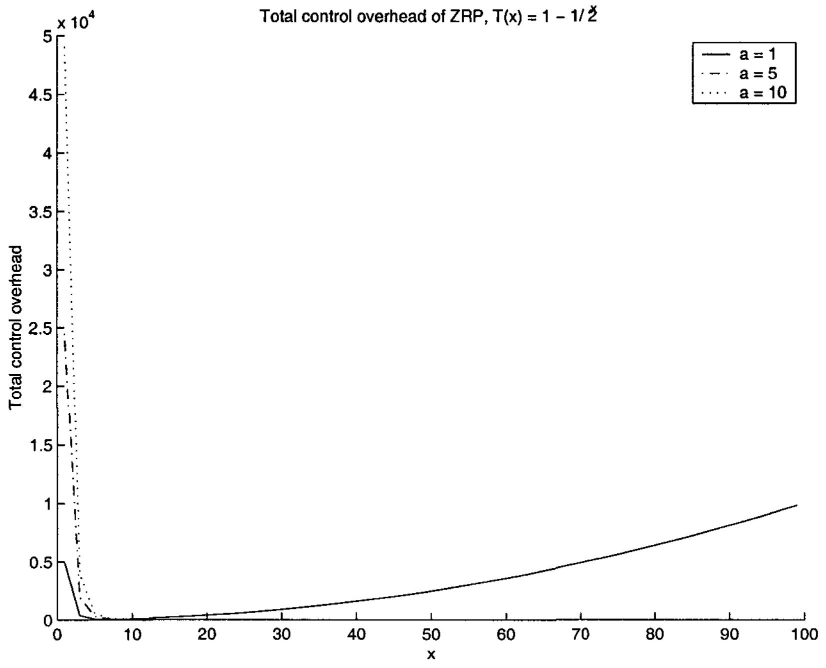


Fig. 39. ZRP control overhead (traffic T3): high call-to-mobility ratio.

Under uniform traffic pattern,

$$TO = \lambda_m * x^2 + \lambda_t * (1 - \frac{x^2}{R^2}) * R^2 = (\lambda_m - \lambda_t) * x^2 + \lambda_t * R^2 \tag{5}$$

The derivative is:

$$\frac{dTO}{dx} = 2 * x * (\lambda_m - \lambda_t) \tag{6}$$

According to the derivative,

1. when $\lambda_m > \lambda_t$, $TO$ is an increasing function, hence it achieves its minimum value at $x = 0$, which is pure reactive;

2. when $\lambda_m < \lambda_t$, $TO$ is a decreasing function, hence it achieves its minimum value at $x = R$, which is pure proactive;

3. when $\lambda_m = \lambda_t$, the value of $x$ is irrelevant, and $TO = \lambda_t * R^2$.

Hence in this case of bordercasting approximation, the smallest control overhead that can be achieved by ZRP is always *equal* to the control overhead of either the pure proactive protocol or the pure reactive protocol. The only role that ZRP can play is

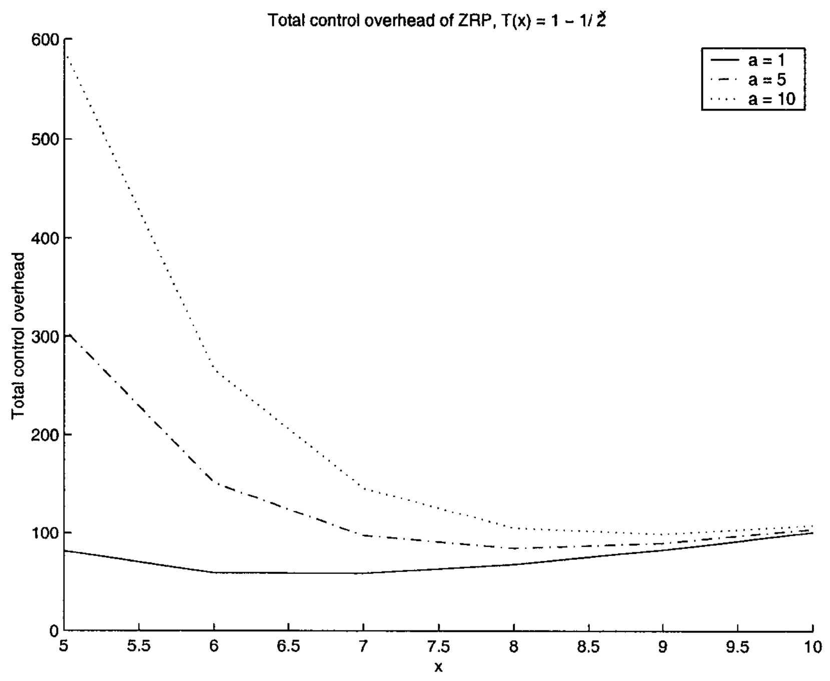


Fig. 40. ZRP control overhead (traffic T3): high call-to-mobility ratio, zoomed in.

to switch between these two basic protocols based on a given network condition, and none of the intermediate zone radius settings can achieve lower total control overhead than basic components.

Comparing the different conclusions we draw about ZRP when using the theoretical bound and the approximation overhead of bordercasting, we can see that ZRP provides a powerful hybrid routing framework that is more than simply switching between basic protocols. Indeed, the synergy provided by exploiting zone topology in *bordercasting* has the potential to make the framework achieve *smaller* control overhead than either of the basic component protocols alone, and this is where the power of ZRP lies.

### IV.5.4 Overhead analysis of TZRP

In this subsection, we analyze the total control overhead of TZRP.

**Property IV.5.7** *In TZRP,* $PO = \lambda_m * |LSU| * (x^2 + y)$*;*

Proof. It is obvious that the overhead for maintaining the Crisp Zone is $\lambda_m * |LSU| * x^2$.

To compute the overhead for maintaining the Fuzzy Zone, we assume HSLS is used as the Fuzzy Proactive component of TZRP. Assume that $y = 2^{\beta}$. Every $\lambda_m$ link changes is broadcast with $TTL = 1$, which is received by $D$ nodes; every $2 * \lambda_m$ link changes is broadcast with $TTL = 2$, which is received by $(2^2) * D$, every $4 * \lambda_m$ link changes is broadcast with $TTL = 4$, which is received by $(4^2) * D$, every $8 * \lambda_m$ link changes is broadcast with $TTL = 8$, which is received by $(8^2) * D$, ..., every $y * \lambda_m$ link changes is broadcast with $TTL = y$, which is received by $(y^2) * D$.

So the Fuzzy Proactive overhead $= \Theta(1 + \frac{2^2}{2} + \frac{4^2}{4} + \frac{8^2}{8} + \ldots + \frac{\beta^2}{\beta}) = \Theta(\frac{1-2^{\beta}}{1-2}) = \Theta(y)$.

□

**Property IV.5.8** *In TZRP, RO* $= \lambda_t * |RREQ| * (FuzzyZoneFailureRate * (T(y) - T(x)) + (1 - T(y))) * \frac{(R-x)^2}{x}$;

Proof. In TZRP, there are two cases that can trigger the execution of bordercasting: (1) the destination is located out of the Fuzzy Zone; (2) Fuzzy Zone routing failure.

Based on Property IV.5.5, the overhead caused by case (1) $= \lambda_t * |RREQ| * (1 - T(y) * \frac{(R-x)^2}{x}$.

The overhead caused by case (2) $= \lambda_t * |RREQ| * FuzzyZoneFailureRate * (T(y) - T(x)) * \frac{(R-x)^2}{x}$. In the expression, $\lambda_t * FuzzyZoneFailureRate * (T(y) - T(x))$ is the rate of traffic that is destined to those nodes that are located out of the Crisp Zone but inside the Fuzzy Zone, and $FuzzyZoneFailureRate$ of them fail due to fuzzy information.

$FuzzyZoneFailureRate$ can be modeled using a very small constant as in [70]. Or it can be modeled as $(\frac{\lambda_m}{D})^D$, which more accurately reflects the influence of mobility and density. □

**Property IV.5.9** *In TZRP, TO* $= \lambda_m * |LSU| * (x^2 + y) + \lambda_t * |RREQ| * (FuzzyZoneFailureRate * (T(y) - T(x)) + (1 - T(y))) * \frac{(R-x)^2}{x}$;

Proof. This property follows immediately from Property IV.5.7 and IV.5.8. □

The difference between TZRP and ZRP can be illustrated using several representative scenarios. In Figure 41 and 42, we show the total control overhead using traffic function $T_1$,and $R = 100, D = 6$ viewed from two different angles. The settings of $a = \frac{\lambda_t}{\lambda_m} = \frac{1}{1}, \frac{1}{4}, \frac{4}{1}, \frac{4}{4}$, respectively. In all these scenarios, we can see that TZRP achieves lowest total control overhead that cannot be achieved by ZRP.

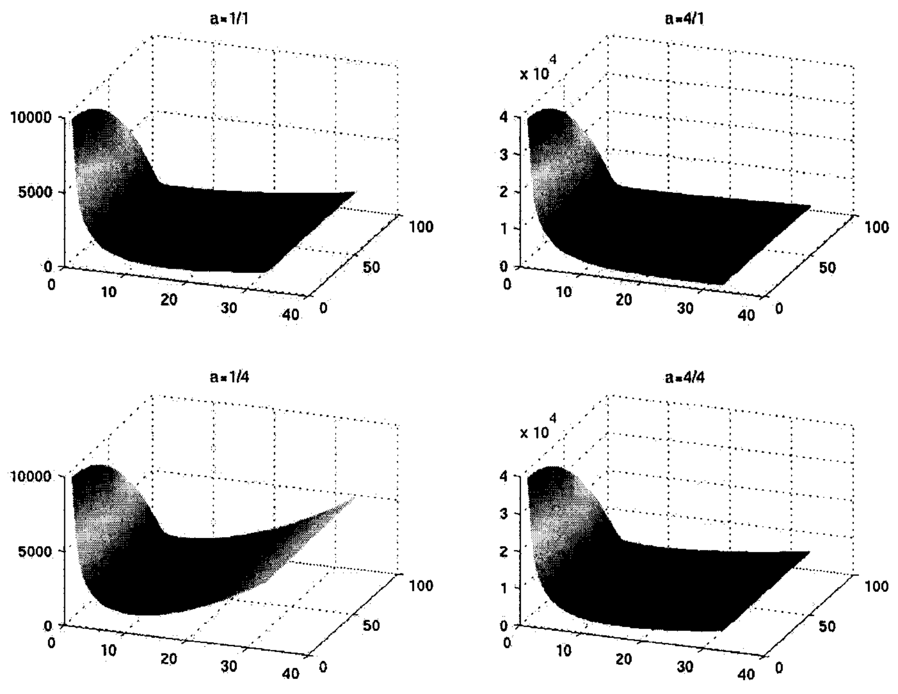


Fig. 41. TZRP control overhead (traffic T1): x = [1, 40], y = [1, 100], angle 1.

The benefit of TZRP becomes more obvious under local traffic pattern $T_2$ and $T_3$ as demonstrated in Figure 43, 44, 45, and 46.

A more comprehensive comparison between TZRP and ZRP requires a more detailed analysis of the constants used in the expressions we derived above, and we leave it as our future work.

## IV.6 SUMMARY

To set the stage for discussing our novel hybrid routing framework, in Section II.2 we have reviewed a number of hybrid MANET routing protocols proposed in the literature. By integrating suitable proactive and reactive components to adapt to changing network conditions, a hybrid protocol can provide better performance in a wide range of MANET environments. One such protocol, the prominent Zone Routing Protocol (ZRP), provides a hybrid routing framework that is locally proactive and globally reactive, with the goal of minimizing the sum of the proactive and reactive control overhead.

We propose Two-Zone Routing Protocol (TZRP) as a general hybrid routing

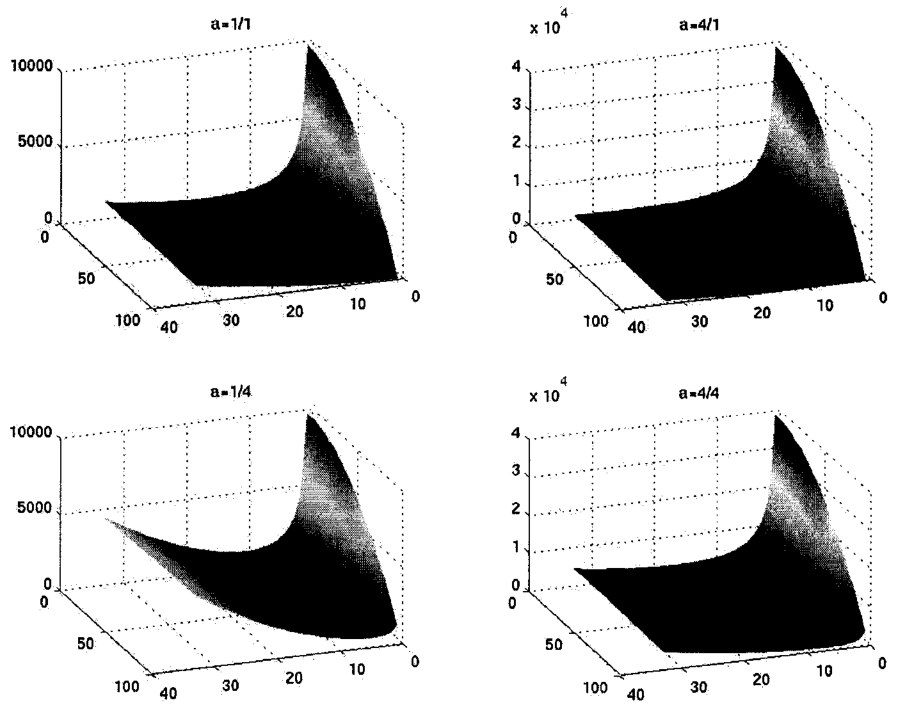


Fig. 42. TZRP control overhead (traffic T1): x = [1, 40], y = [1, 100], angle 2.

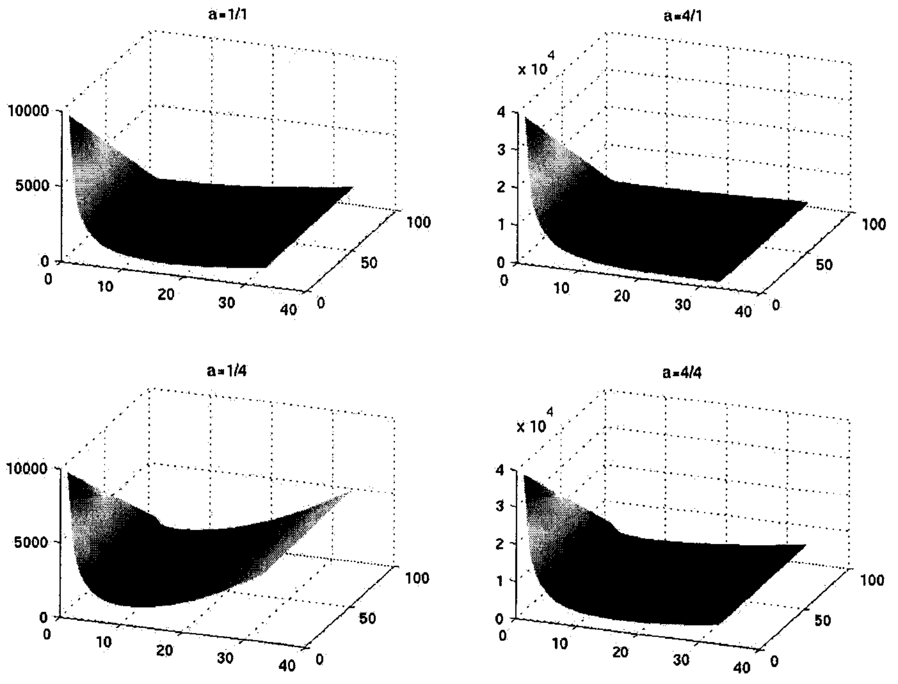


Fig. 43. TZRP control overhead (traffic T2): x = [1, 40], y = [1, 100], angle 1.

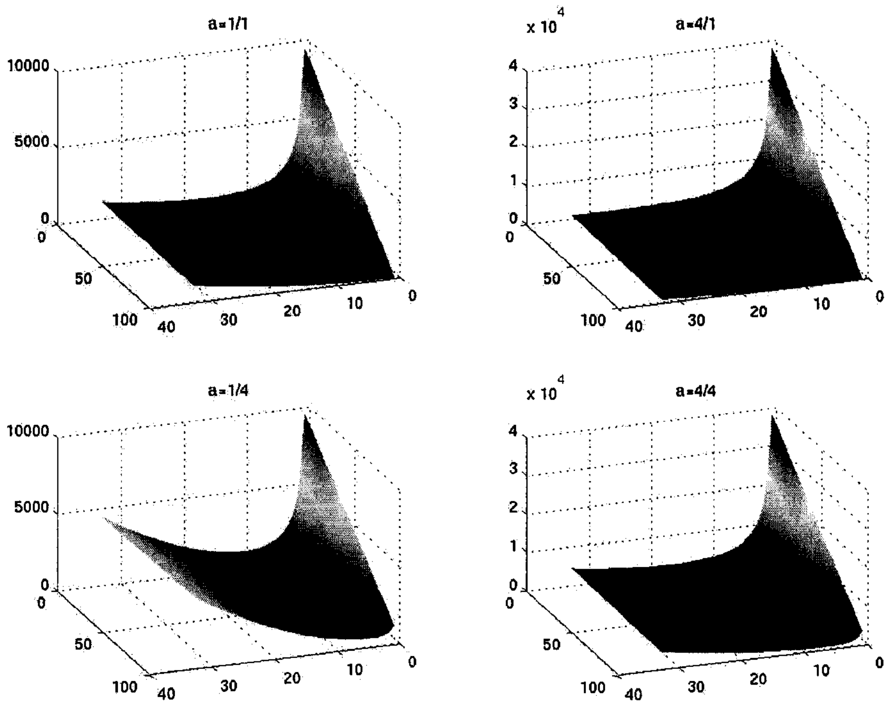


Fig. 44. TZRP control overhead (traffic T2): x = [1, 40], y = [1, 100], angle 2.

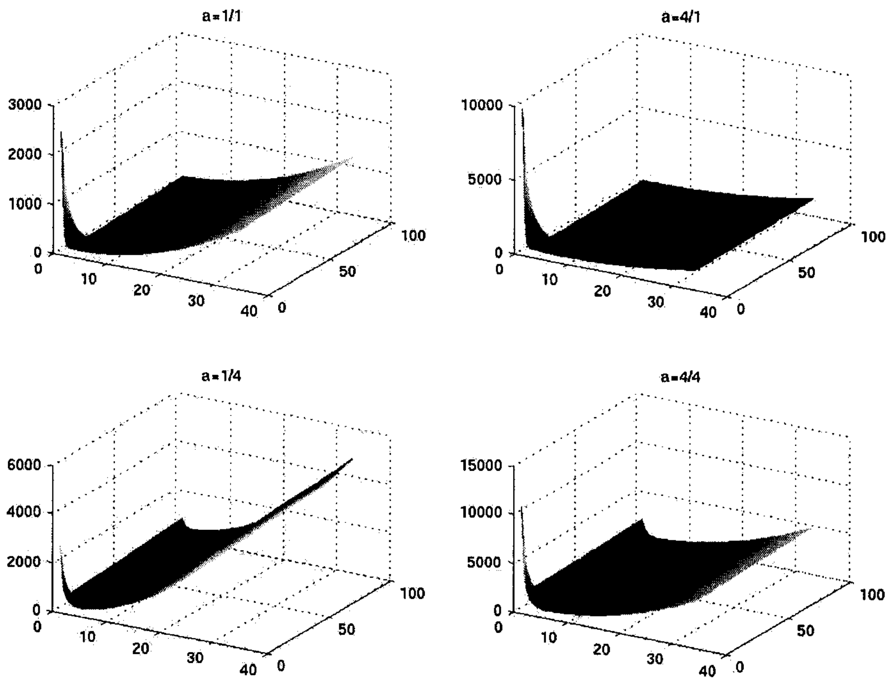


Fig. 45. TZRP control overhead (traffic T3): x = [1, 40], y = [1, 100], angle 1.

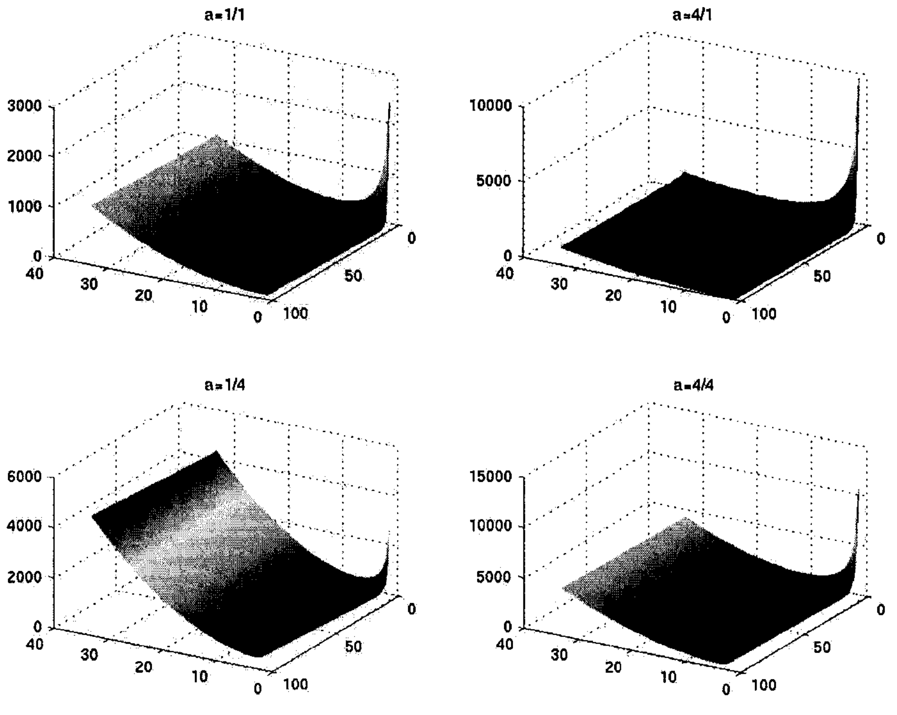


Fig. 46. TZRP control overhead (traffic T3): x = [1, 40], y = [1, 100], angle 2.

framework that can balance the tradeoffs between pure proactive, fuzzy proactive, and reactive routing approaches more effectively than ZRP in a wide range of network conditions. Our key observation is that in the original ZRP the zone serves a dual purpose. TZRP uses two different zones – different in both topology information and update mechanisms – in order to decouple the framework's ability to adapt to traffic characteristics from the ability to adapt to mobility. By adjusting these two radii, a lower total routing control overhead can be achieved. We demonstrate the effectiveness of TZRP through extensive simulations and theoretical analysis.

# CHAPTER V

# SUMMARY AND FUTURE WORK

## V.1 SUMMARY

This dissertation has been focused on studying virtual infrastructures in MANET.

In the first part of the dissertation, we study *cluster* - an *explicit* virtual infrastructure proposed for MANET. Motivated by the idea that providing general-purpose infrastructures makes a large MANET appear *smaller* and *less dynamic*, we propose a novel clustering scheme based on a number of properties of *diameter-2* graphs. Compared to virtual infrastructures with central nodes, our virtual infrastructure is more *symmetric* and *stable*, but still *light-weight*. In our clustering scheme, cluster initialization naturally blends into cluster maintenance, showing the unity between these two operations. Unlike the cluster maintenance algorithm in [47], our algorithm does not require maintaining complete cluster topology information at each node. We call our algorithm *tree-based* since cluster merge and split operations are performed based on a spanning tree maintained at some specific nodes.

In the second part of the dissertation, we study *zone*, an *implicit* virtual infrastructure, and its applications in hybrid routing. We develop a theoretical model for the routing control overhead of zone-based hybrid routing protocols, which provides a deeper insight into the power of hybrid routing. We propose a novel hybrid routing framework TZRP. By integrating pure proactive, fuzzy proactive, and reactive routing approaches under the same framework, TZRP can adapt to a wide range of network conditions. The effectiveness of TZRP has been demonstrated through both detailed *ns-2* simulations and theoretical analysis.

## V.2 FUTURE WORK

There are still many interesting and important research problems to be solved in the above work.

In the *clustering* part, the tree-based clustering algorithm proposed in this dissertation can be further generalized to achieve (*d1*, *d2*)-clustering in which: two clusters merge when the diameter of the resulting cluster is not larger than *d1*, and a cluster is split into several diameter-*d1* clusters if its diameter is larger than *d2*. By adaptively changing the values of *d1* and *d2*, a stable and symmetric general-purpose virtual infrastructure can be achieved efficiently in large-scale MANET.

In the *hybrid routing* part, we are working on a more detailed analytical model of TZRP. Also, efficient adaptive mechanisms to adjust the Crisp/Fuzzy Zone radius dynamically need further investigation. In addition, simulations focusing on larger networks and cross-layer interactions may provide more insight into the performance of TZRP. There is some asymmetry in the use of the Crisp and Fuzzy Zones in servicing route discovery. It would be highly desirable to use the fuzzy information inherent in the Fuzzy Zone to achieve a robust form of fuzzy bordercasting, leading to a hybrid routing framework that is more general than TZRP. This promises to be an exciting area for further work.

Finally, our two main contributions in this work can be combined together under the broad context of QoS provisioning in MANET. Integrating explicit clustering schemes at the node level and TZRP scheme at the cluster level has the potential to provide an adaptive and robust QoS provisioning framework for large-scale MANET.

# REFERENCES

[1] K. M. Alzoubi, P.-J. Wan, and O. Frieder, "New Distributed Algorithm for Connected Dominating Set in Wireless Ad Hoc Networks," *Proc. IEEE HICSS35 Conf.*, Jan. 2002.

[2] K.M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-Optimal Connected-Dominating-Set Construction for Routing in Mobile Ad Hoc Networks," *Proc. MOBIHOC Conf. 2002*, June 2002.

[3] K.M. Alzoubi, P.-J. Wan, and O. Frieder, "Weakly Connected Dominating Sets and Sparse Spanners for Wireless Ad Hoc Networks," *Proc. 23rd International Conference on Distributed Computing Systems*, 2003.

[4] A.D. Amis, R. Prakash, D. Huynh, and T. Vuong, "Max-Min d-Cluster Formation in Wireless Ad Hoc Networks," *Proc. INFOCOM Conf. 2000*, vol. 1, pp. 32-41, 2000.

[5] A.D. Amis and R. Prakash. "Load-Balancing Clusters in Wireless Ad Hoc Networks," *Proc. ASSET Conf. 2000*, Richardson, Texas, March 2000.

[6] B. An and S. Papavassiliou, "A Mobility-Based Clustering Approach to Support Mobility Management and Multicast Routing in Mobile Ad-hoc Wireless Networks," *International Journal of Network Management*, vol. 11, no. 6, pp. 387-395, 2001.

[7] D.J. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm," *IEEE Transactions on Communications*, vol. 29, no. 11, pp. 1694-1701, 1981.

[8] S. Banerjee and S. Khuller, "A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks," *Proc. INFOCOM Conf. 2001*, Apr. 2001.

[9] L. Bao and J.J. Garcia-Luna-Aceves, "Topology Management in Ad Hoc Networks," *Proc. MOBIHOC Conf. 2003*, June 2003.

[10] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," *Proc. MOBICOM Conf. 1998*, pp. 76-84, Oct. 1998.

[11] S. Basagni, "Distributed Clustering for Ad Hoc Networks," *Proc. the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks*, pp. 310-315, June 1999.

[12] S. Basagni, "Distributed and Mobility-Adaptive Clustering for Multimedia Support in Multi-hop Wireless Networks," *Proc. the IEEE 50th International Vehicular Technology Conference*, vol. 2, pp. 889-893, Sept. 1999.

[13] Y. Bejerano, "Efficient Integration of Multi-hop Wireless and Wired Networks with QoS Constraints," *Proc. MOBICOM Conf. 2002*, pp. 215-226, Sept. 2002.

[14] C. Bettstetter and R. Krausser, "Scenario-Based Stability Analysis of the Distributed Mobility-Adaptive Clustering (DMAC) Algorithm," *Proc. MOBIHOC Conf. 2001*, pp. 232-241, Oct. 2001.

[15] L. Blazevic, L. Buttyan, S.G.S. Capkun, J.P. Hubaux, and J.Y.L. Boudec, "Self-Organization in Mobile Ad-Hoc Networks: the Approach of Terminodes," *IEEE Computer Communications Magazine*, vol. 39, no. 6, June 2001.

[16] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols," *Proc. MOBICOM Conf. 1998*, pp. 85-97, Oct. 1998.

[17] E.M. Belding-Royer, "Hierarchical Routing in Ad Hoc Mobile Networks," *Wireless Communication & Mobile Computing*, vol. 2, no. 5, pp. 515-532, 2002.

[18] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communication & Mobile Computing: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483-502, 2002.

[19] M. Chatterjee, S. Das, and D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks," *Cluster Computing*, vol. 5, no. 2, pp. 193-204, 2002.

[20] C.C. Chiang, H.K. Wu, W. Liu, and M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel," *Proc. IEEE Singapore International Conference on Networks*, pp. 197-211, Apr. 1997.

[21] Y. Chen and A. Liestman, "A Zonal Algorithm for Clustering Ad Hoc Networks," *International Journal of Foundation of Computer Science*, vol. 14, no. 2, pp. 305-322, 2003.

[22] H. Chan and A. Perrig, "ACE: An Emergent Algorithm for Highly Uniform Cluster Formation," *Proc. 2004 European Workshop on Sensor Networks*, pp. 154-171.

[23] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 1992.

[24] B. Das, R. Sivakumar, and V. Bhargavan, "Routing in Ad Hoc Networks Using a Spine," *Proc. the International Conference on Computer Communications and Networks (IC3N)*, pp. 376-380, 1997.

[25] S.R. Das, C.E. Perkins, E.M. Royer, and M.K. Marina, "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks," *IEEE Personal Communications Magazine, Special Issue on Ad hoc Networking*, vol. 8, no. 1, pp. 16-28, 2001.

[26] A. Ephremides, J.E. Wieselthier, and D. Baker, "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling," *Proc. of IEEE*, vol. 75, no. 1, pp. 56, 1987.

[27] Y. Fernandess and D. Malkhi, "K-Clustering in Wireless Ad-Hoc Networks," *Proc. the ACM Workshop on Principles of Mobile Computing*, pp. 31-37, 2002.

[28] J.J. Garcia-Luna-Aceves and M. Spohn, "Source-Tree Routing in Wireless Networks," *Proc. ICNP Conf. 1999*, pp. 273, Oct. 1999.

[29] M. Gerla and J. Tsai, "Multicluster, Mobile, Multimedia Radio Network," *Wireless Networks*, vol. 1, no. 3, pp. 255-265, 1995.

[30] M. Gerla, "Fisheye State Routing Protocol (FSR) for Ad Hoc Networks," *IETF MANET Internet Draft*, draft-ietf-manet-fsr-03.txt, work in progress, June 2002.

[31] Z.J. Haas and M.R. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol," *ACM/IEEE Transactions on Networking*, vol. 9, no. 4, pp. 427-438, 2001.

[32] Z.J. Haas, M.R. Pearlman, and P. Samar, "The Bordercast Resolution Protocol (BRP)," *IETF MANET Internet Draft*, draft-ietf-manet-zone-brp-02.txt, work in progress, July 2002.

[33] H. Hadwiger, H. Debrunner, and V. Klee, *Combinatorial Geometry in the Plane*, Holt, Rinehart and Winston, Inc., 1964

[34] A. Helmy, S. Garg, P. Pamu, and N. Nahata, "Contact Based Architecture For Resource Discovery (CARD) in Large Scale MANETs," *Proc. International IEEE/ACM Parallel and Distributed Processing Symposium*, pp. 219-227, Apr. 2003.

[35] T.C. Hou and T.J. Tsai, "An Access-Based Clustering Protocol for Multihop Wireless Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, 2001.

[36] H. Huang, A.W. Richa, and M. Segal, "Approximation Algorithms for the Mobile Piercing Set Problem with Applications to Clustering in Ad-hoc Networks," *Proc. the International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 62-61, 2002.

[37] A. Iwata, C. Chiang, G. Pei, M. Gerla, and T. Chen, "Scalable Routing Strategies for Ad Hoc Wireless Networks," *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, vol. 17, no. 8, pp. 1369-1379, 1999.

[38] P. Jacquet, P. Muhlethaler, and A. Qayyam. "Optimized Link-State Routing Protocol," *IETF MANET Internet Draft*, draft-ietf-manet-olsr-11.txt, work in progress, July 2003.

[39] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, pp. 153-181, 1996.

[40] L. Kleinrock and K. Stevens, "Fisheye: a Lenslike Computer Display Transformation," *Technical Report*, UCLA, Computer Science Department, 1971.

[41] T.J. Kwon and M. Gerla, "Efficient Flooding with Passive Clustering (PC) in Ad Hoc Networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, pp. 44-56, 2002.

[42] P. Krishna, M. Chatterjee, N. Vaidya, and D. Pradhan, "A Cluster-Based Approach for Routing in Ad-Hoc Networks," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 49-64, 1997.

[43] N. Li, J. Hou, and L. Sha, "Design and Analysis of an MST-Based Topology Control Algorithm," *Proc. INFOCOM Conf. 2003*, Apr. 2003.

[44] X.-Y. Li, "Topology Control in Wireless Ad Hoc Networks," *Ad Hoc Networking*, IEEE Press, edited by S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, 2003.

[45] X.-Y. Li, Y. Wang, and W.-Z. Song, "Applications of k-Local MST for Topology Control and Broadcasting in Wireless Ad Hoc Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 12, pp. 1057-1069, 2004.

[46] "Largest Known (Degree, Diameter)-Graphs," http://www-mat.upc.es/grup_de_grafs/grafs/desc_g/desc_g2.html

[47] C.R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265-1275, 1997.

[48] N. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, 1996.

[49] A.B. McDonald and T. Znati, "A Mobility Based Framework for Adaptive Clustering in Wireless Ad-hoc Networks," *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, vol. 17, no. 8, pp. 1466-1487, 1999.

[50] A.B. McDonald and T. Znati, "Design and Performance of a Distributed Dynamic Clustering Algorithm for Ad Hoc Networks," *Proc. the 34th Annual IEEE/ACM Simulation Symposium*, pp. 27-35, Apr. 2001.

[51] A.B. McDonald and T. Znati, "A Dual-Hybrid Adaptive Routing Strategy for Wireless Ad-Hoc Networks," *Proc. WCNC Conf. 2000*, pp. 1125-1130, Sept. 2000.

[52] R. Nagpal and D. Coore, "An Algorithm for Group Formation in An Amorphous Computer," *Proc. the 10th International Conference on Parallel and Distributed Computing Systems*, Oct. 1998.

[53] K. Nakano and S. Olariu, "Randomized Leader Election Protocols in Ad-hoc Networks," *Proc. the 7th International Symposium on Structural Information and Communication Complexity*, pp. 253-268, June 2000.

[54] F.G. Nocetti, J.S. Gonzalez, and I. Stojmenovic, "Connectivity Based k-hop Clustering in Wireless Networks," *Telecommunication System*, vol. 22, no. 1-4, pp. 205-220, 2003.

[55] The Network Simulator - NS-2. http://www.isi.edu/nsnam/ns.

[56] V.D. Park and M.S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *Proc. INFOCOM Conf. 1997*, pp. 1405-1415, Apr. 1997.

[57] D. Patterson and J. Hennessy, *Computer architecture: a quantitative approach*, Morgan Kaufmann Publishers, Inc., 1996.

[58] M.R. Pearlman and Z.J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol," *IEEE Journal on Selected Areas in Communications, special issue on Ad-Hoc Networks*, vol. 17, no. 8, pp. 1395-1414, 1999.

[59] G. Pei, M. Gerla and X. Hong, "LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility," *Proc. MOBIHOC 2000 Conf.*, pp. 11-18, Nov. 2000.

[60] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 212-225, Sept. 1994.

[61] C.E. Perkins and E.M. Royer, "Ad-hoc On-demand Distance Vector Routing," *Proc. 2-nd IEEE Workshop on Mobile Computer Systems and Applications*, pp. 90-99, Feb. 1999.

[62] S. Ramanathan and M. Steenstrup, "Hierarchically-Organized, Multihop Mobile Networks for Multimedia Support," *ACM/Baltzer Mobile Networks and Applications*, vol. 3, no. 1, pp. 101-119, 1998.

[63] S. Ramanathan and R. Rosales-Hain, "Topology Control of Multihop Wireless Networks using Transmit Power Adjustment," *Proc. INFOCOM Conf. 2000*, pp. 404-413, 2000.

[64] V. Ramasubramanian, Z.J. Haas, and E.G. Sirer, "SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks," *Proc. MOBIHOC Conf. 2003*, pp. 303-314, June 2003.

[65] M.Q. Rieck, S. Pai, and S. Dhar, "Distributed Routing Algorithms for Wireless Ad Hoc Networks Using d-hop Connected Dominating Sets," *Proc. the 6th International Conference on High Performance Computing: Asia Pacific Region*, vol. 2, pp. 443-450, 2002.

[66] S. Roy and J.J. Garcia-Luna-Aceves, "Node-Centric Hybrid Routing for Ad Hoc Networks," *Proc. MASCOTS Conf. 2002*, Oct. 2002.

[67] P. Samar, M.R. Pearlman, and Z.J. Haas, "Hybrid Routing: the Pursuit of an Adaptable and Scalable Routing Framework for Ad Hoc Networks," *The Handbook of Ad Hoc Wireless Networks*, CRC Press, Boca Raton, FL, 2003.

[68] C. Santivanez, "Asymptotic Behavior of Mobile Ad Hoc Routing Protocols with respect to Traffic, Mobility and Size," *Technical Report* TR-CDSP-00-52, Communications and Digital Signal Processing Center, ECE Dept., Northeastern University, Boston, MA, 2000.

[69] C. Santivanez, R. Ramanathan, and I. Stavrakakis, "Making Link-State Routing Scale for Ad Hoc Networks," *Proc. MOBIHOC Conf. 2001*, Oct. 2001.

[70] C. Santivanez and R. Ramanathan, "Hazy Sighted Link State (HSLS) Routing: a Scalable Link State Algorithm," *BBN technical memo* BBN-TM-I30I, BBN Technologies, Cambridge, MA, Aug. 2001.

[71] C. Shen, C. Srisathapornphat, R. Liu, Z. Huang, C. Jaikaeo, and E. Lloyd, "CLTC: A Cluster-Based Topology Control Framework for Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 1, pp. 1-15, 2004.

[72] M. Steenstrup, "Cluster-Based Networks," in *Ad Hoc Networking*, C.E. Perkins, Ed., Addison-Wesley, Reading, MA, pp. 75-138, 2000.

[73] S. Srivastava and R. K. Ghosh, "Cluster Based Routing Using a k-Tree Core Backbone for Mobile Ad Hoc Networks," *Proc. the Sixth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Sept. 2002.

[74] R. Sivakumar, B. Das, and V. Bharghavan, "Spine Routing in Ad Hoc Networks," *ACM/Baltzer Publications Cluster Computing Journal, Special Issue on Mobile Computing*, 1998.

[75] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: A Core-Extraction Distributed Ad Hoc Routing Algorithm," *IEEE Journal on Selected Areas in Communications, Special Issue on Wireless Ad Hoc Networks*, vol. 17, no. 8, pp. 1454-1465, 1999.

[76] I. Stojmenović, "Position Based Routing in Ad Hoc Networks," *IEEE Communications Magazine*, vol. 40, no. 7, pp. 128-134, 2002.

[77] I. Stojmenović, 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, pp. 14-25, 2002.

[78] I. Stojmenović and J. Wu, "Broadcasting and Activity-Scheduling in Ad Hoc Networks," in S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, eds. *Ad Hoc Networking*, IEEE Press, 2004.

[79] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," *Proc. the Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 7-14, Aug. 1999.

[80] B. Williams and T. Camp, "Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks," *Proc. MOBIHOC Conf. 2002*, pp. 194-205, June 2002.

# VITA

Lan Wang
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

**Lan Wang** received his B.S. and M.S. degrees in Computer Science from Harbin Engineering University, China in 1992 and 1995, respectively. From 1995 to 1999, he worked as a software engineer at the System Engineering Research Institute of China State Shipbuilding Corporation (CSSC), Beijing, China. Since August 1999, he has been a Ph.D. student at the Computer Science Department of Old Dominion University.

Typeset using LaTeX.