Analytic Models and Distributed Robotics Applications for Mobile Ad Hoc Networks

by

İbrahim Hökelek

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2006

UMI Number: 3232009

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

09/14/2006 — Date

Prof. M. Ümit Uyar — Chair of Examining Committee

09/14/2006 — Date

Prof. Mumtaz K. Kassir — Executive Officer

Prof. M. Ümit Uyar, EE Department, CCNY (Chairperson)

_____

Prof. N. Tarek Saadawi, EE Department, CCNY

_____

Prof. Ibrahim Habib, EE Department, CCNY

_____

Prof. Kaliappa Ravindran, CSc Department, CCNY

_____

Dr. Mariusz Fecko, Telcordia Technologies, Inc.

_____

Supervisory Committee

The City University of New York

Abstract

Analytic Models and Distributed Robotics Applications for Mobile Ad Hoc Networks

by
İbrahim Hökelek

Advisor: Prof. M. Ümit Uyar

In this thesis, we introduce new analytic models to study node link stability for realistic wireless mobile network applications by calculating the exact link failure and creation probabilities. These models divide a geographic area into logical cells, where each node roams into one of its neighboring cells by following the discrete-time random walk mobility model. We calculate the probability distribution for a wireless link to be available between two mobile nodes. We derive a new two-dimensional Markov chain whose states represent a node's degree and its number of link failures. We can thus compute two important metrics characterizing the dynamics of a node's random movement: the expected times for the number of link changes to drop below a failure threshold, and for a node's degree to exceed a degree threshold. Because the model is capable of computing the dynamics and the expected value of the number of a node's neighbors, it can be used for modeling various realistic applications including virtual backbone and clustering stability, and estimating interference levels in mobile ad hoc networks (MANETs). Our modeling framework can be further extended to derive a number of additional important metrics to characterize

network connectivity, capacity, and survivability.

We also developed two real-life applications of our analytic models: Controlled Dissemination Filter (CDF) for MANETs and a testbed implementation for the Dynamic Survivable Resource Pooling (DSRP) concept in distributed robotics systems. We implemented the CDF framework, together with its architecture and protocols, and showed significant performance improvements related to the CDF without jeopardizing the network and application performance for different dissemination scenarios. We also implemented the DSRP mechanism in our FPGA-based distributed robotics testbed and demonstrated its effectiveness in a search-and-rescue scenario, where robots cooperate to provide the rescuer with a set of pictures to build a panoramic view of the disaster site. The measurements collected from the testbed and the numerical results obtained from the analytic models confirm that the DSRP framework significantly improve the reliability of distributed robotics systems.

Acknowledgments

I would like to express my sincere gratitude and thanks to my advisor and committee chair, Prof. M. Ümit Uyar for his guidance, suggestions and encouragement throughout the development of this thesis. I was fortunate to work with him for the past three years in which he provided me valuable knowledge. Until the completion of the thesis, he continued to sacrifice his spare time including many nights for our discussions. Without his expertise in the field, dedication, and sacrifices, this work could not have been done.

I am deeply grateful to Dr. Mariusz Fecko for his constructive comments on the thesis and his contributions to our collaborative work. I would like to thank Mr. Sunil Samtani for his support during my internship studies at Telcordia Technologies, Inc.

I would like to express my special thanks to the members of the thesis committee for showing keen interest to the subject matter and reviewing the thesis. Special thanks go to Prof. Tarek Saadawi for his support during my Ph.D. studies.

Without the scholarships and the financial support from the CUNY and CCNY, Curricular Practical Training opportunities at Telcordia Technologies Inc., the collaborative participation in the Communications & Networks Consortium sponsored by the U.S. Army Research Lab under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011, and the National Science Foundation grant ECS-0421159, I would not be able to concentrate on this research.

I would like to thank my M.S. thesis advisor Prof. Nail Akar of Bilkent University, Ankara, Turkey, who introduced me to the field of networking.

As I am completing this thesis at their home in New Jersey, special thanks go to my friends Dr. Kemal Karakayali and Mr. Mustafa Karakayali. I would like to thank my friends and colleagues Dr. Aycan Yurtsever, Mr. Murat Uysal, Mr. Jianping Zou, and Mr. Samrat Batth for their camaraderies during my graduate studies.

It is a great pleasure for me to mention that this work could not have been accomplished without the patience and support of my father Ömer, my mother Gülhanim, my three brothers Abdullah, İsmail, and Bekir, and my sister Meryem. I dedicate this thesis to them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mobile ad hoc networks (MANETs) (RFC2501) [26] suffer from a harsh communication environment due to node mobility and peculiarities of the wireless medium such as fading, interference, and asymmetric links. Since mobile nodes are often subject to random movement, the network topology may change rapidly and unpredictably. Typical MANET applications require survivability in case of rapid and unpredictable network topology changes. Reliability of services is critical but difficult to provide in MANETs, especially when applications such as military communications, real-time transactions, videoconferencing, and disaster recovery necessitate that the underlying network infrastructures are survivable. Developing accurate node link stability models to provide valuable insights into MANET behavior for these applications has been a challenging task for the researchers in wireless networking area [10, 22, 61, 90, 92, 97, 51, 4].

The *reliable server pooling (RSP)* [99, 101, 102] is one of the frameworks [46, 108] that address the reliability of services by introducing redundancy in the number of servers available to a client. It also provides an abstraction of all the functionally equivalent servers, whereby the client can access these servers as a single entity, termed *server pool*. Servers with the same functionality, also called *Pool Elements (PEs)*, are grouped into server pools identified by a pool handle. In the RSP, the *Name Servers (NSs)* are responsible for maintaining server pools, load balancing, and server discovery. The client, also called *Pool User (PU)*, resolves the mapping from a server-pool handle to the addresses of servers registered in this pool by querying its *Primary Name Server (PNS)*.

A popular approach to improve reliable service discovery in MANETs [37, 60, 65, 64] is to create clusters such as a *virtual backbone (VB)* [21]. A VB consists of a subset of nodes such that the backbone nodes are able to discover and communicate with one another. The VB nodes are dynamically selected in a distributed fashion. To handle frequent topology changes that may happen due to the node mobility, most backbone formation algorithms [59, 61, 90] include the maintenance feature that dynamically reassigns nodes to either join or leave the VB according to the number and stability of their links.

A new RSP architecture called *Dynamic Survivable Resource Pooling (DSRP)* [37, 36] deploys NSs on a dynamic VB for MANETs. In a mobile environment, DSRP is well-suited to dynamically provision distributed network managers that pool various servers for higher availability and failover: FCS [84] backbone managers, Bandwidth Brokers [8],

Situation Awareness (SA), Common Network Picture (CNP), SIP [85], and others. The reliable pooling can also be extended to resources such as hardware entities with specific capabilities (e.g., sensors used in robotics).

In the literature, there are no sufficient analytic models nor realistic simulations with multiple servers (i.e., PEs) are reported for service discovery over VB. A key end-user metric is the expected delay to resolve a service request. To quantify this delay, we have to analyze the DSRP architecture with respect to formation and maintenance of the VB, where the most stable nodes are dynamically selected as the backbone nodes.

In this thesis, we first present novel analytic models to study the stability of virtual backbones. These new models allow for the computation of two important metrics characterizing the dynamics of a node's random movement in MANETs: *(i)* average time for the number of link changes of the node (i.e., instances of link creation and failure) to either drop below or exceed a given threshold, and, analogously,*(ii)* average time for the number of active links of the node to either drop below or exceed a given threshold. The former defines the stability of the node's links, and the latter is the node's degree (the number of active neighbors).

These novel models capture the dynamics of the DSRP nodes (i.e., NS, PE, and PU) driven by random node movements. A VB formed and maintained by the algorithms in Ref. [61] constitutes a minimum connected dominating set: each node in the network is either a member of this set, or is only one hop away from a member. To preserve this

property in a mobile environment, the nodes frequently evaluate whether they should join, leave, or remain in the backbone by using two basic rules. The first selection rule applies the *normalized link failure frequency (nlff)*, which simply is the number of link losses within a time-window, but normalized by the number of total links at the end of the window. According to this rule, the nodes with *nlff* values higher than a given threshold cannot join the backbone. The second rule employs the comparison between the degrees of the remaining nodes to select the backbone nodes. Our models will help analyze the performance of various paradigms that rely on such link-based mobility metrics [21, 53, 59, 60, 106] including backbone assisted [90] or cluster-assisted [10, 22] routing protocols and service discovery architectures [60, 61]. In addition, certain routing protocols [97] and bandwidth estimation techniques [93, 92] make decisions based directly on link stability.

These models divide a geographic area into logical cells, where each node roams into one of its neighboring cells by following the discrete-time random walk mobility model. The wireless link creation and failure probabilities are derived to represent transitions from an available to an unavailable state (or vice versa) as nodes move. We derive a two dimensional Markov chain whose states represent the node degree and *nlff*. We calculate the expected first passage times to move between states defined by degree and *nlff* values. In our model, we can represent different network characteristics by varying the following three parameters: *(i)* the geographic area size, *(ii)* a node's communication range, and *(iii)* the number of mobile nodes.

These first analytic models use steady state approximation of link creation and failure probabilities to study node link stability in wireless mobile networks with different node densities. We then introduce a second set of analytic models which are capable of analyzing node link stability for more realistic MANET applications by calculating the exact link failure and creation probabilities. These new comprehensive models are scalable for realistic mobile networks, computationally efficient and relatively simple to apply, and, hence, can be used to evaluate the performance of algorithms and protocols at different layers of MANETs.

Another potential limitation of the first analytic models is the use of random walk mobility model which allows a node to roam into its neighboring cells with only a fixed speed. However, in real-life scenarios, a node may move in any direction with any speed (i.e., random way point). Our new comprehensive models enhance the random walk mobility pattern to cover the fast/slow node movements with different speeds. This extension will allow our model to analyze more realistic scenarios in MANETs where speed variation affects the system performance.

Our analytic models analyzing node link stability can be utilized for designing efficient data dissemination protocols and evaluating their performance in wireless mobile networks. Efficient data dissemination mechanism is one of key components for creating a successful MANET since each mobile node has a limited amount of energy for both performing their computational tasks and information transmission to other nodes. In

addition, the wireless links connecting mobile nodes have limited amount of bandwidth, and, therefore, it is essential that data dissemination protocol should require minimum computational energy, and that the amount of disseminated data should be minimized without jeopardizing the network and application performance. For this purpose, we introduced the concept of Controlled Dissemination Filter (CDF).

The Proactive Integrated Link Selection for Network Robustness (PILSNER) program at Telcordia Technologies, Inc. develops agent technologies that support automatic link selection over different and widely varying transmission paths to bypass network congestion and outages. To perform optimizations as the network changes dynamically, agents need to rely on information that is dispersed in the network. We implemented the CDF framework, together with its architecture and protocols for the PILSNER agents and showed significant performance improvements (e.g., total bandwidth savings for the relevant data dissemination) related to the CDF without affecting the performance of these agents. For example, Unicast Routing Control Agent (URCA) involves setting the OSPF link metrics that distributes traffic evenly within the network and utilizes the preferred routes. The choice is influenced by the network topology and the traffic demand between each source and destination pair within the network. The CDF is responsible for efficient traffic demand and capacity information dissemination to the URCA lead, which runs the global URCA algorithm to determine the optimum routes (hence corresponding link weights) for the overall network.

Distributed robotics systems are suitable candidates for node link stability analysis tools since wireless connections among mobile robots can become broken due to severe network stress, robot or link failures, and constant mobility. Typical applications of distributed robotics systems, such as real-time transactions, disaster recovery, and search-and-rescue operations, include exchanging large amounts of data among robots. As the wireless connections are frequently broken during a session, the traditional abort-and-restart approach often results in long delays and puts a heavy burden on end users for these applications.

Dynamic Survivable Resource Pooling (DSRP) is a new mechanism to improve the survivability and reliability in distributed robotics systems. DSRP is based on a virtual backbone (VB), which is a highly distributed, scalable, and survivable network, formed and maintained through one-hop beacons among mobile robots. We implemented the DSRP framework, together with its architecture and protocols, in a distributed robotics environment at the City College of the City University of New York (CCNY). The robotic nodes are referred to as smart tiny auto-configurable robots (STARs). The STAR nodes are built on Xilinx ML310 development boards powered by Virtex-II Pro FPGA device which have two on-chip 400 MHz embedded IBM PowerPC 405 (PPC405) processor cores [107]. The processors run Wind River VxWorks 5.5 Real Time Operating System (RTOS) [105] which provides deterministic timing required for time sensitive applications, as well as a small foot print suitable for embedded systems. In this adaptation of the VB framework, a pool of STARs is viewed as a single service endpoint, and, therefore, is able to provide

reliable services for distributed robotics applications. We introduce a search-and-rescue image retrieval application, as a proof-of-concept for using DSRP successfully in a distributed robotics environment. The current testbed implementation includes a total of 22 nodes emulating real-life mobile robots, where 2 nodes are FPGA boards and the remaining ones are laptop and desktop PCs. Using our earlier generic MANET models, we also introduce new analytic models to evaluate the performance of search-and-rescue applications in distributed robotics systems. The measurements collected from the testbed and the numerical results obtained from the analytic models confirm that the DSRP framework significantly improve the system reliability.

The rest of this thesis is organized as follows. Chapter 2 summarizes the literature review. In Chapter 3, we introduce our new analytic models for node link stability in mobile ad hoc networks and present related numerical and simulation results. Exact solutions for the models introduced in Chapter 3 are presented in Chapter 4. Application of these models to the CDF problem are studied in Chapter 5. A testbed implementation for distributed robotics applications of our models and measurements for image retrieval applications are presented in Chapter 6. The concluding remarks and future research directions for this work are provided in Chapter 7.

Chapter 2

Literature Review

2.1 Service Reliability and Link Stability

In MANETs, wireless connections among mobile nodes can become frequently broken due to severe network stress, node or link failures, constant mobility in the system, and peculiarities of the wireless medium such as fading, interference, and asymmetric links. The related issue is the service discovery in MANETs, where clients can automatically discover network services and also advertise their own capabilities to the rest of the network. Under MANETs' highly dynamic, multi-hop, and infrastructure-less characteristics, service discovery is a necessary mechanism for creating self-configurable networks. For a client which needs to use a certain service from one or more servers, service discovery can be

defined as mapping of a service class and an attribute list to a simple IP address or a group of IP addresses. Typical applications of MANETs, such as real-time transactions, disaster recovery, video-conferencing, and search-and-rescue operations, include exchanging large amounts of data among mobile nodes. Frequently broken wireless connections among nodes cause rapid and unpredictable changes in the network topology and thus service interruptions. The traditional abort-and-restart approach often results in long delays and puts a heavy burden on end users for these applications. Reliability of services and network survivability are critical but difficult to provide and maintain concepts in these environments.

System reliability can be achieved in different levels, from data transfer mechanisms in the transport and lower layers, where hardware, firmware or software units ensure successful data transmission, to upper layers (above transport) where an application is desired to continue its operation without (or with minimum) interruptions due to failures, mobility, or QoS degradation. The *reliable server pooling (RSP)* [99, 101, 102] mechanism is designed to deal with the reliability using the latter approach. RSP provides a redundancy in the number of servers available to an end user such that the user can access these functionally equivalent servers as a single entity, termed *server pool*. Servers with the same functionality (also called *Pool Elements (PEs)*), are grouped into server pools identified by a pool handle. In the RSP, the *Name Servers (NSs)* are responsible for maintaining server pools, load balancing, and server discovery. The client, also called *Pool User (PU)*,

resolves the mapping from a server-pool handle to the addresses of servers registered in this pool by querying its *Primary Name Server (PNS)*. In case of session failures (including connection, node, and link failures), most applications can be transparently switched to another server without restart.

A popular approach to improve (reliable) service discovery in MANETs [37, 60, 65, 64] is to create clusters such as a *virtual backbone (VB)* [21]. A VB consists of a subset of nodes such that the backbone nodes are able to discover and communicate with one another. The VB nodes are dynamically selected in a distributed fashion. To handle frequent topology changes that may happen due to the node mobility, most backbone formation algorithms [59, 61, 90] include the maintenance feature that dynamically reassigns nodes to either join or leave the VB according to the number and stability of their links.

Recently, we have defined and demonstrated a new RSP architecture called *Dynamic Survivable Resource Pooling (DSRP)* [37, 36, 38], which deploys NSs on a dynamic VB for ad hoc networks. In a mobile environment, the DSRP is well suited to dynamically provision distributed configuration/network managers that pool various servers for higher availability and failover: FCS [84] backbone managers, Bandwidth Brokers [8], Situation Awareness (SA), Common Network Picture (CNP), SIP [85], and others. We also extended the reliable pooling to resources that include servers, services, and hardware entities with specific capabilities (e.g., sensors used in robotics). In the literature [61], only simulations are available for single-PE discovery over VB. These simulations results do

not apply to multiple-PE discovery; moreover, no analytical models exist for either type of PE discovery over VB. A key end-user metric which needs to be analytically modelled is the expected delay to get service request resolved. To quantify this delay, we have to analyze two largely independent parts of the DSRP architecture, each incurring its own delays: *(i)* formation and maintenance of the backbone, where the most stable nodes are dynamically selected as the backbone nodes, and *(ii)* distribution of resource registrations, requests, and replies over the mesh of backbone nodes. The model presented in this study can be applied to the first part: the stability of an NS, i.e., the expected time for an NS to leave/join the backbone, is immediately available in our model; other related metrics can also be obtained by modest extensions to the baseline model. These include the probability of a PE/PU not having an operational PNS and the expected delay for a PE/PU to find another PNS when the previous one becomes unavailable.

There are a number of schemes that rely on the link-based mobility metrics [10, 22, 61, 90, 93, 97]. An associativity based routing (ABR) [97] protocol uses a route stability metric such that a preferred route consists of nodes whose associativity states imply stable links with neighbors. A distributed clustering algorithm MOBIC [10] exploits a novel mobility metric for selection of clusterheads. The metric is based on the ratio between the received power levels of consecutive transmissions from neighbors, and can be approximated in our model from the measure of link-state changes. In Ref. [93, 92], a bandwidth estimation algorithm computes the available bandwidth in a tunnel through an ad hoc network. The

algorithm relies solely on end-to-end packet measurements, with the accuracy strongly dependent on the stability of links along the network paths.

Ref. [61] proposes a distributed service discovery architecture that relies on a virtual backbone (VB) for locating and registering available services within a dynamic topology. Service discovery architectures in MANETs are classified into two main groups in Ref. [61]: directory-less and directory architectures. Servers and clients can communicate directly with each other in directory-less architectures, while directory agents (DAs) are needed to provide communication between users and servers in directory architectures. One disadvantage of the directory architecture is the requirement of the dynamic assignment for DAs, which increase not only system complexity but also creates an extra overload to the network. However, there are two main motivations for using a directory system. The first one is major advantages inherent to using DAs such as scalability, less response time for locating services, preventing to overload a server during too many service requests, and load balancing in the DA node. The second motivation follows from utilizing virtual backbones or clusters for better efficiency and quality for MANET routing protocols. In Ref. [61], the authors show that the directory architecture is a good candidate for service discovery in MANETs. There are two independent components in their solution for service discovery: *backbone management* and *distributed service discovery*. For a given network topology, a dynamic backbone is formed in the backbone management phase such that each node in the network is either a part of the backbone or one hop away from at least

one of the backbone nodes. Finding virtual links among the selected backbone nodes and adapting the backbone nodes to the topology changes are performed in this part as well. In the distributed service discovery part, the request and registration messages from the service discovery agents to the backbone nodes are distributed efficiently by forming a multicast tree.

VB formation algorithms apply node's degree and *nlff* [61] to resolve conflicts between the nodes competing to join the backbone. Similar to Refs. [60, 61], CEDAR [90] builds a routing core such that each node that needs to find a dominator (i.e., backbone or core node) selects the highest-degree node with the maximum effective degree in its first neighborhood. To preserve the connected dominating set property in a mobile environment, the nodes frequently evaluate whether they should join, leave, or remain in the backbone or routing core. The first selection rule in [61] says that the nodes with *nlff* values higher than a given threshold are eliminated from joining the backbone. The second rule in [90, 61] employs the comparison between the degrees (or effective degrees) of the remaining nodes to select backbone nodes. We approximate both rules by comparing *nlff* and node's degree with constant thresholds (i.e., *nlff* and node degree thresholds. Hence the metrics computed by our model are directly applicable to the backbone formation algorithm used in the DSRP.

Ref. [3] proposes a simplified random walk model capturing the movement of mobile users in Personal Communications Services (PCS) networks. In that study, the service

areas are covered by radio base stations whose radio coverage is called a cell, where they configure cells as hexagonal or mesh networks. In Ref. [3] only mobile station can move in one unit time, and link availability from a fixed base station to this mobile station is calculated. Similarly, Ref. [98] uses the discrete-time random walk model to predict the route lifetime in multihop mobile ad hoc networks. However, two mobile stations can move in one unit time, and a vector representing a wireless link between two mobile stations is called a *state*. A random walk model is applied to formulate how a wireless link changes states, where there are 19 possible combinations for the next link state after the state reduction technique is applied. They find these link states, together with their associated probabilities, from the discrete-time random walk model and develop the state transition diagram using the state transition probabilities. A state transition matrix M such that each element $M_{i,j}$ represents the probability to transit from the i-th state to the j-th state within one time unit is constructed. The M matrix is used to develop several probabilistic functions. Finally, they calculate the expected route lifetime using these probabilistic functions. Our model starts with Ref. [98] and derives a new model for computing useful node link stability metrics as described in Chapters 3 and 4.

Among the more recent analytic models reported in the literature, [51] characterizes key performance measures of cellular networks such as mean handover rate and mean sojourn times from the point of view of an arbitrary cell using the random way point mobility model. In [4], long-run location and speed distributions of a mobile node over one dimen-

sional regions such as highways are presented. This generalized random mobility model of [4] can capture realistic movement profiles and is numerically efficient to be used for calculating long-run location and speed behavior. In this thesis, we study more generic metrics such as node degree and *nlff* using relative distances (i.e., link states between two mobile nodes) changing with node movements in two dimensional regions. A distributed approximation algorithm for computing the minimum connected dominating set (MCDS) in a unit disk graph is proposed in [6], where MCDS can be used for reducing the communication overhead, increasing the convergence speed, and simplifying the connectivity management for MANETs. The size of the MCDS depends on the network density and nodes' locations and may change with random node movements.

2.2 Data Dissemination Mechanisms

Efficient data dissemination mechanism is one of key components for creating a successful MANET since each mobile node has limited amount of energy and bandwidth. Hence, it is essential that data dissemination protocol should be controlled. The main objective is to deliver information in an efficient manner by minimizing the dissemination of unnecessary data over the limited available bandwidth without adversely affecting the network and application performance. For example, PILSNER program at Telcordia Technologies, Inc. develops agent technologies that support automatic link selection over different and widely

varying transmission paths to bypass network congestion and outages [40]. To perform optimizations as the network changes dynamically, agents need to rely on information that is dispersed in the network. Efficient data dissemination mechanism specifically designed for each agent by considering their application requirements is defined as a key requirement for the success of this program.

Publish/subscribe (pub/sub) system [34, 20] together with the concept of the multicast channel is one common method heavily used for an efficient data dissemination in MANETs, where there are three main entities: publishers, subscribers, and information (e.g., events or data flows). Publishers are the source entities which provides, advertise and subsequently publish their information to the network. Subscribers are the destination entities which are interested in receiving information by subscribing their interests.

In general, the optimal data dissemination problem can be split into three main subproblems: *(i) Data grouping* is to divide the data types into a set of logical data groups. The challenge is to determine the granularity and hence the number of diverse data groups to minimize the overhead of data distribution [73, 87, 1]. *(ii) Channelization* is to find an optimal mapping of information flows and data recipients to the set of multicast trees given the set of data groups, the associated information flows, and a fixed number of data distribution structures (e.g., multicast trees) [1, 73]. *(iii) Filter placement* is to define an optimal placement of filters at both the end nodes and the intermediate nodes of a multicast tree given the sets of multicast trees and the filtering rules [87, 15].

One can interpret the above problems as finding an optimum between two nave approaches. The first approach would be to create a single data group, say *CDF data*, build a multicast tree that includes all the potential recipients in the group, and then perform all data filtering at end points. This would result in unnecessary data dissemination overhead and hence wasted bandwidth. The second approach is to create a multicast tree for every data type, which would lead to better bandwidth utilization, but would not scale due to complexity and overhead of maintaining a large number of multicast groups.

The channelization problem is defined in a DARPA-sponsored work [1]. Since multicast groups require resources (e.g., router state) and management overhead (e.g., to set up and maintain the multicast routes) it is often not feasible or desirable to allocate a separate multicast group to each flow. With a limited number of multicast groups that will be created, the channelization problem is to find an optimal mapping of information flows to a fixed number of multicast groups and a mapping of receivers to multicast groups so as to minimize a cost function involving the total bandwidth consumed and the amount of unwanted information received by the data recipients. This problem has been proven to be NP-hard; however, different approximation algorithms are available that find good solutions over a range of problem configurations. Specifically, the mapping must be such that all data needed by a user is mapped to one or more multicast groups to which the user either subscribes or is designated as the data recipient (*no false exclusion*). The mapping should also be such that the amount of unneeded data received by users

belonging to various multicast groups carrying the needed data is minimized (*minimum false inclusion*). Our CDF design starts with similar channelization formulations but also considers the geographic locations of the users into the objective function. The users, which are interested in the similar data groups, are assigned to the same multicast tree if they satisfy a certain distance requirement to one of the group members.

Ref. [73] creates a set of dissemination channels, each being an independent tree of brokers, and each one containing different content. The content of each channel is specified by the collection of profiles of all the clients attached to the specific channel. A coordinator node collects user interests and is responsible for performing channelization and managing the dissemination channel. Source and gateway brokers are used for aggregating publishers and user interests, respectively; and hence, the channelization algorithm is run for less number of nodes. After mapping data and user interests to multiple multicast trees, the data is delivered from source brokers to gateway brokers without performing any intermediate filtering. Their algorithm is centralized and requires collecting all information to a coordinator node.

The filter placement problem is studied in Ref. [87], where there is a single multicast tree from a source to multiple destinations and a fixed number of filters (software modules) is used to avoid unnecessary data transmission. The problem is to place a fixed number of filters into appropriate intermediate nodes such that the maximum bandwidth savings is obtained. They propose an optimal centralized filter placement and reduction algorithm

and a suboptimal filter placement and reduction heuristic. While the optimal algorithms calculate a new filter placement for each batch, the heuristic approach allows only one filter to be moved for each batch. In our CDF design, we relax the number of filters to be placed and propose a distributed filter placement and reduction algorithm.

Data dissemination from one source to multiple intended users is proposed using a single broadcast tree in [57]. The broadcast tree is setup using inefficient flooding mechanism and reorganized in larger time scale when needed. It is hybrid in the sense that they use inefficient flooding for setting up the broadcast tree and use efficient data dissemination over this broadcast tree. There is no need for pub/sub system in this approach since all data will be disseminated to all nodes in an optimized broadcast manner. Similarly, Ref. [15] propose a new method called *Kyra* which balances trade-offs between two approaches: filter-based and channelization-based approaches. They combine the advantages of content-based filtering and event-space partitioning in the existing approaches to achieve better overall routing efficiency. The main idea is to construct multiple smaller routing networks, so that filter-based routing is implemented in each one with lower cost. In this architecture, the servers are organized into server cliques based on their network proximity. Servers in the same clique know about each other and communicate through unicast. At the second level, multiple routing trees are built, each for routing a subset of events. Our CDF design including the DSRP concept has a spanning tree consists of overlay VB nodes and further improves dissemination efficiency by deploying a distributed

filter placement and reduction algorithm for filtering out unnecessary data.

Ref. [47] shows that no one data dissemination protocol can be optimized for all applications; instead, a family of protocols are needed, with guidance to match protocol to application. They present two new dissemination algorithms: push and one-phase pull diffusion. Our CDF design includes several different dissemination functionalities, where one can be selected for a particular application depending on its data requirement.

2.3 Distributed Mobile Robotics Systems

In recent years, the emphasis of robotics research has shifted from the control of individual robot to more challenging topics such as distributed robotics, swarm robotics, and mobile sensor networks. The vision is that a large number of well-coordinated miniature robots can gather sensory information from multiple viewpoints simultaneously, allowing the robotic system to understand the environment more quickly and comprehensively. Miniature robots in cola-can size scale have many advantages over expensive large size robots for particular applications. Low-cost miniature robots can operate in difficult-to-access areas, and hide in inconspicuous places to evade detection. The small size and large number also make individual robots expendable without jeopardizing the overall mission. Robustness is enhanced since the failure of any one robot can be compensated for by other team members. These advantages can be exploited in many applications such as re-

connaissance, environmental monitoring, search and rescue, and tool/weapon delivery to confined places, whereas a single large robot may be unable to gain access or may perform poorly. The deployment of multiple robots in complex environments creates demands for reliable communication protocols in order for the user to access the information collected by multiple robotics. While most fields of distributed robotics have progressed based on rapid advances in computing and information systems, distributed, efficient and reliable communication protocols to coordinate hundreds of tiny robotics and to support service discovery among them is still a major issue for the robotics' researchers.

There has been increasing interest in systems composed of a group of distributed robots working cooperatively on a common task [16], including security and surveillance [81], cooperative transport [18, 69], exploring unknown environments [89, 95, 50, 74, 13] and playing soccer [24]. In these applications, if robots can communicate and coordinate their actions with others, the tasks can be accomplished much more effectively but maintaining communication among the robots is a major task. To support these applications, algorithms have been designed to guarantee a complete coverage of the free space by the mobile robots [79], inter-robot communication [2, 43], and resource scheduling [72]. Robomote [88] and CotsBots [11] are examples of such systems, based on MICA boards [48] that, together with sensing and motor control stacks, constitute a wireless sensing nodes. In some applications, robots are assumed that their wireless communication are constrained by line of sight of each other [95, 9, 78, 7, 83]. Another strategy is to employ

leader-follower relationships between the robots [95, 9, 17]. Other approaches include communication based on planned and reactive behavior [104], signal strengths [96, 78], embedded networks [71].

The Center for Distributed Robotics at the University of Minnesota has been actively involved in research in the design of usable miniature robotic systems [31]. Targeting at application areas of surveillance, urban reconnaissance, urban search and rescue, and close quarter inspection, they have developed various small robotics, such as UMN Scout [31, 49, 55, 82, 75], COTS Scout [52, 30, 62], MegaScout [103], eROSI and CRAWLER Scout [91]. Those Scout series robotics are equipped with different wheels for various movement styles, with cameras to sensor the environments, and communication channels to accept commands and transit video signal back.

Compared with these successful platforms, our distributed robotics system utilizes high-end FPGA devices to achieve more powerful onboard computing capability, as well as hard/soft reconfigurability. To ensure reliability of services for mission critical distributed mobile robotics systems, STARs nodes utilize the DSRP concept to create and maintain a reliable service pool in the distributed robotics environment such that there would not be a single point of failure. In DSRP framework, each robot that provides a service is a pool element (PE) and the user that receives the service is a pool user (PU). The Name Server robots (NSs) chosen from the PE population are responsible for maintaining the PE pools, load balancing, and PE discovery. The PU resolves the mapping from a server-pool handle

to the addresses of PEs registered in this pool by querying its NS. In DSRP, whenever a PE fails, PUs that utilize that PE should transparently switch over to another PE in the pool, possibly migrating the session to the new PE, and hence providing a survivable access to resources in a mobile robotics network. All reliability related protocol signaling, such as VB and DSRP control messages are transported over UDP, as reliability of the periodic control messages is not mandatory for the success of the VB formation and DSRP service registrations and requests.

For communication among two neighboring robots several alternatives are possible, ranging from a low-power short-range communication technology, such as Bluetooth [45, 44], and IEEE 802.15.4 [42] for wireless sensor networks to a high-power long-range technology such as the IEEE 802.11 wireless LAN [29, 27] protocol. Bluetooth's key features are low complexity, low power and low cost, which operates in the unlicensed ISM band at 2.4GHz, avoiding interference from other signals by hopping to a new frequency after transmitting or receiving a packet. Although both Bluetooth and IEEE 802.15.4 consume less power, they have the disadvantages of low bandwidth and short communication range. To the best of our knowledge, there is no industry-wide standard hardware and protocol stack currently available to be utilized as a multihop routing mechanism for either Bluetooth or IEEE 802.15.4 in mature embedded platforms such as Xilinx ML310 [107]. For these reasons, we use the IEEE 802.11 wireless LAN protocol for the wireless communication among the robots, in spite of its disadvantages such as relatively larger size and higher

power consumption. As either Bluetooth or IEEE 802.15.4 hardware providing a multi-hop communication becomes commercially available for Xilinx platform [107], we plan to investigate utilizing their low cost and low power consumption features as candidates for wireless communication among the robots.

Typically, mobile ad hoc network protocols use two types of routing protocols to extend the geographical coverage of mobile robots beyond one-hop neighbors: (i) proactive routing such as DSDV [77], CGSR [80], WRP [70], and (ii) reactive routing such as AODV [19], DSR [54], LMR [25], TORA [76], ABR [97] and SSR [32]. In proactive routing, information about all paths are kept updated in each node, whereas in reactive routing the paths are established from a source node to a destination on-demand. We use a reactive routing protocol AODV in our design to reduce the overhead introduced by routing protocols and due to its proven efficiency in distributed robotics applications.

Chapter 3

Novel Analytic Models for Node Link Stability

In this chapter, we present new analytic models to study node link stability in MANETs by computing two important metrics that characterize the dynamics of a node's random movement: *(i)* average time for the number of link changes of a node to either drop below or exceed a given threshold (i.e., stability of a node's links), and, analogously, *(ii)* average time for the number of active links (i.e., the number of active neighbors) of a node to either drop below or exceed a given threshold.

Tseng et al. [98] propose a probabilistic model based on Markovian link state changes using the random-walk movement. Using this approach as a starting point, we develop models

for continuous link creation and failures in mobile ad hoc networks. We analytically derive link creation and failure probabilities between two randomly moving mobile nodes, the probability distribution and the above-mentioned mobility metrics for the node degree, and link creation and failure. Except for the mobility model used only to calculate the link creation and failure probabilities, our model is generic and independent of the underlying mobility patterns. Our approach can thus be used for different mobility models [14] provided that their link creation and failure probabilities can be derived from the rules that determine node movements.

We first construct a state transition matrix M (extended from Tseng et al [98]) where each element $M_{i,j}$ represents the probability to transit from the i-th to the j-th link state within one time unit. Second, by using the steady state probabilities for M, we derive a new Markov chain whose states represent the number of active links for a specific node (i.e., the node degree). In our new Markov chain, there may be multiple link arrivals and departures in one time unit, which we characterize by a pair of random variables. Based on these random variables, we obtain the stochastic point process that defines the probabilities of a node's degree to drop below or exceed a threshold degree as the node moves. The expected time of this event determines when a VB node turns into a non-backbone node (and vice versa). Because most VB formation algorithms [59, 61, 90] give the preference to nodes with the small number of link changes or the high degree, the link arrivals and departures determine the probabilities (and thus the expected times) for a

node to leave, join, or remain in the backbone. These times characterize the stability of the dynamic structure of a VB.

In Section 3.1, we introduce our new analytic models for node link stability. The first passage analysis for our proposed models is described in Section 3.2 We present numerical and simulation results in Section 3.3 and 3.4, respectively.

3.1 New Node Link Stability Models

3.1.1 New State Transition Diagram for Mobility Model

Our goal is to obtain statistical metrics for the node degree (the number of active neighbors) and its stability (the expected time that a node will have a certain number of active neighbors). We extend the model in Ref. [98], which only studies a link failure to predict the lifetime of a routing path by considering both link creation and failure models simultaneously. Therefore, their state transition diagram is more limited than ours as explained below.

The geographic area is partitioned into hexagonal cells as shown in Fig. 3.1 where a mobile node can move in from a neighboring cell within one time unit. A vector representing a wireless link between two mobile stations is called a *state*. In Fig. 3.2, for a mobile node N1 in location (0,0) and another mobile node N2 in location (x,y), the link state between these

nodes is $< x, y >$ (i.e., $< x - 0, y - 0 >$). A random walk model is applied to formulate how a wireless link changes states. Consider the wireless link $< x, y >$ connecting mobile nodes N1 and N2: after one time unit each mobile node moves into one of its six neighbor cells with probability of 1/6 for each direction as shown in Fig. 3.2. For example, if N1 and N2 move into the neighboring cells in the directions of D4 and D3, respectively, then the wireless link between these nodes will be $< x+1, y >$ in the next time unit. However, if they move in the same direction (e.g., D1), the wireless link $< x, y >$ will remain the same in the next time unit. There are 36 possible next link states, and as some of them will result in the same vector, there are only 19 possible combinations for the next link state. These vectors, together with their associated probabilities, are shown in Table 3.1.

The transmission range of a mobile host, currently resident in cell (0,0), will be modeled by the number of layers that it can reach. Cells are grouped into layers such that cell (0,0) is at layer 0, the six cells neighboring cell (0,0), namely cells (0,1), (1,0), (1,-1), (0,-1), (-1,0), and (-1,1), are at layer 1. In general, the cells surrounding the cells at layer i are on layer $i+1$. Fig. 3.1 shows a 4-layer network, where adjacent cells with the same color are in the equivalent layer (e.g., link states $< 3, 0 >$, $< 3, -1 >$, $< 3, -2 >$, $\ldots$, $< 2, 1 >$ are in layer 3). We should note that all possible link states in Fig. 3.1 can be represented by the cells, $< 0, 0 >$, $< 1, 0 >$, $< 2, 0 >$, $< 1, 1 >$, $< 3, 0 >$, $< 2, 1 >$, $< 4, 0 >$, $< 3, 1 >$, and $< 2, 2 >$, shown with thicker boundaries (i.e., the state reduction). Each layer may have a different number of link states after the state reduction technique applied. For

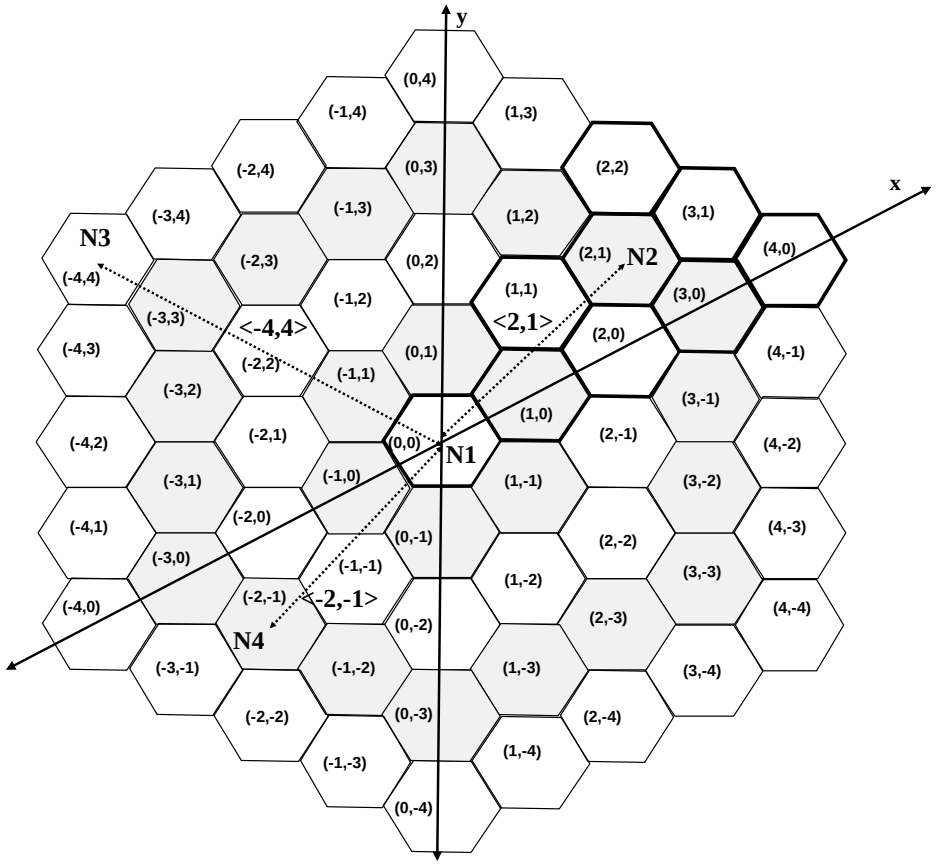

Figure 3.1: Cellular area and link state reduction

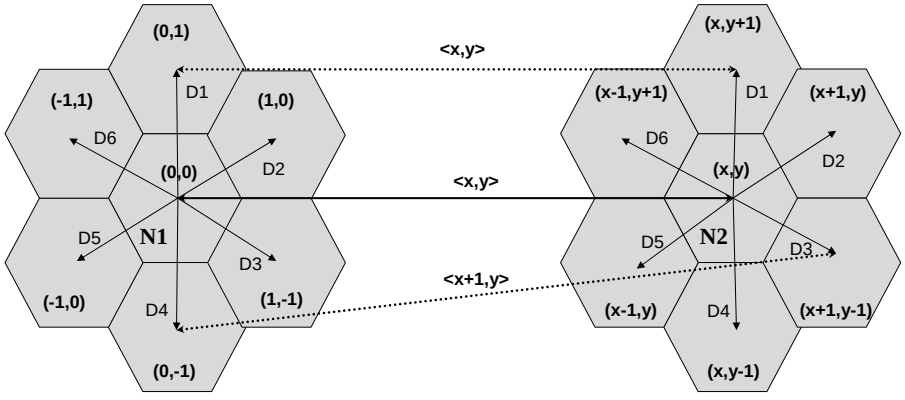

Figure 3.2: Example of link state changes

example, while layers 0 and 1 have only one state each (states $< 0, 0 >$ and $< 1, 0 >$, respectively), layer 20 has 11 states ($< 20, 0 >, < 19, 1 >, \ldots, < 10, 10 >$). In Fig. 3.1, the link state between N1 and N2 is $< 2, 1 >$ which belongs to layer 3, while the link state between N1 and N3 is $< -4, 4 >$ and can be represented by the link state $< 4, 0 >$, and hence belongs to layer 4. Similarly, the link state between N1 and N4 is $< -2, -1 >$ and can be represented by $< 2, 1 >$. The state reduction allows for representation of the equivalent states with a single state, and hence alleviating the computational cost. We develop the state transition diagram using the state transition probabilities from Table 3.1. Our model extends the number of layers in the state transition diagram shown in Fig. 3.3, where n_{tot} represents the number of layers, and determines the maximum distance between two mobile nodes. For simplicity, we assume that n_{tot} is an even number ($n_{tot} = 2i$) in Fig. 3.3. In the state transition diagram, there are certain transition probabilities among the link states which belong to layers less than or equal to n_{tot}. The exact transitions, with associated probabilities, from a given state can be derived from Table 3.1. To simplify the drawing, the transitions only from the link states of layers less than or equal to 5 are shown in Fig. 3.3.

For a given number of mobile nodes in a geographic area, all links may become active or inactive after some time units. Hence, our model needs to consider all possible link states with their corresponding transition probabilities in a state transition diagram. Let R be the center-to-center distance between two neighboring hexagonal cells. Then, without

Table 3.1: Probability distribution for a wireless link to switch from state $< x, y >$ to state $< x', y' >$

x',y'	x,y	x-1,y	x-1,y-1	x,y-2	x+1,y-2	x+1,y-1	x+1,y
Probability	6/36	2/36	2/36	1/36	2/36	2/36	2/36
x',y'	x,y-1	x+2,y-2	x+2,y-1	x+1,y+1	x,y+1	x+1,y	x,y+2
Probability	2/36	1/36	2/36	2/36	2/36	1/36	1/36
x',y'	x-1,y+2	x-1,y+1	x-1,y+2	x-2,y+1	x-2,y		
Probability	2/36	2/36	1/36	2/36	1/36		

loss of generality, the center-to-center distance from a mobile node currently resident in cell (0,0) to other nodes will be at most $n_{tot} \times R$. Let n_{av} represent the number of layers whose link states belong to available link states. The link state between two mobile stations is called an *available* link state if they are communicating with each other (i.e., one mobile station is in the other mobile station's radio coverage), otherwise an *unavailable* link state. Let a link state between two mobile stations be $< x, y >$ where x and y are integers ($-n_{tot} \leq x, y \leq n_{tot}$). Note that using the state reduction technique presented in Ref. [98], $< x, y >$ has an equivalent state $< x_e, y_e >$ such that x_e and y_e are non-negative integers ($0 \leq x_e, y_e \leq n_{tot}$). For example, the link state between N1 and N3 in Fig. 3.1 is $< -4, 4 >$ and its equivalent state is $< 4, 0 >$, which belongs to layer 4 (4+0=4). The link state between these two mobile stations is *available* if $x_e + y_e \leq n_{av}$, otherwise it is *unavailable*. If n_{av} is equal to 3, the link state between N1 and N2 is available (i.e.,

$2+1=3 \leq n_{av}$) and the link state between N1 and N3 is unavailable (i.e., $4+0 > n_{av}$).

We define a link state $< x, y >$ as a *feasible* state for $x_e + y_e \leq n_{tot}$, otherwise the state is called *infeasible*. From Table 3.1, observe that a feasible link state which belongs to layer n_{tot}-1 or n_{tot} may move into an infeasible link state (i.e., into a link state of layer n_{tot}+1 or n_{tot}+2) in the next time unit. However, to consider the case where the center-to-center distance between two mobile nodes is limited to $n_{tot} \times R$, only transitions among feasible states are allowed in the state transition diagram. Specifically, assume this link state has transitions to α feasible and β infeasible states, where $\alpha+\beta=19$. The sum of the transition probabilities of moving from a given state to the β infeasible states is distributed among the transition probabilities of moving from the given state to the α feasible states. This sum is distributed proportionally with the original transition probabilities of moving from the given state to the α feasible states. And then, we eliminate the transitions from this feasible state to β infeasible states by assigning zero probability. As a result, when the center-to-center distance between two mobile nodes is equal to $(n_{tot} - 1) \times R$ or $n_{tot} \times R$ in the current time, it will be less than or equal to $n_{tot} \times R$ in the next time unit.

3.1.2 Link Failure and Creation Models

The state transition diagram shown in Fig. 3.3 can be mapped to a state transition matrix M where $M_{i,j}$ represents the probability of moving from the i-th state to j-th state within one time unit. For a wireless link with initial state i initially (either available

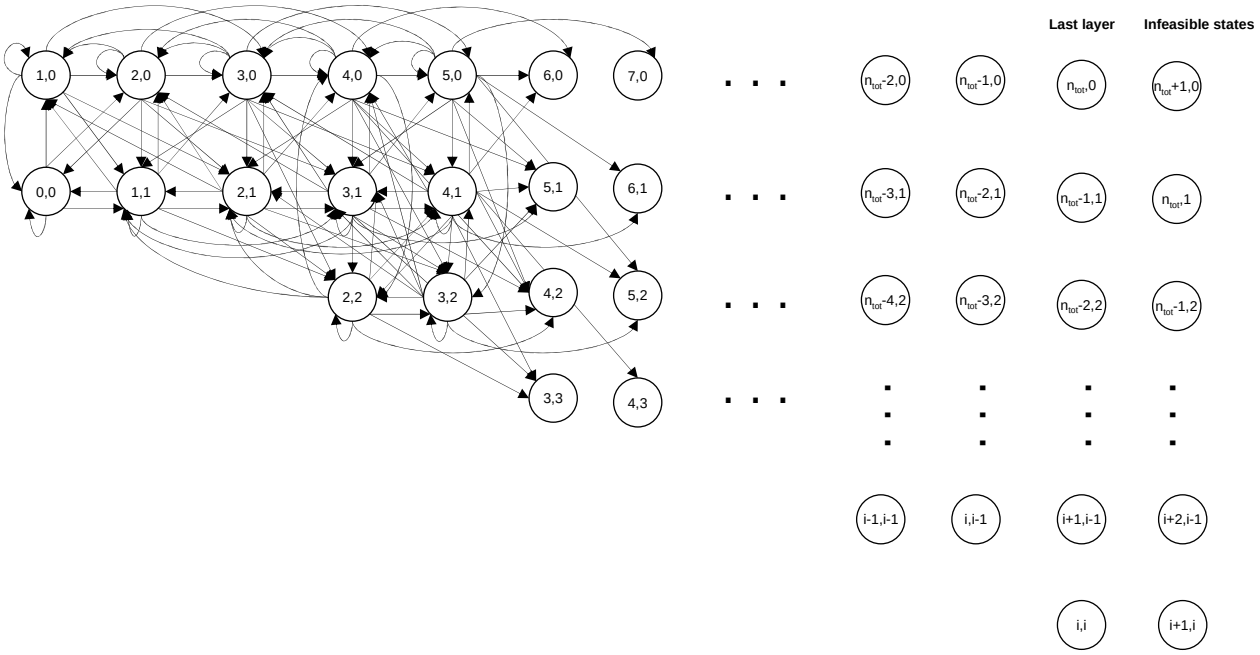

Figure 3.3: State transition diagram of a wireless link

or unavailable), let $P_a(i)$ and $P_u(i)$ denote the probabilities that the link will be available or unavailable in the next time unit, respectively:

$$P_u(i) = \sum_{j=s_a+1}^{S_T} M_{i,j} = 1 - \sum_{j=1}^{s_a} M_{i,j} = 1 - P_a(i) \tag{3.1}$$

$$P_a(i) = \sum_{j=1}^{s_a} M_{i,j} \tag{3.2}$$

where j represents available link states for $1 \leq j \leq s_a$ and unavailable link states for $s_a + 1 \leq j \leq S_T$.

After the state reduction, the number of available link states, s_a, depends on n_{av}:

$$s_a = \begin{cases} 1 & \text{if } n_{av} = 0 \\ \dfrac{(n_{av}+1)\times(n_{av}+3)}{4} & \text{if } n_{av} > 0 \text{ and } \ n_{av} \text{ is odd} \\ \dfrac{n_{av}\times(n_{av}+4)}{4}+1 & \text{if } n_{av} > 0 \text{ and } \ n_{av} \text{ is even} \end{cases} \tag{3.3}$$

Similarly, the number of all possible (available and unavailable) link states, S_T, can be found by replacing n_{av} with n_{tot} in Eq. (3.3) (The proof sketch is given in Theorem 1 of Section 4.1.3).

Consider a particular node z located in cell (0,0) of a network with N nodes. There are $K = N - 1$ possible bi-directional links from z to all other nodes in the network. If node z has initially k available links, then it has $K_u = K - k$ unavailable links. The following formulation is presented for this particular node z with k available and $K_u = K - k$ unavailable links.

First, we will calculate the probability of having k+1 available links in the next time unit, and then we will generalize the result for calculating the probability of having k+h available links, where $0 \leq k + h \leq K$. The latter one will provide us with the transition probabilities of a new Markov chain whose state is the number of available links for a particular node (i.e., the transition probability of moving from k-th to $(k+h)$-th available links).

The node z will have k+1 available links if l links out of k available links disappear and l+1 links out of K_u unavailable links appear in the next time unit for $l = 0, 1, \ldots, k$.

Let $P_{dap}(k,l)$ denote the probability that l of k available links will disappear and k-l of k available links will remain available. Then,

$$P_{dap}(k,l) = \sum_{i=1}^{s_a} f_s(i) \times \binom{k}{l} \times P_u^l(i) \times P_a^{k-l}(i) \tag{3.4}$$

where $f_s(i)$ is the probability density function of the available link states for $1 \leq i \leq s_a$. Here $P_u(i)$ and $P_a(i)$ denote the probability of becoming unavailable and available, respectively, in the next time unit given that the link is in state i initially and $P_u^l(i)$ represents the l-th power of $P_u(i)$.

Similarly, let $P_{ap}(K_u, l+1)$ denote the probability that $l+1$ of K_u unavailable links will appear and $K_u - l - 1$ of K_u unavailable links will remain unavailable in one time unit. Then,

$$P_{ap}(K_u, l+1) = \sum_{j=s_a+1}^{S_T} f_s(j) \times \binom{K_u}{l+1} \times P_a^{l+1}(j) \times P_u^{K_u-l-1}(j) \tag{3.5}$$

where S_T represents total number of possible link states and $f_s(j)$ represents the probability density function of the unavailable link states for $s_a + 1 \leq j \leq S_T$.

Eqs. (4.5) and (3.5) assume that all available links are in the same state of i and all unavailable links are in the same state of j, respectively. Suppose, for example, there are 10 available links whose initial states are $i_1, i_2, \ldots, i_{10}$. $P_a(i)$ and $P_u(i)$ in Eq. (4.5) will most likely be different for each link state i if the state transition matrix $M_{i,j}$ for one time unit is used without the stationary distribution. Then, for example, the probability that 3 links will disappear in the next time unit can be calculated by adding the probabilities of $\binom{10}{3}$=120 possible combinations. One of these combinations is that links i_1, i_2, and i_3 will

disappear and the remaining links will stay available in next time unit. This probability can be calculated as follows:

$$P_u(i_1) \times P_u(i_2) \times P_u(i_3) \times P_a(i_4) \times P_a(i_5) \times \ldots \times P_a(i_{10}) \tag{3.6}$$

Without loss of generality, using the steady state values of the state transition matrix $M_{i,j}$ will simplify the expression in (4.11) where $P_a(i)$ and $P_u(i)$ will be P_a and P_u, respectively, for each link state i without depending on the initial link states. Another benefit of using steady state probabilities is as follows. There are some unavailable link states, from where the probabilities of moving to an available state in one time unit is zero. For some initial unavailable link states, the probability of going from an unavailable state to all available states in one time unit is zero when $P_a(i)$ and $P_u(i)$ are used. However, these unavailable link states, after certain time, may become available with a non-zero probability. In order for this non-zero probability to be accounted for in our model, we need to use the stationary distribution of $M_{i,j}$.

The final metric we seek to compute in this analysis is the expected first times to reach a certain number of available links for a particular node. Instead of the steady state probabilities of M, if we were to use different M^m at the m-th time unit, the results for the expected time would depend on the initial state. However, we expect that, when averaged over different starting points, the results would approach the steady state analysis that we conduct in this study. Furthermore, if the initial values of the number of links were considered, the formulation of the problem would become prohibitively more complex

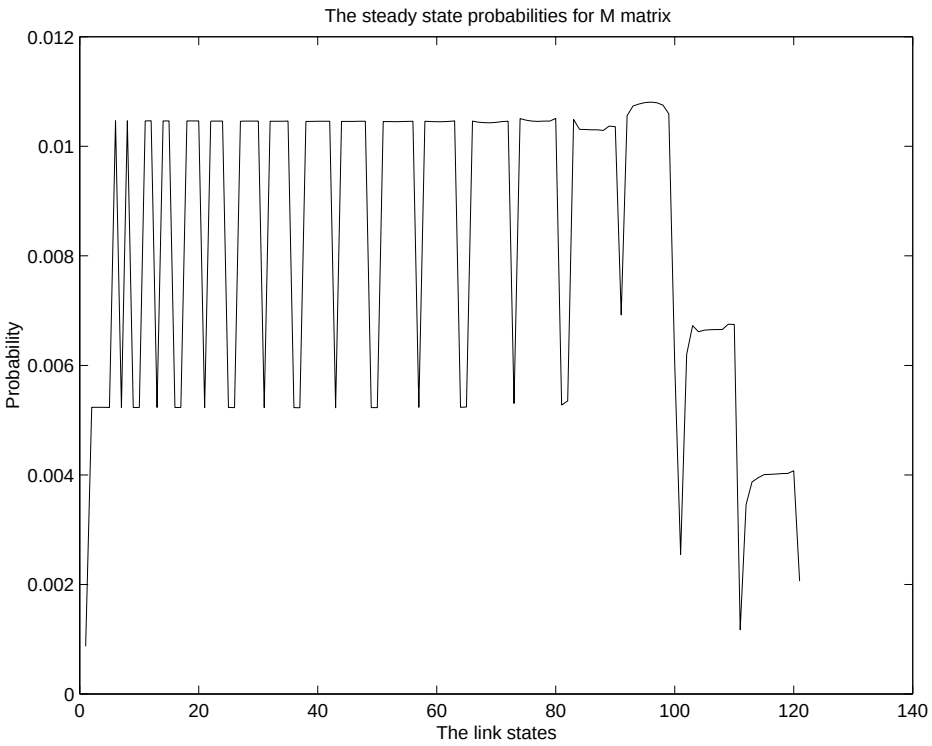

Figure 3.4: Steady state probabilities for M matrix.

without yielding significantly different results than our approximation.

The steady state values of $P_a(i)$ and $P_u(i)$ are calculated as:

$$P_u(i) = \sum_{j=s_a+1}^{S_T} M_{i,j}^{\infty} = 1 - \sum_{j=1}^{s_a} M_{i,j}^{\infty} = 1 - P_a = P_u \tag{3.7}$$

$$P_a(i) = \sum_{j=1}^{s_a} M_{i,j}^{\infty} = P_a = 1 - P_u \tag{3.8}$$

The steady state probabilities of the M matrix are shown in Fig. 3.4. Here the M matrix is constructed for n_{tot}=20, therefore there are 121 possible link states (i.e., available or unavailable). The steady state probabilities of the link states in the last two layers are smaller due to the bouncing back effect which arises when we construct the M matrix.

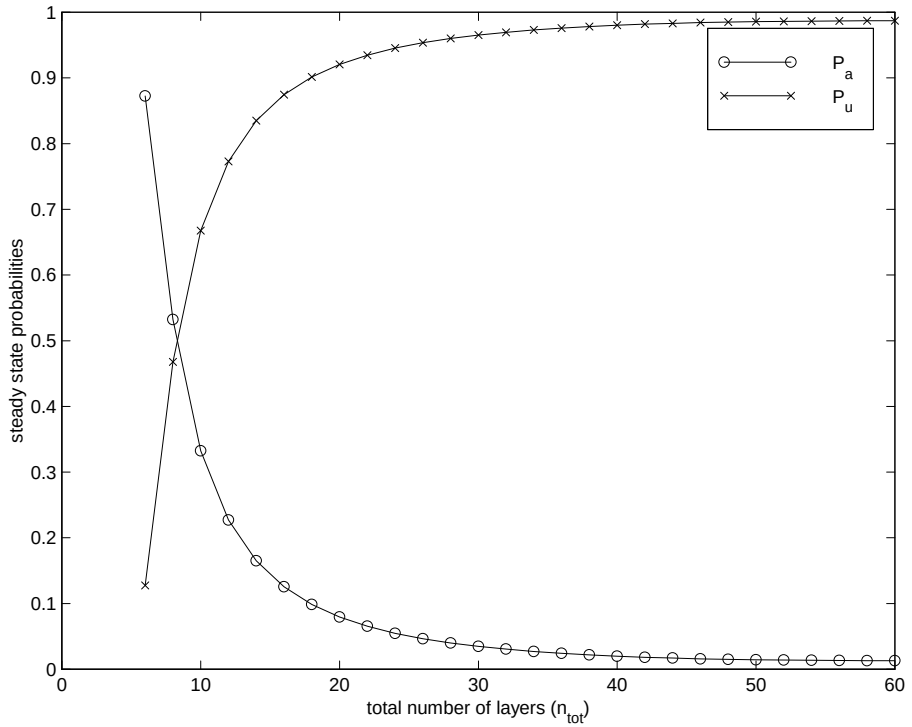

Figure 3.5: Steady state probabilities P_a and P_u for different n_{tot} values.

From the steady state probabilities of the M matrix, we calculate P_u and P_a using Eqs. (3.7) and (3.8), respectively. Fig. 3.5 shows P_u and P_a for different n_{tot} values. Here n_{av} is 5, therefore there are 12 available link states. When we increase n_{tot}, the number of feasible link states increases. Since the number of available link states is fixed (i.e., 12), the number of unavailable link states increases, therefore, P_a decreases while P_u increases as shown in Fig. 3.5.

Using the steady state probabilities of the M matrix will simplify $P_{ap}(K_u, l+1)$ and

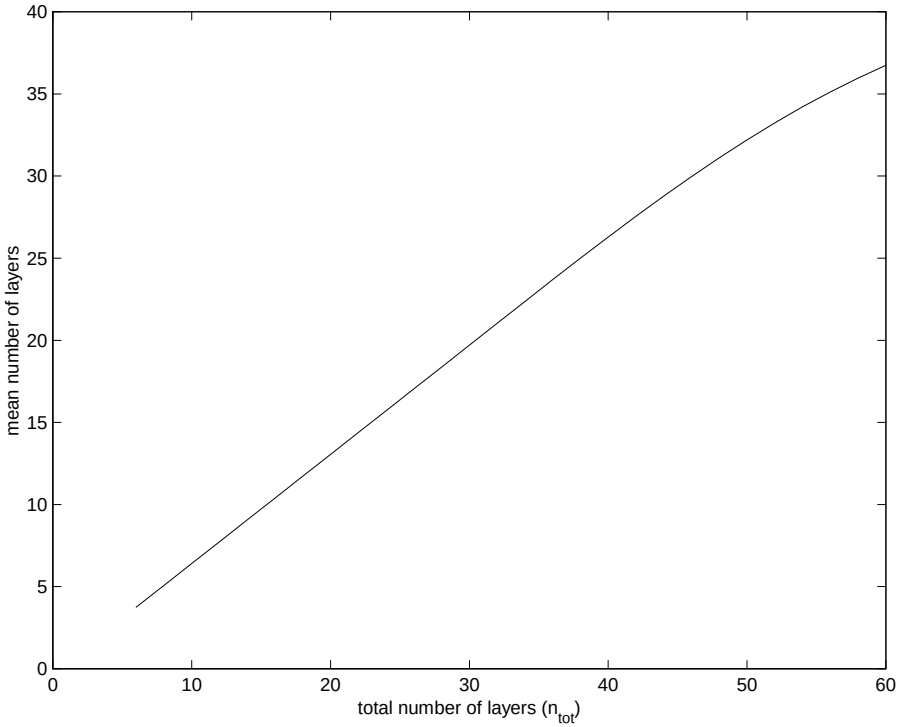

Figure 3.6: Mean number of layers in M matrix for different n_{tot} values.

$P_{dap}(k,l)$ as:

$$P_{dap}(k,l) = \binom{k}{l} \times P_u^l \times P_a^{k-l} \tag{3.9}$$

$$P_{ap}(K_u, l+1) = \binom{K_u}{l+1} \times P_a^{l+1} \times P_u^{K_u-l-1} \tag{3.10}$$

where P_a and P_u are the steady state values of the probabilities that a given link will be available and unavailable at the steady state, respectively.

Each link state belongs to a layer. Let $\overline{L}$ represent the mean number of layers for the link states of the M matrix. Suppose that the steady state probabilities of the M matrix is $\gamma_i = Pr(linkstate = i)$ for $i = 1, 2, \ldots, S_T$. $\overline{L}$ can be calculated using the steady state

probabilities of the M matrix:

$$\overline{L} = \sum_{i=1}^{S_T} l(i) \times \gamma_i \tag{3.11}$$

where the link state i belongs to the layer $l(i)$. The mean number of layers for different n_{tot} values are depicted in Fig. 3.6.

3.1.3 New Markov Chain Representing Node Degree

For a node z, with initially k available links, to have k+1 available links in the next time unit means that l of k available links disappear and l+1 of K_u unavailable links appear in the next time unit. Therefore, the probability for having k+1 links is:

$$P_{k,k+1} = \sum_{l=0}^{k} P_{dap}(k,l) \times P_{ap}(K_u, l+1) \tag{3.12}$$

Eq (3.12) represents only the probability for a transition from k-th state to $(k+1)$-th state. However, there may be a transition from k-th state to $(k+h)$-th state in one time unit for $0 \leq k \leq K$ and $0 \leq k+h \leq K$ (i.e., a transition from a given state to all states). The latter one is realizable since there are certain probabilities for multiple link arrivals and departures for a particular node.

Given k available links, we wish to calculate that there will be $k+h$ available links in the next time unit. This is possible only if l of k available links disappears and $l+h$ of K_u unavailable links appears in the next time unit for $0 \leq l \leq k$ and $0 \leq l+h \leq K_u$. Let $P_{ap}(K_u, l+h)$ denote the probability that $l+h$ of K_u unavailable links will appear and

$K_u - l - h$ of K_u unavailable links will remain unavailable in one time unit. $P_{dap}(k, l)$ is given in Eq. (4.8). Without using the stationary values of $P_{av}(i)$ and $P_{unav}(i)$ we have:

$$P_{ap}(K_u, l+h) = \sum_{j=s_a+1}^{S_T} f_s(j) \times \binom{K_u}{l+h} \times P_a^{l+h}(j) \times P_u^{K_u-l-h}(j) \tag{3.13}$$

where $f_s(j)$ is the probability density function of the unavailable link states for $s_a + 1 \leq j \leq S_T$.

If we use the steady state values of $P_a(i) = P_a$ and $P_u(i) = P_u$, Eq. (4.9) can be simplified as:

$$P_{ap}(K_u, l+h) = \binom{K_u}{l+h} \times P_a^{l+h} \times P_u^{K_u-l-h} \tag{3.14}$$

For a node z, with initially k available links, the probability for having $k+h$ links is:

$$P_{k,k+h} = \sum_{l=0}^{k} P_{dap}(k, l) \times P_{ap}(K_u, l+h) \tag{3.15}$$

where $0 \leq k \leq K$ and $0 \leq k+h \leq K$. By substituting Eqs. (4.8) and (4.12) in Eq. (4.16), we obtain:

$$P_{k,k+h} = \sum_{l=0}^{k} \binom{k}{l} \times \binom{K_u}{l+h} \times P_a^{k+h} \times P_u^{K_u-h} = \binom{K}{k+h} \times P_a^{k+h} \times P_u^{K_u-h} \tag{3.16}$$

where $0 \leq l + h \leq K_u$.

Eq. (4.17) represents a new finite state Markov chain obtained using the stationary distribution of $M_{i,j}$. Fig. 3.7 shows this new Markov chain, where a state k is the number of available links for a specific node. Since a node can have up to $K = N - 1$ available

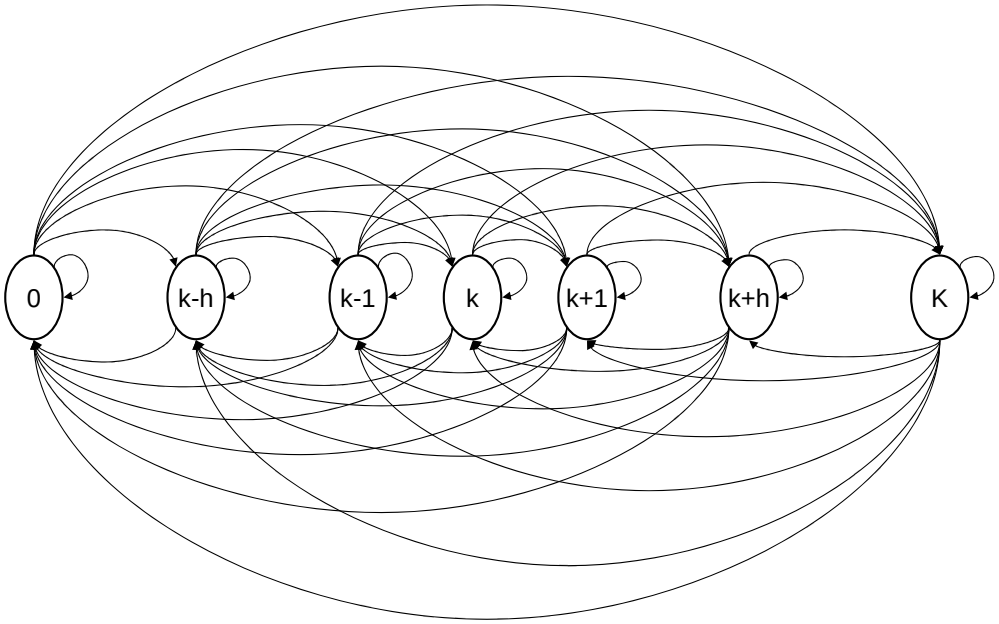

Figure 3.7: New Markov chain whose state represents node degree.

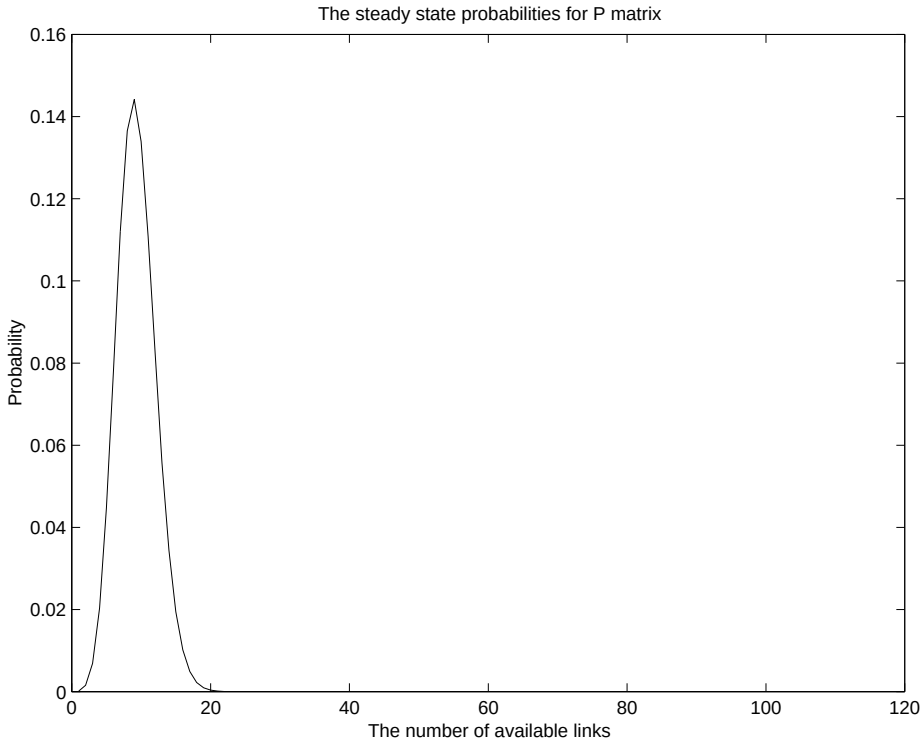

Figure 3.8: Steady state probabilities for P matrix.

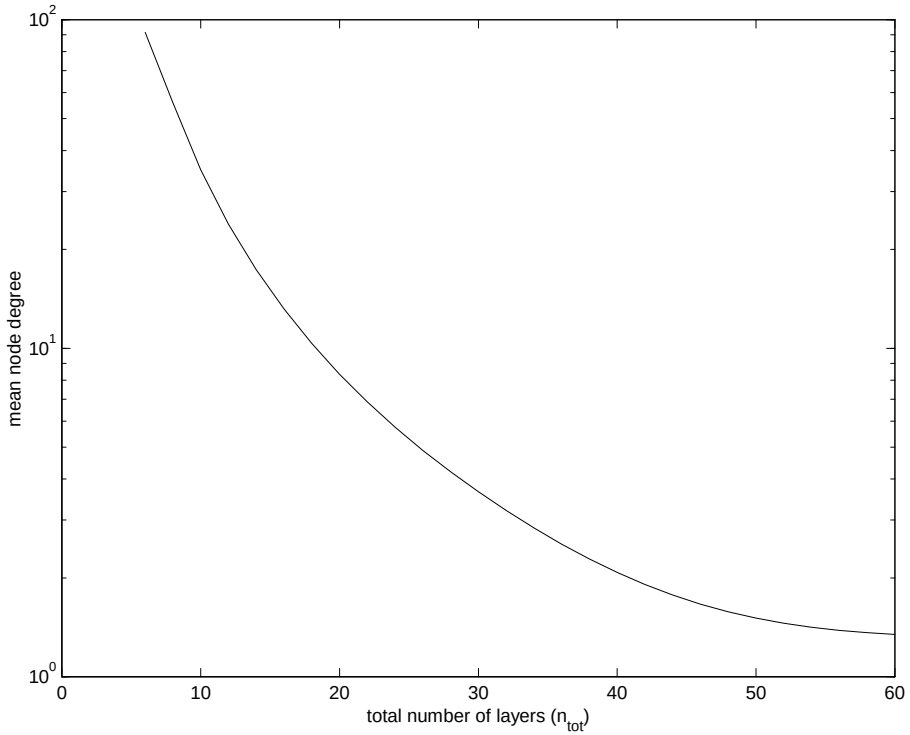

Figure 3.9: Mean node degree for different n_{tot} values.

links for a network with N mobile nodes, there are K states in the new chain and there is a transition probability from any state to any other state: for each $k = 0, 1, \ldots, K$ and $k + h = 0, 1, \ldots, K$, $P_{k,k+h}$ is the probability of moving from k-th state to $(k + h)$-th state. This new Markov chain, P, is ergodic (i.e., it is finite, connected, and aperiodic), and hence possesses a stationary distribution. Suppose that the stationary solution of this process is $\pi_k = Pr(state = k)$ for $k = 0, 1, \ldots, K$.

The stationary solution of P, π_k, is depicted in Fig. 3.8 for n_{tot}=20 and n_{av}=5. Let $\overline{N}$ denote the mean number of available links for a particular node. $\overline{N}$ can be calculated

using the stationary solution of P, π_k, as follows:

$$\overline{N} = \sum_{k=0}^{K} k \times \pi_k \tag{3.17}$$

Fig. 3.9 shows the mean number of available links of a particular node for different n_{tot} values while N=106 and n_{av}=5. When n_{tot} increases, the number of hexagonal cells in the network increases, and therefore the network size increases. Since N and n_{av} are fixed, the number of mean node degree for a particular node decreases. In the following section, we will use P with its stationary solution to model the number of link changes.

3.1.4 Modeling Degree Change

Now let us calculate the probability mass function for the degree change of a node in one time unit. The new markov chain describes multiple link arrivals and departures within one time unit, where the degree change is found by subtracting the number of new link arrivals from the number of new link departures. The degree change of a node can be positive (i.e., more arrivals than departures), negative (i.e., more departures than arrivals), or zero (i.e., equal number of arrivals and departures). For a random variable Z representing the degree change of a node in one time unit, the probability mass function of Z can be calculated as:

$$p_z(l) = Pr(Z = l) = \sum_{k=0}^{K} P(k, k+l) \times \pi_k \tag{3.18}$$

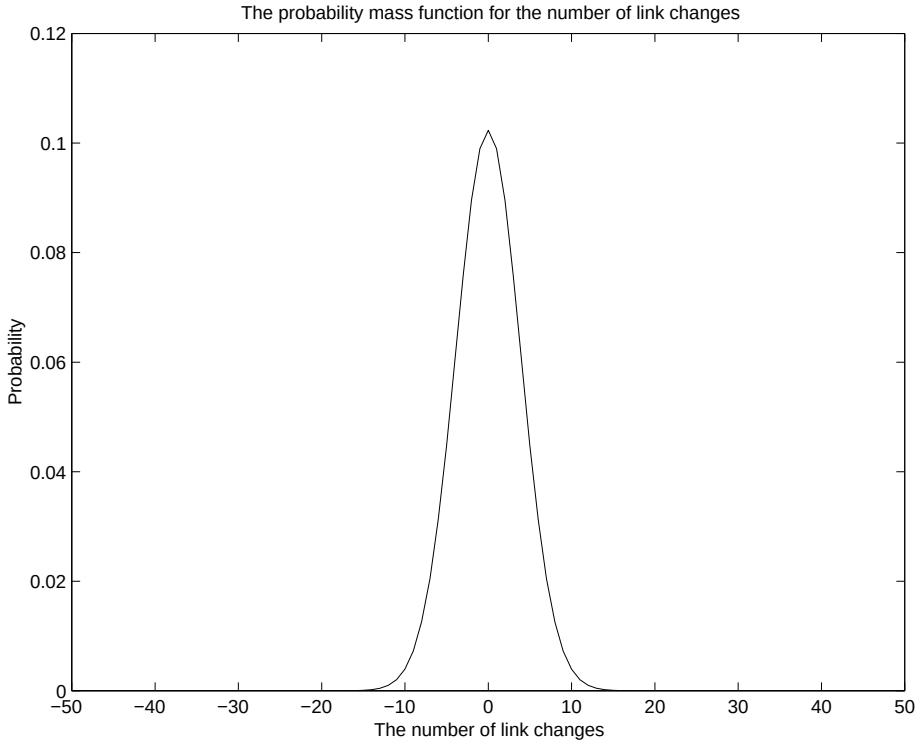

Figure 3.10: Probability mass function of degree changes.

where K is the maximum number of neighbors and l is an integer ($-K \leq l \leq K$). Since $p_z(l)$ is a probability mass function, the summation of $p_z(l)$ for all possible l values should add to 1:

$$\sum_{l=-\infty}^{\infty} p_z(l) = \sum_{l=-K}^{K} \sum_{k=0}^{K} P(k, k+l) \times \pi_k = \sum_{k=0}^{K} \sum_{l=-K}^{K} P(k, k+l) \times \pi_k = \sum_{k=0}^{K} \pi_k = 1 \quad (3.19)$$

Eq. (3.18) can be solved by off-the-shelf numerical analysis software (e.g., Maple[1]), obtaining that

$$\begin{aligned} p_z(l) = {} & P_a^l (1-P_a)^{2K-l} \Gamma(K+1) \left(\frac{P_a^2}{1-2P_a+P_a^2} \right)^{-1/2\,l} \left(-\frac{-1+2P_a}{1-2P_a+P_a^2} \right)^{K} \\ & \times \mathit{LegendreP}\left(K, -l, -\frac{2P_a^2+1-2P_a}{-1+2P_a}\right) \left(\Gamma(K-l+1)\right)^{-1} \end{aligned} \quad (3.20)$$

[1]Maple is a registered trademark of Maplesoft.

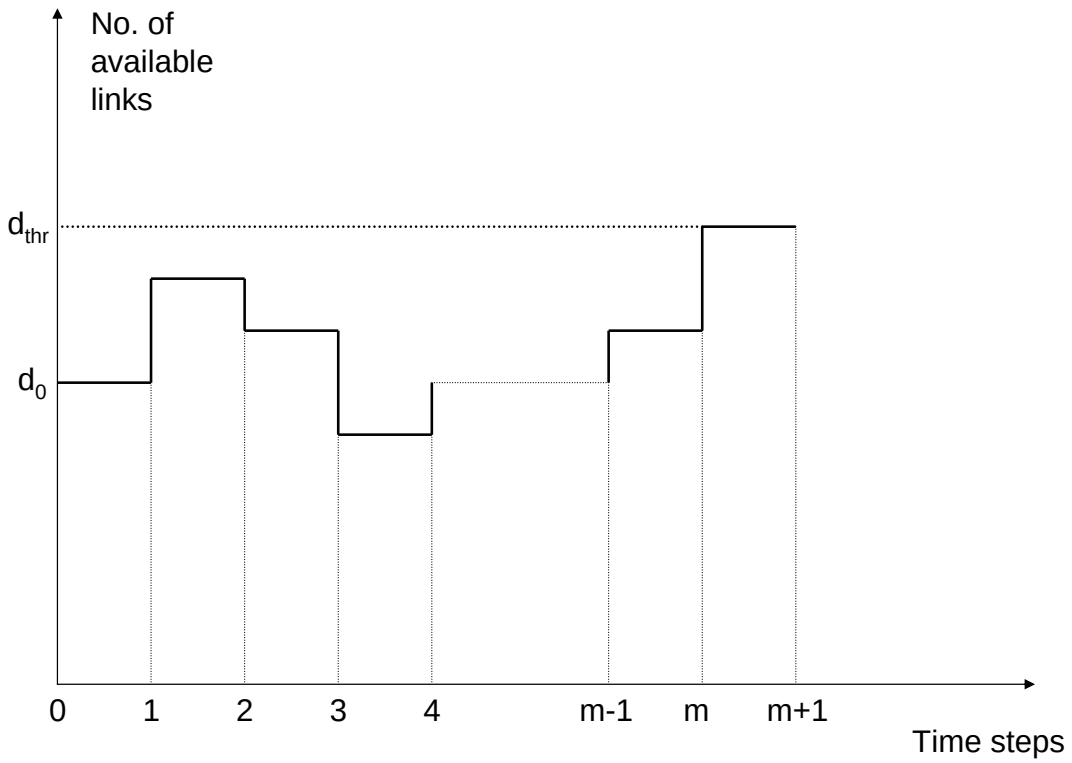

Figure 3.11: Number of available links in each step.

where *LegendreP* is the associated Legendre function of the first kind. This expression may be used to numerically evaluate $p_z(l)$.

Let $\overline{T}$ be the expected time that a non-backbone node, whose initial degree d_0 is less than a threshold d_{thr}, will become a backbone node when its degree meets or exceeds d_{thr}. Fig. 3.11 shows the number of available link changes at each step for a mobile node whose initial degree d_0 is less than the threshold degree d_{thr}. In each step, the number of available links changes according to the probability distribution of p_z given in Eq. 3.18 and this change is independent and identically distributed for each step. When and if the degree of this node becomes or exceeds d_{thr}, we classify this node as a backbone node.

Suppose the first time that the degree of a node will meet or exceed d_{thr} happens at the m-th step as shown in Fig. 3.11, which implies that until the m-th step, the degree of this node remains less than d_{thr}. Let a new set of random variables $Z_1, Z_2, \ldots, Z_m$ represent the number of link changes for the 1-st, 2-nd, ..., and m-th steps, respectively. Then the total number of link changes from the initial time to the m-th step will be the sum of the link changes occurred in each step. A new random variable S_m for the total number of link changes until the m-th step (i.e., the degree of a node will be equal to or greater than d_{thr}) is:

$$S_m = Z_1 + Z_2 + \ldots + Z_m \tag{3.21}$$

The difference between d_{thr} and the initial number of available links is:

$$thr_0 = d_{thr} - d_0 \tag{3.22}$$

A_m is the probability that the degree of a node will meet or exceed d_{thr} for the first time at the $m - th$ step:

$$A_m = \prod_{i=1}^{m} Pr(S_i \geq thr_0 | (\forall j : 0 \leq j < i) S_j < thr_0) \tag{3.23}$$

From Eq. 3.23, the expected time that a non-backbone node will become a backbone node can be calculated as:

$$\overline{T} = \sum_{m=0}^{\infty} m \times A_m \tag{3.24}$$

3.1.5 Modeling Number of Link Arrivals and Departures

Let us now calculate the probability mass functions for the number of link arrivals and departures in one time unit, since the new process indicates that there might be multiple link arrivals and departures in one time unit. We will find a probability distribution for the number of link arrivals in one time unit, and then a probability distribution for the number of link departures in one time unit. Let two new random variables X and Y represent the number of link arrivals and the number of link departures for one time unit, respectively. Then, the probability mass functions of X and Y are p_x and p_y respectively, can be calculated as follow:

$$p_x(l) = Pr(X = l) = \sum_{k=0}^{K} P_{dap}(k, l) \times \pi_k \tag{3.25}$$

$$p_y(l) = Pr(Y = l) = \sum_{k=0}^{K} P_{ap}(K_u, l) \times \pi_k \tag{3.26}$$

where $l = 0, 1, \ldots, K$.

Existing virtual-backbone formation algorithms use either node degree [59] or both node degree and normalized link failure frequency ($nlff$) [61] to resolve conflicts between the nodes competing to join the backbone. So far, we presented an analytic model for calculating the average time for a node degree to either drop below or exceed a given threshold. Now we will extend our analytic model to find the average time it takes for not only the node degree but also the number of link changes of the node (i.e., instances of link creation

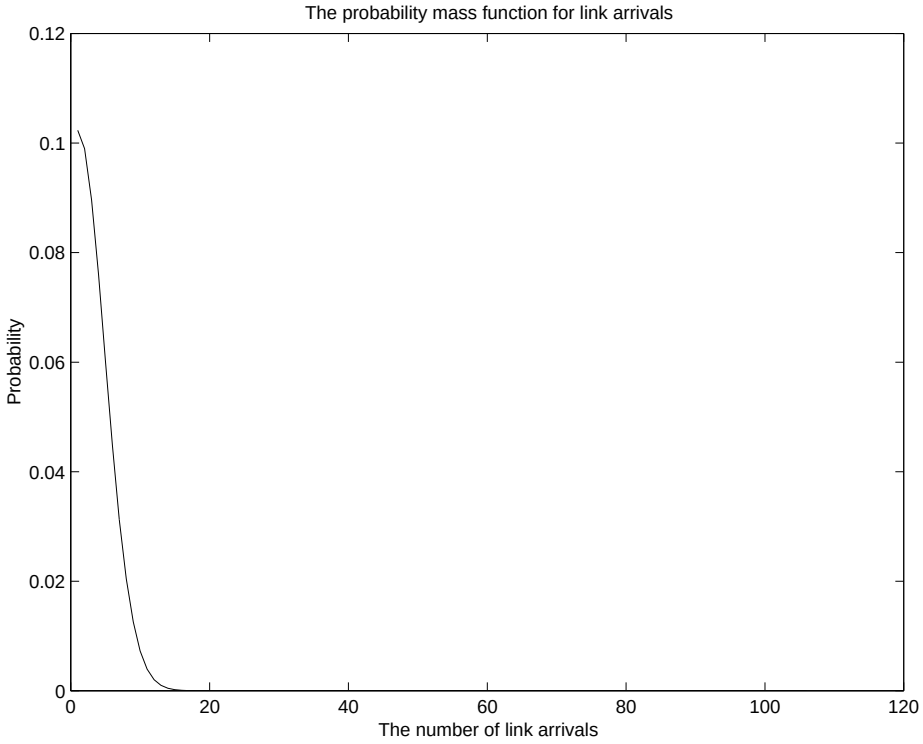

Figure 3.12: Probability distribution for new link arrivals.

and failure) to either drop below or exceed a given threshold. In this extended model, $nlff$ represents the stability of a node's links.

We will use $P_{dap}(k, l)$ and $P_{ap}(K_u, l + h)$, given in Eqs. (4.8) and (4.12) respectively, as a starting point for our extended analytic model, where we consider a particular node z of a network with N nodes which has initially k available and $K_u = K - k$ unavailable links. $P_{dap}(k, l)$ and $P_{ap}(K_u, l + h)$ denote the probability of l link departures out of k available links and probability of $l + h$ link arrivals out of K_u unavailable links in one time unit, respectively.

Given k available links and y link departures initially, the probability that there will be l

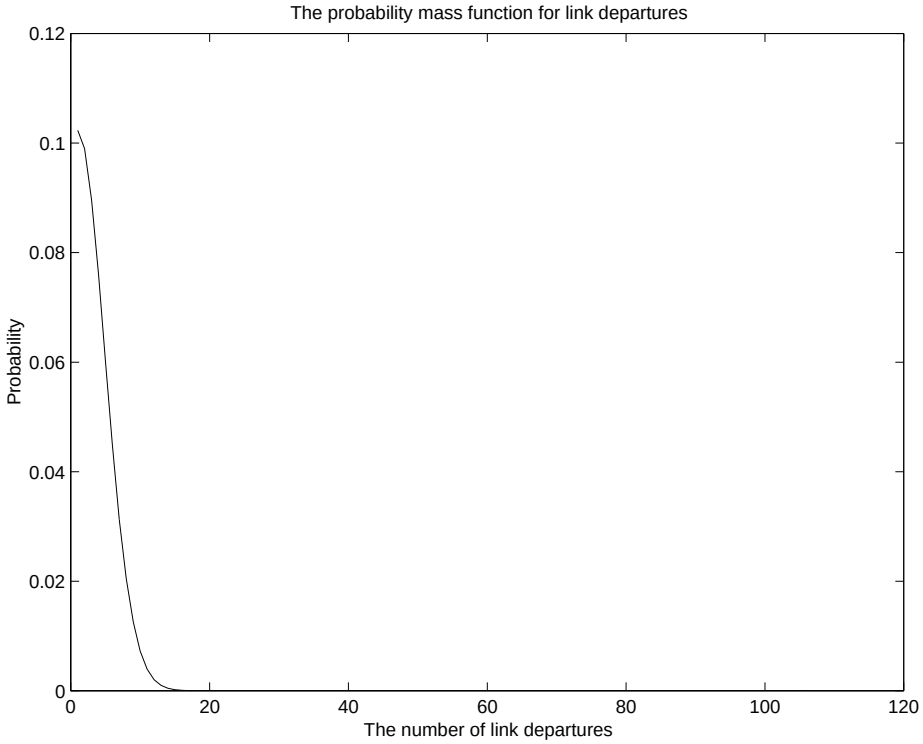

Figure 3.13: Probability distribution for new link departures.

new link departures and $l + h$ new link arrivals, hence $k + h$ available links, in the next time unit is:

$$P2_{(k,y),(k+h,l)} = P_{dap}(k, l) \times P_{ap}(K_u, l + h) \tag{3.27}$$

where $0 \leq k \leq K$, $0 \leq k+h \leq K$, $0 \leq y \leq K$ and $0 \leq l \leq K$. By substituting Eqs. (4.8) and (4.12) in Eq. (4.13), we obtain:

$$P2_{(k,y),(k+h,l)} = \binom{k}{l} \times \binom{K_u}{l+h} \times P_a^{k+h} \times P_u^{K_u-h} \tag{3.28}$$

Eq. (4.14) represents a new two-dimensional finite state Markov chain, $P2$, with state (k, y), where k and y are the number of available links and the number of new link departures for a specific node, respectively.

In our DSRP architecture, $nlff$ is used as the first selection rule for the VB formation algorithm as presented in Ref. [61]. The $nlff$ metric is defined as the number of new link departures within a time-window, but normalized by the number of total links at the end of the window. Then, a state $(k+h, l)$ will represent the node degree, (i.e., $k+h$), and the $nlff$ value, (i.e., $l/(k+h)$). Here, we assume that the time-window for calculating the $nlff$ value is equal to one unit time used in our analytic model.

In Section 4.1.6, we provided the formulation for the expected time that the degree of a node will meet or exceed d_{thr} for the first time. This expected time determines the mean time that a non-backbone node will become a backbone node, where only the node degree is used to form the VB. Using the new two-dimensional Markov chain with the transition matrix $P2$, we can formulate the expected time that a non-backbone node will become a backbone node, where both *nlff* and the node degree are used as the selection rules for the VB formation algorithm. We will present the first passage time analysis [12] in the following section, which can be used to find the desired expected time from the new two-dimensional Markov chain with the transition matrix $P2$.

3.2 Expected Time Calculation Using First Passage Times

The first passage time analysis [12] allows us to find the number of transitions made by a process in moving from one state to another for the first time. The expected time $\overline{T}$ that a non-backbone node, with d_0 initial links ($d_0 < d_{thr}$), will become a backbone node is defined by Eq. (3.24). This equation is hard to solve directly; therefore, we make several modifications to the Markov chain with the transition matrix P in order to use the first passage time analysis to obtain $\overline{T}$.

3.2.1 First Passage Time Analysis for nonVB to VB

The expected time $\overline{T}$ that a nonVB node, with d_0 initial links ($d_0 < d_{thr}$), will become a backbone node can be obtained as the solution to Eq. 3.24 using the first passage time analysis. Given that a node has a smaller number of available links than the threshold (i.e., state i where $i = d_0 < d_{thr}$), we want to find the expected first time that the number of available links will be equal to the threshold (i.e., state j where $j = d_{thr}$) or greater than the threshold (i.e., state j where $j = d_{thr} + 1, d_{thr} + 2, \ldots, K$). To address this question, we modified the markov chain P by combining the states, which represent the number of available links equal to or greater than the threshold, into a single state (d_{thr}). The entries of the new transition matrix Q are shown in Table 3.2. The size of Q is

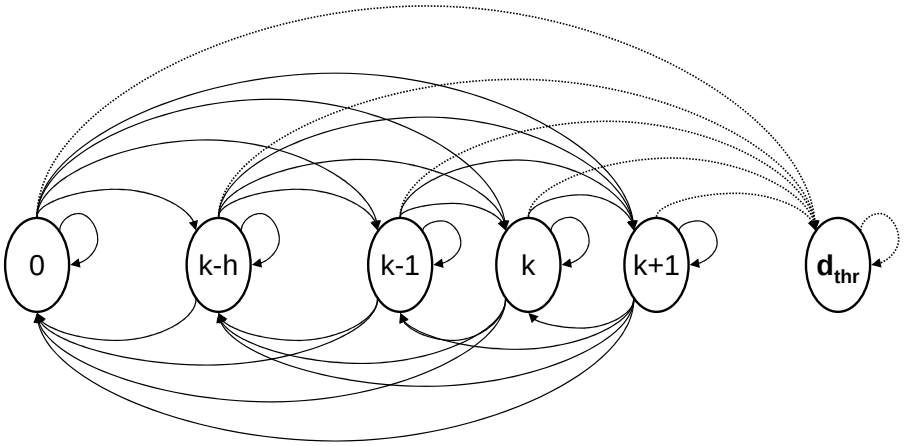

Figure 3.14: New markov chain for first passage time analysis of nonVB-to-VB case.

Table 3.2: New transition matrix Q for first passage time analysis of nonVB-to-VB case

$Q(i,j) = P(i,j)$ if $i < d_{thr}, j < d_{thr}$
$Q(i,d_{thr}) = \sum_{j=d_{thr}}^{K} P(i,j)$ if $i < d_{thr}, j \geq d_{thr}$
$Q(d_{thr},j) = 0$ if $i \geq d_{thr}, j < d_{thr}$
$Q(d_{thr},d_{thr}) = 1$

$(d_{thr}+1) \times (d_{thr}+1)$ and the corresponding markov chain has $d_{thr}+1$ different states.

Let us define a new set of random variables $X_0, X_1, \ldots, X_m$ that represent the number of available links at the initial, 1-st, ..., and m-th time units, respectively. The number of transitions made by the process in moving from state i to j for the first time is:

$$T_{ij} = min\{m \geq 1 : X_m = j \mid X_0 = i\} \tag{3.29}$$

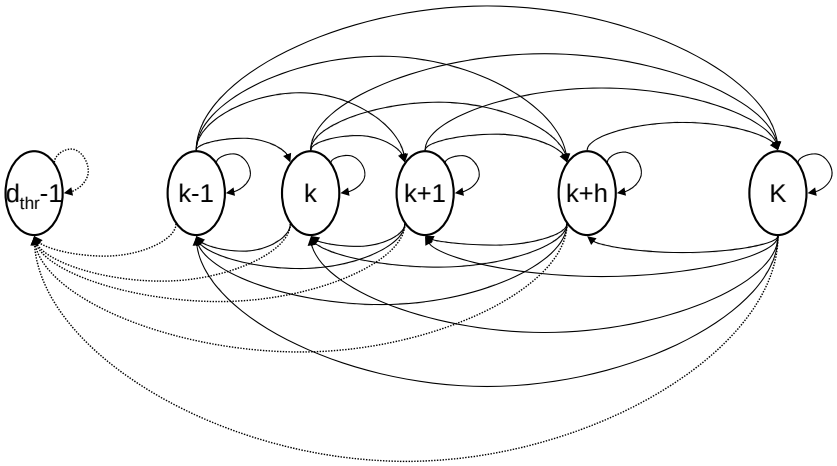

Figure 3.15: New markov chain for first passage time analysis of VB-to-nonVB case.

Let $f_{ij}^{(m)}$ denote the probability that T_{ij} is m. Then,

$$f_{ij}^{(1)} = q_{ij}^{(1)} = q_{ij}, \qquad f_{ij}^{(m)} = \sum_{k \neq j} q_{ik} \times f_{kj}^{(m-1)} \tag{3.30}$$

$$f_{ij}^{(m)} \geq 0, \qquad \sum_{m=1}^{\infty} f_{ij}^{(m)} \leq 1 \tag{3.31}$$

where q_{ij} is the element of the i-th row and j-th column of Q matrix, which denotes the state transition probability of moving from state i to j in one time unit. The expected first passage time can then be calculated as:

$$\mu_{ij} = \begin{cases} \infty & \text{if } \sum_{m=1}^{\infty} f_{ij}^{(m)} < 1 \\ \sum_{m=1}^{\infty} m \times f_{ij}^{(m)} & \text{if } \sum_{m=1}^{\infty} f_{ij}^{(m)} = 1 \end{cases} \tag{3.32}$$

Table 3.3: New transition matrix R for first passage time analysis of VB-to-nonVB case

$R(i,j) = P(i,j)$ if $i \geq d_{thr}, j \geq d_{thr}$
$R(i, d_{thr}-1) = \sum_{j=0}^{d_{thr}-1} P(i,j)$ if $i \geq d_{thr}, j < d_{thr}$
$R(d_{thr}-1, j) = 0$ if $i < d_{thr}, j \geq d_{thr}$
$R(d_{thr}-1, d_{thr}-1) = 1$

3.2.2 First Passage Time Analysis for VB to nonVB

All the steps used in the nonVB-to-VB case will also be valid here. The only difference is that P matrix will be modified for the VB-to-nonVB case, resulting in a different transition matrix called R. Given a VB node in state i where $i = d_0 \geq d_{thr}$, we wish to find the expected first time the node will be a nonVB node by moving to state j where $j < d_{thr}$. The original chain of P can be modified by combining the states, which represent the number of available links smaller than the threshold, into a single state $(d_{thr} - 1)$. The entries of the new transition matrix R has the size of $(K - d_{thr} + 2) \times (K - d_{thr} + 2)$ and the corresponding markov chain has $K - d_{thr} + 2$ different states (Table 3.3).

3.3 Numerical Results

In this section, we provide numerical results for the mean number of active neighbors $(\overline{N}(n_{tot}, n_{av}))$, the expected time that a nonVB node becomes a VB $(\overline{T}_{VB}(n_{tot}, n_{av}))$ and

the expected time that a VB node becomes a nonVB ($\overline{T}_{nonVB}(n_{tot}, n_{av})$). The expected times and the mean number of neighbors depend on n_{tot} and n_{av} for a fixed N. In our calculations, the number of mobile nodes is 106 which implies that there are 105 possible bi-directional links for a single node.

One time unit in this study is equal to the time for a mobile node to move from its current hexagonal cell to the next. This time unit depends on the size of the cell and the speed of the mobile node. The calculation of the time unit for different node speed and cell size distributions is beyond the scope of this study and will be handled as an extension of this work in the future.

3.3.1 Expected First Times from nonVB to VB

Table 3.4 shows that the expected times from any initial state to a certain threshold state are independent of the initial state (i.e., the initial number of the available links). This result is intuitive since a given link without depending on its initial link state will be available with the probability of P_a and unavailable with the probability of P_u in the next time unit as given in Eq. 4.17.

As an example, consider a mobile network with 16 nodes (i.e., each node can have up to 15 links). In this network, for a node with 3 active links to have 8 active links (i.e., moving from state 3 to 8) in the next time unit, there are four possibilities as shown in

Table 3.4: Expected times for the nonVB-to-VB case

d_0	3	3	3	3	3	3	4	5	5	1	1	9
d_{thr}	5	6	7	8	9	10	6	7	10	10	15	10
$\overline{T}_{VB}(20,5)$	1.08	1.17	1.35	1.66	2.19	3.10	1.17	1.35	3.10	3.10	53.00	3.10

Table 3.5. For another node with 7 active links (i.e., in state 7) to increase its active links to 8 (i.e., moving from state 7 to 8), there are 8 possibilities as tabulated in Table 3.6. The total number of combinations for the first and the second cases are identical and equal to 6,435. Therefore,

$$P_{3,8} = P_{7,8} = 6,435 \times P_a^8 \times P_u^7$$

This result can be generalized using Eq. 4.17, which indicates that the probability of having a certain number of available links in the next time unit is independent of the number of available links in the current time. The probability of having 8 number of available links in the next time unit given that there are k number of available links currently:

$$P_{k,8} = \binom{15}{8} \times P_a^8 \times P_u^7 == 6,435 \times P_a^8 \times P_u^7$$

where $0 \leq k \leq 15$.

Therefore, the state transition matrix Q has the identical elements at each column, and the probability mass functions of the first passage times from all states to this threshold

Table 3.5: Possible cases of having 8 neighbors from initially 3 neighbors

Number of Links Failures	Number of Link Arrivals	Total Number of Combinations
3	8	$\binom{3}{3} \times \binom{12}{8} = 495$
2	7	$\binom{3}{2} \times \binom{12}{7} = 3,168$
1	6	$\binom{3}{1} \times \binom{12}{6} = 2,772$
0	5	$\binom{3}{0} \times \binom{12}{5} = 792$

Table 3.6: Possible cases of having 8 neighbors from initially 7 neighbors

Number of Link Failures	Number of Links Arrivals	Total Number of Combinations
7	8	$\binom{7}{7} \times \binom{8}{8} = 1$
6	7	$\binom{7}{6} \times \binom{8}{7} = 56$
5	6	$\binom{7}{5} \times \binom{8}{6} = 588$
4	5	$\binom{7}{4} \times \binom{8}{5} = 1,960$
3	4	$\binom{7}{3} \times \binom{8}{4} = 2,450$
2	3	$\binom{7}{2} \times \binom{8}{3} = 1,176$
1	2	$\binom{7}{1} \times \binom{8}{2} = 196$
0	1	$\binom{7}{0} \times \binom{8}{1} = 8$

state are the same for a given threshold. Therefore, we can set the initial number of available links to 0 and obtain the expected times for different threshold values.

Instead of using the steady state probabilities of the first markov chain (M) for this analysis, if we used different M for each step (each time unit), the results for T (the expected time) would have depended on the initial state. However, we expect that, when averaged over different starting points, the results will approach the steady state analysis that we conducted in this study. Let T_i be the expected value for reaching d_{thr} starting from the initial state i. Then the exact formulation for n different initial states would yield the expected value of $T_{exact} = \frac{1}{n}\sum_{i=1}^{n} T_i$. If the initial values of the number of links are considered, the formulation of the problem becomes prohibitively more complex. In addition, it is not clear that such an exact formulation will give different results than the approximation. Using the steady state probabilities produces the same T for every initial state. We conjecture that T_{exact} is very close to the one formulated in this chapter.

Table 3.7 shows the network parameters used in our numerical results, where n_{tot} determines the geographic size of the network and n_{av} represents the communication range between two nodes, both in terms of layers. For all numerical results, except for the Medium Density Network II, N and n_{av} are fixed to 106 and 5, respectively, and n_{tot} is varied to represent mobile networks with different densities. We define *sparsest*, *sparse*, *medium density I*, *dense*, and *densest* networks with n_{tot} values of 40, 30, 20, 15, and 10, respectively. In medium density network II, N, n_{tot}, and n_{av} are 106, 40 and 10,

Table 3.7: Network parameters

Network types	(n_{tot}, n_{av})	P_a	P_u	$\overline{N}$
Densest	(10,5)	0.3324	0.6676	34.90
Dense	(15,5)	0.1432	0.8568	15.03
Medium Density I	(20,5)	0.0794	0.9206	8.33
Medium Density II	(40,10)	0.0705	0.9295	7.54
Sparse	(30,5)	0.0348	0.9652	3.65
Sparsest	(40,5)	0.0198	0.9802	2.07

respectively. Our aim is to observe the effects of doubling n_{tot} and n_{av} simultaneously by comparing the results with the medium density network I.

In Table 3.7, P_u and P_a are calculated numerically using Eqs. 3.7 and 3.8 and denote the probabilities that the wireless link will be unavailable and available in the next time unit, respectively. $\overline{N}$ is calculated numerically using Eq. 4.18 and denotes the mean number of neighbors for a given network. The numerical results show that P_a (and hence $\overline{N}$) has the highest value for the densest network, decreases as the network density decreases and reaches the lowest value for the sparsest network.

As discussed in detail in the following parts of this section, the numerical results show that, for a given threshold d_{thr}, it takes more steps to have d_{thr} active links in the sparsest network than in the densest network. For example, it is reasonable to assume that d_{thr} should be around the mean value of the number of active links. The numerical results

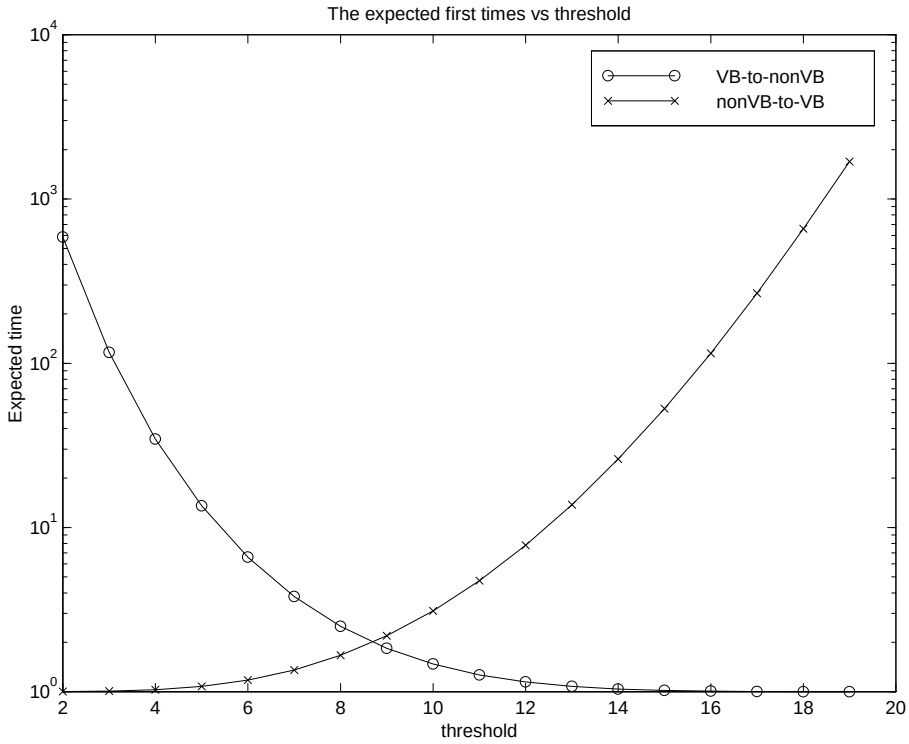

Figure 3.16: Expected time vs threshold to become VB and nonVB in medium density network (n_{tot}=20, n_{av}=5).

support this expectation that it takes almost the same number of steps to reach the mean number of active links for each case.

3.3.2 Medium Density Network I ($n_{tot} = 20$ and $n_{av} = 5$)

The numerical results for the expected times in Table 3.8 are presented using the logarithmic scale in y-axis in Fig. 3.16. When $d_{thr} = 1$, there are only two states, namely states 0 and 1, that represent the number of available links for a particular node. Since the initial state is zero, at least one unit time is needed for going from state 0 to 1 with

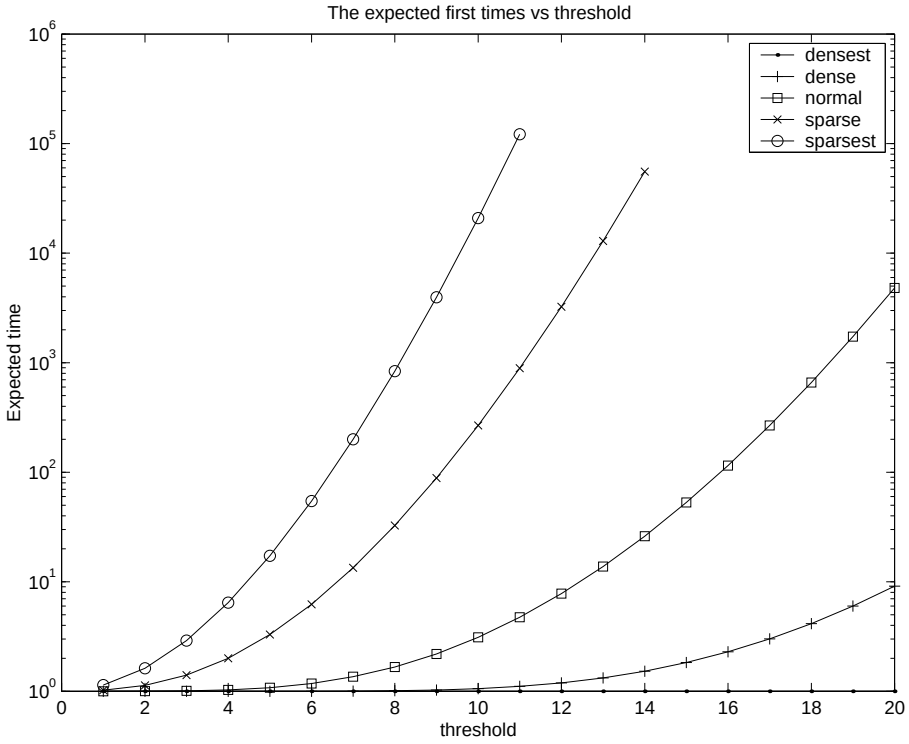

Figure 3.17: Expected times vs threshold to become a VB in different density networks.

expected time of 1.0002 (1+0.0002) where 0.0002 is due to the fact that there is a certain probability of staying at state 0 in the next time unit. When $d_{thr} = 2$, there are three states: 0, 1, and 2. The probabilities for staying at states 0 or 1 before moving into state 2 are higher in this case, and, therefore, the expected time is higher compared to the case of $d_{thr} = 1$. It is intuitive that the expected first times to reach or exceed d_{thr} number of available links increase when d_{thr} increases as shown in Table 3.8 and Fig. 3.16.

Using Eq. 4.18, $\overline{N}(20, 5)$ is calculated as 8.3334. We observe that the expected first times are very small when $d_{thr} \approx 10$. However, when $d_{thr} > 10$, the expected first times increase exponentially as depicted in Fig. 3.16.

3.3.3 Sparsest Network ($n_{tot} = 40$ and $n_{av} = 5$)

In this case, the total number of layers is 40 and the number of layers representing the available link states is 5. The probability of being available in the next time unit at the steady state P_a will be much lower than those for the medium and densest networks: from Eq. 4.18, $\overline{N}(40, 5)$ is calculated as 2.0770 whereas $\overline{N}(20, 5)$ and $\overline{N}(10, 5)$ are calculated as 8.3334 and 34.9017, respectively. Therefore, we expect that $\overline{T}_{VB}(40, 5)$ will be greater than both $\overline{T}_{VB}(20, 5)$ and $\overline{T}_{VB}(10, 5)$ for a certain threshold degree (Table 3.9). Moreover, we could not obtain the expected first times for the threshold values greater than 11. Since $\overline{N}(40, 5) \approx 2$, it takes much longer time to visit the states greater than 11.

3.3.4 Densest Network ($n_{tot} = 10$ and $n_{av} = 5$)

The probability of being available in the next time unit at the steady state P_a will be the highest in densest networks. Using Eq. 4.18, we calculate $\overline{N}(10, 5) = 34.9017$, $\overline{N}(20, 5) = 8.3334$ and $\overline{N}(40, 5) = 2.0770$, which imply that $\overline{T}_{VB}(10, 5)$ will be much lower than $\overline{T}_{VB}(20, 5)$ and $\overline{T}_{VB}(40, 5)$ as shown in the Table 3.10. Moreover, the expected values are almost 1 for $d_{thr} \ll \overline{N}(10, 5) = 34.9017$. It only takes one time unit to move from the initial state to the next state. The probability of having $d_{thr} < \overline{N}(10, 5)$ available links is very high for the densest network with the highest P_a value and 106 nodes. $\overline{N}(10, 5) \approx 35$ indicating that it will take much less time to visit the states less than 35.

Table 3.8: Expected time that a nonVB node becomes a VB for the medium density network for different d_{thr} values (n_{tot}=20, n_{av}=5, $\overline{N}(20,5) = 8.3334$)

d_{thr}	1	2	3	4	5	6	7
$\overline{T}_{VB}(20,5)$	1.0002	1.0017	1.0087	1.0298	1.0795	1.1787	1.3579
d_{thr}	8	9	10	11	12	13	14
$\overline{T}_{VB}(20,5)$	1.6669	2.1941	3.1074	4.7400	7.7858	13.753	26.083
d_{thr}	15	16	17	18	19	20	
$\overline{T}_{VB}(20,5)$	53.000	115.14	266.90	658.71	1727.5	4805.7	

Table 3.9: Expected time that a nonVB node becomes a VB for the sparsest network for different d_{thr} values (n_{tot}=40, n_{av}=5, $\overline{N}(40,5) = 2.0770$)

d_{thr}	1	2	3	4	5	6
$\overline{T}_{VB}(40,5)$	1.1399	1.6201	2.9040	6.4397	17.242	54.530
d_{thr}	7	8	9	10	11	
$\overline{T}_{VB}(40,5)$	199.99	837.83	3960.1	20890	121872	

Table 3.10: Expected time that a nonVB node becomes a VB for the densest network for different d_{thr} values (n_{tot}=10, n_{av}=5, $\overline{N}(10,5) = 34.9017$)

d_{thr}	1	10	15	20	25	30	31
$\overline{T}_{VB}(10,5)$	1.0000	1.0000	1.0000	1.0004	1.0138	1.1505	1.2217
d_{thr}	32	33	34	35	36	37	38
$\overline{T}_{VB}(10,5)$	1.3200	1.4548	1.6392	1.8920	2.2412	2.7287	3.4193
d_{thr}	39	40	41	42	43	44	
$\overline{T}_{VB}(10,5)$	4.4149	5.8791	8.0807	11.472	18.830	25.528	

Table 3.11: Expected time that a nonVB node becomes a VB for the densest network for different d_{thr} values (n_{tot}=40, n_{av}=10, $\overline{N}(40,10) = 7.5403$)

d_{thr}	1	2	3	4	5	6	7
$\overline{T}_{VB}(40,10)$	1.0004	1.0037	1.0170	1.0542	1.1362	1.2934	1.5744
d_{thr}	8	9	10	11	12	13	14
$\overline{T}_{VB}(40,10)$	2.0653	2.9316	4.5086	7.5087	13.5195	26.2599	54.8894
d_{thr}	15	16	17	18	19		
$\overline{T}_{VB}(40,10)$	123.1535	295.8567	759.1659	2074.7	6037.8		

Table 3.12: Expected time that a VB node becomes a nonVB for the medium density network for different d_{thr} values (n_{tot}=20, n_{av}=5, $\overline{N}(20,5) = 8.3334$)

d_{thr}	2	3	4	5	6	7	8
$\overline{T}_{nonVB}(20,5)$	586.95	116.53	34.557	13.571	6.5957	3.7930	2.4995
d_{thr}	9	10	11	12	13	14	
$\overline{T}_{nonVB}(20,5)$	1.8374	1.4745	1.2674	1.1474	1.0784	1.0399	
d_{thr}	15	16	17	18	19	20	
$\overline{T}_{nonVB}(20,5)$	1.0192	1.0088	1.0038	1.0015	1.0006	1.0002	

3.3.5 Medium Density Network II ($n_{tot} = 40$ and $n_{av} = 10$)

For this network, Eq. 4.18 yields $\overline{N}(40,10) = 7.5403$. When we doubled the numbers of both total and available layers, the numbers of available and unavailable link states will not exactly double: the latter will increase more than the former. Therefore, although we doubled both the total and available layers, P_a and P_u values will be different from the Medium Density Network I due to the structure of the link states in the M matrix, and therefore $\overline{N}(40,10)$ will be different from $\overline{N}(20,5)$.

Doubling both the total and available layers means that the same geographic area as in the Medium Density Network I, but smaller hexagonal cells. The radius of a hexagonal cell will be the half of the Medium Density Network I. Since one time unit in this study is the time a mobile node moves from its current hexagonal cell to the next, for

the same node speed, one unit time in the Medium Density Network II will then be approximately half of the Medium Density Network I. From Tables 3.11 and 3.8, the values for $\overline{T}_{VB}(40,10)$ are approximately twice of $\overline{T}_{VB}(20,5)$. But, since there are more than twice cells in $\overline{T}_{VB}(40,10)$, one unit time in $\overline{T}_{VB}(40,10)$ is approximately half of the unit time in $\overline{T}_{VB}(20,5)$ and hence $\overline{T}_{VB}(40,10)$ and $\overline{T}_{VB}(20,5)$ correspond to almost the same expected times.

Also, for small threshold values, the comparison must be based on the fractional values of the expected times. For example, for d_{thr}=1, $\overline{T}_{VB}(20,5)$=1.0002 (3.8) and $\overline{T}_{VB}(40,10)$=1.0004 (3.11) which means that the expected time for $\overline{T}_{VB}(40,10)$ is twice of $\overline{T}_{VB}(20,5)$ since we consider only the fractional values (1 time unit is deterministically included moving from one state to another).

Another interpretation of doubling n_{tot} and n_{av} values of the Medium Density Network I is that although the hexagonal cell size remains the same, the geographic area is larger here (i.e., less dense network compared to the Medium Density Network I). Therefore, it is intuitive that the expected first times will be higher in this case as can be seen from the Tables 3.8 and 3.11. We assume time units are same for both cases since the hexagonal cell sizes and node speeds are assumed to be same.

3.3.6 Different Number of Nodes on Same Hexagonal Area ($n_{tot} = 20$ and $n_{av} = 5$)

We should note that different network densities can be obtained by varying a single parameter among N, n_{tot}, and n_{av}. Here, n_{tot} and n_{av} are fixed to 20 and 5, respectively while the number of nodes on this hexagonal area is varied as 50, 106 and 150 to obtain the same geographic area with different node densities. For three different network densities, the network having 50 mobile nodes has the sparsest density and its mean number of nodes is calculated as 3.96 from Eq. 4.18, while the network having 150 mobile nodes is the densest with a mean node degree of 11.89. The mean node degree is calculated as 8.33 for the network with 106 mobile nodes. Fig. 3.18 shows the expected first times calculated numerically for these three different network densities. The numerical results verify the same behavior as with the previous network densities that, for a given threshold d_{thr}, it takes longer to have d_{thr} active links in the sparsest network than in the densest network.

3.3.7 Expected First Times from VB to nonVB

For $n_{tot} = 20$ and $n_{av} = 5$, Table 3.12 shows the expected first passage times for moving from a VB to nonVB state. The expected first time for the number of available links to drop below the threshold increases when the threshold decreases (Fig. 3.16).

An interesting observation is that the expected first times for both nonVB-to-VB and VB-

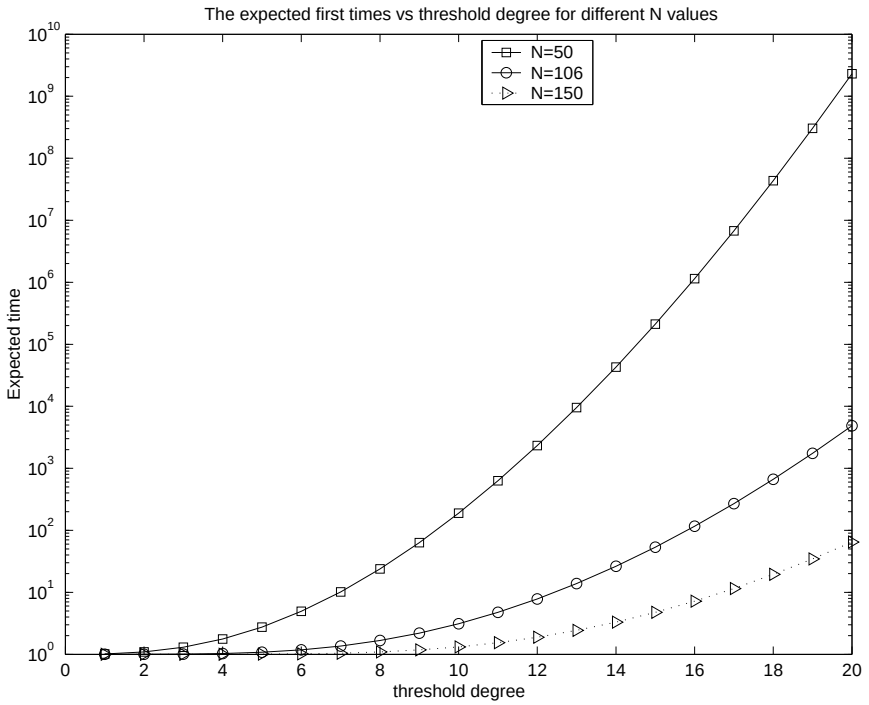

Figure 3.18: Expected first times obtained using different number of nodes in the same hexagonal area.

to-nonVB cases are close when the threshold is around $\overline{N}(20, 5)$=8.3334 (Fig. 3.16). For example, for the VB-to-nonVB case, when the threshold is set to 9, in Table 3.12, we found the expected first times to reach state 8 from the states higher than 8 as 1.8374. However, for the nonVB-to-VB case, when we set the threshold to 8, we found the expected first times to reach state 8 from the states lower than 8 as 1.6669. Therefore, the expected first times to reach a certain threshold of active links from both lower and higher number of active links are approximately the same when the threshold is set to the mean number of active links.

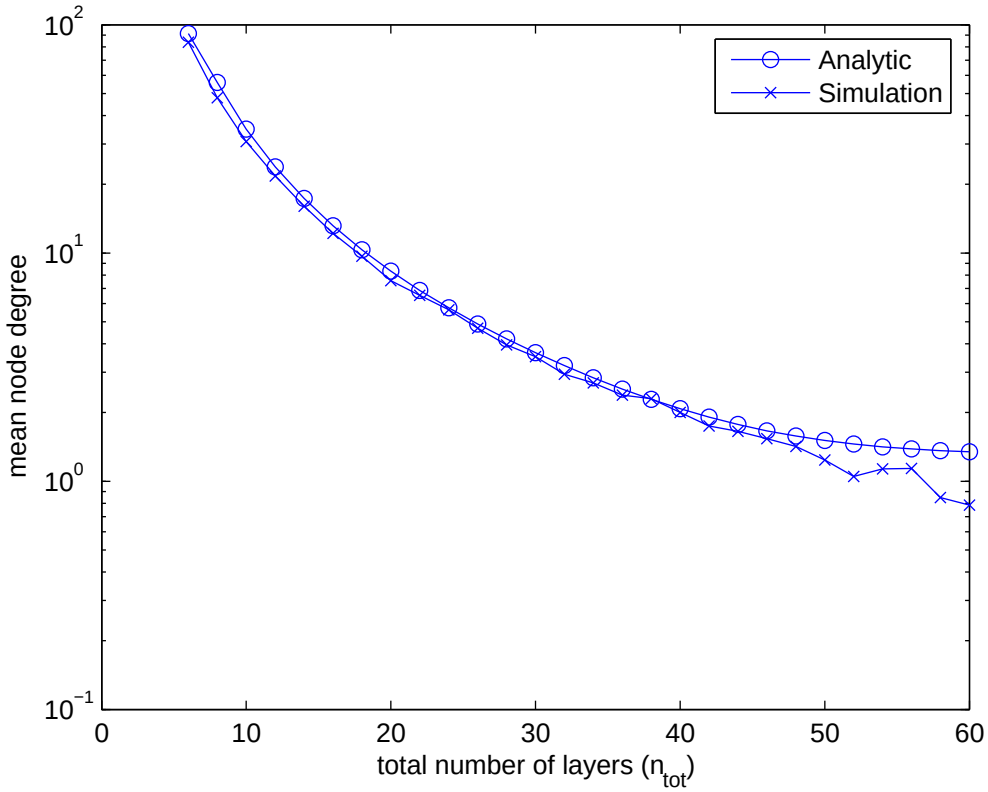

Figure 3.19: Mean node degree for different n_{tot} values.

3.4 Simulation Results

We conducted randomized simulations to verify the correctness of our analytic results in terms of the mean node degree. In our analytic model, we have obtained the mean node degree for a particular node located on the center cell of an hexagonal network as shown in Fig. 3.1.

In our simulations, we randomly distributed 106 mobile nodes on a hexagonal geographic area partitioned into hexagonal cells as shown in Fig. 3.1 (i.e., n_{tot}=4). Different network densities can be obtained by varying a single parameter among N, n_{tot}, and n_{av}. In this

case, N and n_{av} are fixed to 106 and 5 while we repeated our simulations for different n_{tot} values to simulate the different dense networks. n_{tot} is varied from 6 to 60 to obtain the broad range of different network densities. Each mobile node roams following a discrete-time random walk model. In order to provide a consistency with our analytic model, after each time unit we checked the mobile nodes located on the center cell (0,0), and reported the node degree only for these mobile nodes. One mobile node is said to be a neighbor of another node if the distance between these two nodes is less than or equal to n_{av} layers (i.e., n_{av}=5). We used 50,000 time units in these simulations and took the average of the node degrees.

Fig. 3.19 shows the mean node degree obtained from the simulation study and our numerical analysis for different n_{tot} values. As can be observed from Fig. 3.19, the analytic results match the simulation results, where the mean node degree decreases as the network size (n_{tot}) increases while the number of mobile nodes is kept constant. In Fig. 3.19, there is a slight difference between the analytic and simulation results when n_{tot} is between 50 and 60. Since the number of visits to the center cell (0,0) in the simulation decreases as the network size increases, the mean node degree may not be accurate for the simulation case.

From the second set of simulations, Fig. 3.20 shows the expected first times to reach different d_{thr} values obtained from the simulation experiments and numerical analysis for the medium density network I. In Fig. 3.20, all nodes are roamed randomly following the

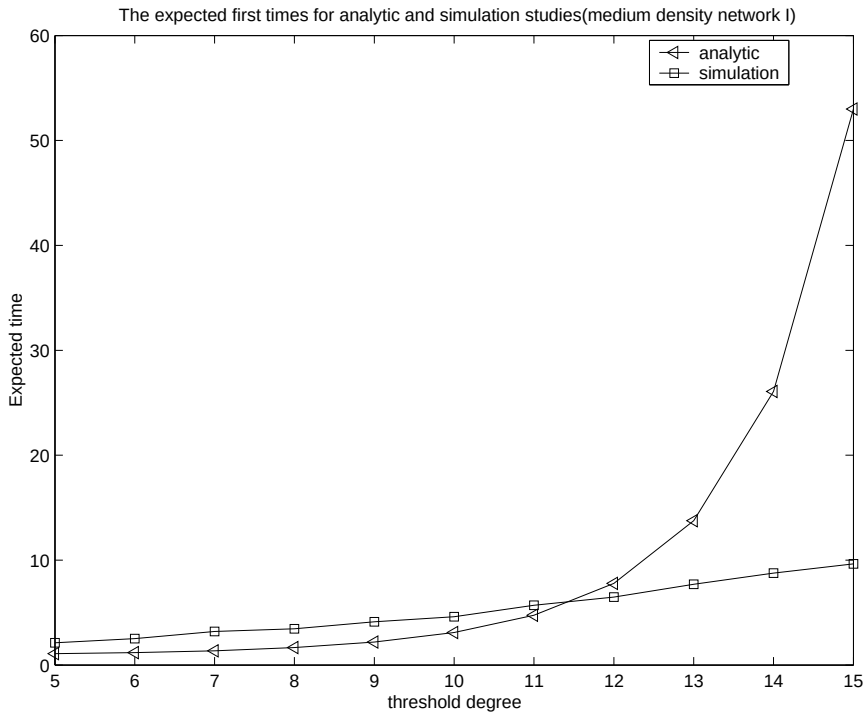

Figure 3.20: Expected first time obtained from the simulation study against our numerical analysis for the medium density network I.

discrete-time random walk model and the expected first times are collected only for the nodes which are not on or close to the boundary cells. A cell is defined as a non-boundary cell if its shortest distance to outmost cells is less than n_{av} layers (i.e., $n_{av} = 5$ for the medium density network I). For a node located on a non-boundary cell, the timer starts when this node has a degree less than d_{thr} and stops when the degree is equal to or higher than d_{thr} for the first time. We repeated this procedure for all nodes which are on the non-boundary cells for 80,000 time units and took the average of the first passage times that this event happens the first time. As seen from Fig. 3.20, the numerical results are close to the simulation results especially around the mean node degree ($\overline{N}(20, 5) = 8.33$), where the expected first time increases with d_{thr} in both cases.

The difference between the numerical and simulation results increases when d_{thr} is greater than 13; however, having a node degree greater than 13 for the medium density network I is not typical. One can conclude that the regime which our model accurately predicts the expected first times is around the mean node degree. The degrees which are very small or very high compared to the mean degree are typically less of an interest for network designers.

Chapter 4

Comprehensive Analytic Models for Node Link Stability

Our models presented in Chapter 3 use steady state approximation of link creation and failure probabilities to study node link stability in MANETs. In this chapter, we introduce new comprehensive analytic models to study node link stability for more realistic wireless mobile network applications. These comprehensive models provide more realistic tools for MANET behavior by *(i)* calculating the exact link failure and creation probabilities, *(ii)* allowing a node to have different speeds, and *(iii)* being a relatively simple and computationally efficient. Our comprehensive analytic models hence can be used to evaluate the performance of algorithms and protocols at different layers of MANETs.

The models divide a geographic area into logical cells, where each node roams into one of its neighboring cells by following the discrete-time random walk mobility model. The wireless link creation/failure probabilities are derived to represent transitions from an available to an unavailable state (or vice versa) as nodes move. We further derive a two dimensional Markov chain, $P2$, whose states represent the node degree and ($nlff$) for a node. The node degree is defined as the number of active neighbors and the *nlff* metric is the number of new link departures within a time-window, but normalized by the node degree (see for example Kozat et al. [61]). A numerically efficient first passage time analysis is applied to $P2$ for calculating the expected first passage times to move between states (i.e., black or white) defined by degree and *nlff* values. When a node's degree is equal to or higher than a certain degree threshold and its *nlff* value is smaller than an *nlff* threshold, this state will be *black*, otherwise *white*. In other words, a *black* state represents a stable node with a desired node degree. The expected time it takes for a node to change its color from *black* to *white* (or vice versa) provides a valuable indicator to determine stability of a node link in a MANET. In our model, we were able to represent different network characteristics by varying one or more of the following three parameters: *(i)* the total number of layers (n_{tot}) (i.e., the geographic area size), *(ii)* the number of available layers (n_{av}) (i.e., a node's communication range), and *(iii)* the number of mobile nodes.

In Section 4.1, our new node link stability models are presented. Expected time calculations using an efficient first passage time analysis is described in Section 4.2. Model

applications are highlighted in Section 4.3. Numerical and simulation results are presented in Sections 4.4 and 4.5, respectively.

4.1 Comprehensive Analytic Models

4.1.1 Comprehensive Mobility Patterns

A potential limitation of the random walk mobility model presented in Chapter 3 is that it allows a node to roam into its neighboring cells with only a fixed speed. However, in real-life scenarios, a node may move in any direction with any speed (i.e., random way point). Therefore, the random walk mobility pattern should be extended to cover the fast/slow node movements with different speeds. This extension will allow our model to analyze more realistic scenarios in MANETs where speed variation affects the system performance.

For modelling the fast/slow movements in a hexagonal geographic area, suppose there are six possible directions with two possible speeds, namely v and $h \times v$, where $h = 1, 2, \ldots$. A node with the speed of v will move into a neighboring cell which is one layer away, while another node with the speed of $h \times v$ into a cell which is h layers away. Compared to the 19 possible next link states for the fixed speed in Table 3.1, this extended case with two different speeds will have 73 next link states for an initial link state of $< x, y >$ as shown

in Table 4.1, where $A = \alpha^2/36$, $B = \beta^2/36$, $C = \alpha\beta/18$, and α and β are the probabilities that a node moves with a speed of v and $h \times v$ in the next time unit, respectively. Since only two speeds exist for this case, $\alpha + \beta = 1$. In this framework, one can vary the parameters α, β, and h to obtain a broad range of mobility scenarios. Note that when α is set to 1 (i.e., β=0), the random walk model described in Section 3.1.1 will be obtained, where Table 4.1 will only have 19 non-zero probabilities (i.e., the probabilities that contain A) and Table 4.1 becomes identical to Table 3.1.

4.1.2 Formal Definitions

Let us now present the formal definitions and notations for the analytic mobility models introduced above. For the general case, a mobility model for a MANET with a comprehensive mobility pattern can be defined as:

***Definition* 1** *Let $M_C(D, \alpha, v, \beta, h \times v, n_{tot})$ be a Markovian Chain with each link state representing a physical distance between two nodes, where:*

- *Each node moves into one of D directions, each with $1/D$ probability, in one time unit ($D = 1, 2, 3, \ldots$),*
- *Each node speed is either v with probability α, or $h \times v$ with probability β ($0 \leq \alpha, \beta \leq 1$, $\alpha + \beta = 1$, $h = 1, 2, \ldots$),*

Table 4.1: Probability distribution for a wireless link to switch from state $< x, y >$ to state $< x', y' >$, where a node can move with speeds of v and $h \times v$ with probabilities of α and β, respectively ($A = \alpha^2/36$, $B = \beta^2/36$, and $C = \alpha\beta/18$)

x',y'	x,y	x,y+h-1	x+1,y-1	x+h,y-1	x+1,y-2	x+h,y-h-1
Probability	6(A+B)	C	2A	C	2A	C
x',y'	x,y-h+1	x+1,y-h	x+h,y-h	x+1,y-h-1	x+h,y-2h	x,y-2h
Probability	C	C	2B	C	2B	B
x',y'	x+h-1,y	x,y-1	x+h-1,y-h	x-2,y	x-h-1,y	x-2,y+1
Probability	C	2A	C	A	C	2A
x',y'	x-2h,y	x-h-1,y+1	x-2h,y+h	x-1,y+2	x-1,y+h+1	x,y+1
Probability	B	C	2B	2A	C	2A
x',y'	x-h+1,y+h	x,y+h	x-h+1,y+h-1	x-2h,y+2h	x,y+2	x,y+h+1
Probability	C	2B	C	B	A	C
x',y'	x+h,y+h	x+1,y+h-1	x+h,y	x+2,y	x+h+1,y	x+2,y-1
Probability	2B	C	2B	A	C	2A
x',y'	x-h,y+h-1	x-1,y+h	x,y-h	x-h,y+2h	x+1,y+h	x+h+1,y-h+1
Probability	C	C	2B	2B	C	C
x',y'	x-h+1,y-1	x-1,y+1	x+2,y-2	x,y+2h	x-1,y	x+h,y-h+1
Probability	C	2A	A	B	2A	C
x',y'	x-1,y-1	x-1,y-h+1	x-h,y+h	x-2,y+2	x+1,y	x+h+1,y-1
Probability	2A	C	2B	A	2A	C
x',y'	x,y-2	x,y-h-1	x-1,y-h	x-h,y-h	x-h-1,y+h	x-h,y+1
Probability	A	C	C	2B	C	C
x',y'	x+h-1,y+1	x+h-1,y-h+1	x+1,y+1	x+h,y+1	x+h+1,y-h	x+2h,y
Probability	C	C	2A	C	C	B
x',y'	x+2h,y-2h	x-h,y+h+1	x-h+1,y	x-h,y	x-h-1,y+h+1	x-h,y-1
Probability	B	C	C	2B	C	C
x',y'	x+2h,y-h					
Probability	2B					

- *n_{tot} represents the maximum number of layers between two mobile nodes,*
- *Each link state bounces back to one of the feasible link states to meet the criteria that the distance between two nodes can be up to n_{tot} layers.*□

M_C is a comprehensive model which includes different parameters, each with its own effect on the probability distribution of M_C. Analyzing mobility patterns using a fixed speed will provide valuable insight towards developing an effective methodology to study system behavior with different node speeds. In this study, we introduce a simplified case of M_C defined below.

***Definition* 2** *Let $M \equiv M_C(D = 6, \alpha = 1, v, \beta = 0, h \times v, n_{tot})$, where nodes move into one of the six possible directions with a fixed speed of v (as shown in Fig. 3.2).*□

In this thesis, without loss of generality, the link state transition matrix for the simplified model M is also referred to as M.

4.1.3 Construction of Link State Transition Matrix

For the simplified model M, let us develop a state transition diagram using the probabilities from Table 3.1. An algorithm to construct M is given in Fig. 4.1, which accepts n_{tot}, and the triplet values of T_X, T_Y, and T_P from Table 3.1 as its inputs. For example, for the initial state of $< x, y >$ and the next state of $< x + 1, y >$, the triplet values are

```
Inputs: n_tot, T_X, T_Y, T_P
Output: M
M(:,:) = 0, sum_f(:) = 0, sum_inf(:) = 0

1  begin
2    for y_cur = 0 : ⌈n_tot/2⌉ {
3      for x_cur = y_cur : n_tot − y_cur {
4        i = findIndex(x_cur, y_cur)
5        for k = 1 : 19 {
6          x_new = x_cur + T_X(k)
7          y_new = y_cur + T_Y(k)
8          j = findIndex(x_new, y_new)
9          if (i <= S_{T−2}) {
10           M(i,j) = M(i,j) + T_P(k)
11         }
12         else if (S_{T−2} < i <= S_T) {
13           if (j <= S_T) {
14             M(i,j) = M(i,j) + T_P(k)
15             sum_f(i) = sum_f(i) + T_P(k)
16           }
17           else {
18             sum_inf(i) = sum_inf(i) + T_P(k)
19           }
20           else {}
21         }
22       }
23     }
24   }
25   for i = S_{T−2} + 1 : S_T {
26     for j = S_{T−4} + 1 : S_T {
27       M(i,j) = M(i,j) × (1 + (sum_inf(i)/sum_f(i)))
28     }
29   }
30 end
```

Figure 4.1: Algorithm to construct link state transition matrix M

```
Inputs: x, y
Outputs: I_{x,y}, L_{x,y}, x_e, y_e

1  function [I_{x,y}, L_{x,y}, x_e, y_e] = findIndex(x, y)
2     if ((x < 0 & y < 0) || (x >= 0 & y >= 0)) {
3        x_e = max(abs(x), abs(y))
4        y_e = min(abs(x), abs(y))
5     }
6     else {
7          x_e = max(abs(x + y), min(abs(x), abs(y)))
8          y_e = min(abs(x + y), min(abs(x), abs(y)))
9     }
10    L_{x,y} = x_e + y_e
11    if (L_{x,y} == 0) {
12       I_{x,y} = 1
13    }
14    else if (L_{x,y} == 1) {
15       I_{x,y} = 2
16    }
17    else {
18       if (L_{x,y} is odd) {
19          I_{x,y} = (L_{x,y} - 1) × (L_{x,y} + 3)/4 + y_e + 2
20       }
21       else {
22          I_{x,y} = L_{x,y} × (L_{x,y} + 2)/4 + y_e + 1
23       }
24    }
25 end
```

Figure 4.2: Pseudocode for `findIndex` function

1 (i.e., $x + 1 - x = 1$), 0 (i.e., $y - y = 0$), and 1/36, respectively. In Fig. 4.1, the first two `for` loops consider all possible x and y values for a link state. The `for` loop between lines 5 and 22 calculates $M(i, j)$ values for the possible 19 next link states (without the infea-

sible states). Finally, the `for` loop between lines 25 and 29 distributes the infeasible state transition probabilities among the feasible ones. As a result, when the center-to-center distance between two mobile nodes is equal to $(n_{tot} - 1) \times R$, or $n_{tot} \times R$, it will be less than or equal to $n_{tot} \times R$ in the next time unit. In the calculation of $M(i, j)$ values, the indices of x_{cur}, y_{cur} (in line 4) and x_{new}, y_{new} (in line 8) are calculated using the algorithm called `findIndex` which maps a link state $< x, y >$ to its equivalent state $< x_e, y_e >$ (lines 2-9 in Fig. 4.2), and then its index $I_{x,y}$ (lines 11-24 in Fig. 4.2) in M.

Once the values of M matrix is calculated, the corresponding state transition diagram for a wireless link can be obtained as shown in Fig. 3.3 (to simplify the drawing, only the transitions from the link states of layers less than or equal to 5 are shown).

***Theorem* 1** *The number of states in M is solely dependent on n_{tot} and can be reduced to:*

$$S_T = \begin{cases} 1 & \text{if } n_{tot} = 0 \\ \dfrac{(n_{tot}+1) \times (n_{tot}+3)}{4} & \text{if } n_{tot} > 0 \text{ and } n_{tot} \text{ is odd} \\ \dfrac{n_{tot} \times (n_{tot}+4)}{4} + 1 & \text{if } n_{tot} > 0 \text{ and } n_{tot} \text{ is even} \end{cases} \tag{4.1}$$

***Proof Sketch*:** All possible link states in Fig. 3.1 can be represented by cells shown in thicker boundaries (i.e., the state reduction). One can observe that the number of states (i.e., cells with thicker boundaries) for layers 0 and 1 are 1, for layers 2 and 3 are 2, for layer 4 is 3 and so on. This sequence can be generalized as S_T values given in this theorem.□

Let us now consider the number of link states in a more general case of M_C:

***Theorem* 2** *The number of states in $M_C(D, \alpha, v, \beta, h \times v, n_{tot})$ solely depends on n_{tot}.*

Proof Sketch: In Fig. 3.1, the number of different link states depends solely on the number of layers and is independent of the node speed. Therefore, the number of different states in the state transition diagram shown in Fig. 3.3 will remain the same for the two speed case (the associated probabilities in the state transition diagram will be different). This fact can also be explained by the method used to construct M matrix, where all link states are spanned by the parameters x_{cur} and y_{cur}, which depend solely on n_{tot}.□

From Theorems 1 and 2, S_T is only function of n_{tot} which leads to the following corollary for the general case of M_C:

***Corollary* 1** *Increasing the number of directions and different speeds does not affect the number of link states in M_C (for non-zero speeds).*

Using the mobility model M_C, a new model for representing node degree and *nlff* can be defined as follows:

***Definition* 3** *Let $P2_C(M_C, N, n_{av})$ be a Markov Chain, where:*

- *M_C is the Markov Chain describing the underlying comprehensive mobility pattern,*

- *N is the number of mobile nodes,*

- *n_{av} is the communication range, in terms of layers, of a mobile node.*□

In this thesis, we consider a simplified case of $P2_C$, called $P2$, as defined below:

***Definition* 4** *Let $P2 \equiv P2_C(M, N, n_{av})$, where M is the simplified case of M_C given in Definition 2.*□

We present the formulations to obtain $P2$ from M, N and n_{av} in the following sections.

4.1.4 Exact Link Failure and Creation Probabilities

Each element $M_{i,j}$ in the state transition matrix $M = \{M_{i,j}\}$, which is constructed in Section 4.1.3, represents the probability of moving from the i-th link state to j-th link state. For a wireless link with initial state i, $P_a(i)$ and $P_u(i)$ denote the probabilities that the link will be available or unavailable in the next time unit, respectively:

$$P_u(i) = \sum_{j=s_a+1}^{S_T} M_{i,j} = 1 - \sum_{j=1}^{s_a} M_{i,j} = 1 - P_a(i) \tag{4.2}$$

$$P_a(i) = \sum_{j=1}^{s_a} M_{i,j} \tag{4.3}$$

where j represents available link states for $1 \leq j \leq s_a$ and unavailable link states for $s_a + 1 \leq j \leq S_T$. The number of available link states, s_a, can be found by replacing n_{tot} with n_{av} in Theorem 1.

Up to this point, we only consider a single wireless link between two nodes and calculate the probabilities of its availability and unavailability as nodes move. Now, we will use these probabilities to study more generic metrics such as node degree and *nlff* in two dimensional regions. Consider a particular node z located in cell (0,0) of a network with N nodes. There are $K = N - 1$ possible bi-directional links from z to all other nodes. The following formulation is presented for this particular node z with initially k available and $K_u = K - k$ unavailable links.

Let $P_{dap}(k, l)$ denote the probability that l of k available links will disappear and k-l of k available links will remain available. Suppose, for example, there are 4 available links whose initial states are i_1, i_2, i_3, and i_4. $P_a(i_m)$ and $P_u(i_m)$ will most likely be different for each link state i_m, where $m = 1, \ldots, 4$. Then, for example, the probability that 2 links will disappear in the next time unit can be calculated by adding the probabilities of $\binom{4}{2}$=6 possible combinations. One of these combinations is that links i_1 and i_2 will disappear and i_3 and i_4 will stay available in next time unit. This probability can be calculated as follows:

$$P_u(i_1) \times P_u(i_2) \times P_a(i_3) \times P_a(i_4) \tag{4.4}$$

The probabilities for the remaining 5 combinations need to be calculated as well to find

the exact probability that 2 links will disappear in the next time unit given that initial link states are known. However, for an available link state, the initial link state could be any state among the possible available link states and the stationary vector of M will be used to calculate the probability distribution of these link states. Let f is the stationary vector of M and be partitioned conformally as in $f = (f_a f_u)$, where f_a and f_u are the subvectors of f associated with available and unavailable link states, respectively. Then,

$$P_{dap}(k,l) = \sum_{m=1}^{s_a} \frac{f_a(i_m)}{\|f_a\|_1} \times \binom{k}{l} \times P_u^l(i_m) \times P_a^{k-l}(i_m) \tag{4.5}$$

where $f_a(i_m)$ is the probability that the available link state is i_m for $1 \leq m \leq s_a$ and $\|f_a\|_1$ is used to normalize the length of the subvector f_a to 1. Here $P_u(i_m)$ denotes the probability of becoming unavailable in the next time unit given that the link state is i_m initially and $P_u^l(i_m)$ represents the l-th power of $P_u(i_m)$. Let P_{au} and P_{aa} represent the probabilities that a wireless link will be unavailable and available, respectively, in the next time unit given that it is available initially:

$$P_{au} = \sum_{m=1}^{s_a} \frac{f_a(i_m)}{\|f_a\|_1} \times P_u(i_m) \tag{4.6}$$

$$P_{aa} = \sum_{m=1}^{s_a} \frac{f_a(i_m)}{\|f_a\|_1} \times P_a(i_m) \tag{4.7}$$

When we use Eqs. (4.6) and (4.7), Eq. (4.5) will become:

$$P_{dap}(k,l) = \binom{k}{l} \times P_{au}^l \times P_{aa}^{k-l} \tag{4.8}$$

Similarly, let $P_{ap}(K_u, l+h)$ denote the probability that $l+h$ of K_u unavailable links will

appear and $K_u - l - h$ of K_u unavailable links will remain unavailable in one time unit:

$$P_{ap}(K_u, l+h) = \sum_{m=s_a+1}^{S_T} \frac{f_u(j_m)}{\|f_u\|_1} \times \binom{K_u}{l+h} \times P_a^{l+h}(j_m) \times P_u^{K_u-h-1}(j_m) \tag{4.9}$$

where S_T represents total number of possible link states, $f_u(j_m)$ represents the probability mass function of the unavailable link states for $s_a + 1 \leq m \leq S_T$ and $\|f_u\|_1$ is used to normalize the length of the subvector f_u to 1. Let P_{uu} and P_{ua} represent the probabilities that a wireless link will be unavailable and available, respectively, in the next time unit given that it is unavailable initially:

$$P_{uu} = \sum_{m=s_a+1}^{S_T} \frac{f_u(j_m)}{\|f_u\|_1} \times P_u(j_m) \tag{4.10}$$

$$P_{ua} = \sum_{m=s_a+1}^{S_T} \frac{f_u(j_m)}{\|f_u\|_1} \times P_a(j_m) \tag{4.11}$$

When we use Eqs. (4.10) and (4.11) in Eq. (4.9):

$$P_{ap}(K_u, l+h) = \binom{K_u}{l+h} \times P_{ua}^{l+h} \times P_{uu}^{K_u-l-h} \tag{4.12}$$

4.1.5 Comprehensive Markov Chain for Node Degree and *nlff*

In the previous section, we derive the probabilities of l link departures out of k available links and $l + h$ new link arrivals out of K_u unavailable links. Let $P2_{(k,y),(k+h,l)}$ represent that these two events happen at the same time:

$$P2_{(k,y),(k+h,l)} = P_{dap}(k, l) \times P_{ap}(K_u, l+h) \tag{4.13}$$

where $0 \leq k, k+h, y \leq K$ and $0 \leq l \leq k$. By substituting Eqs. (4.8) and (4.12) in Eq. (4.13), we obtain:

$$P2_{(k,y),(k+h,l)} = \binom{k}{l} \times \binom{K_u}{l+h} \times P_{aa}^{k-l} \times P_{au}^{l} \times P_{ua}^{l+h} \times P_{uu}^{K_u-l-h} \tag{4.14}$$

Eq. (4.14) represents a new two-dimensional finite state Markov chain, $P2$, obtained using the state transition matrix $M = \{M_{i,j}\}$. For a state (k, y), k and y represent the number of available links and the number of new link departures for a specific node, respectively. The *nlff* metric is defined as the number of new link departures within a time-window, but normalized by the node degree at the end of the window. Then, a state $(k+h, l)$ will represent the node degree, (i.e., $k+h$), and the *nlff* value, (i.e., $l/(k+h)$). Here, we assume that the time-window for calculating the *nlff* value is equal to one time unit. Note that the state transition probabilities does not depend on y since the *nlff* value is calculated using only one time unit for the sake of simplicity. However, the model can be extended to cover multiple time units for the *nlff* calculation.

Markov chain $P2$ is ergodic (i.e., finite, connected, and aperiodic), and hence possesses a stationary distribution. Suppose that the stationary solution of this process is $\theta = \{\theta_r, r = 1, 2, \ldots, S_T\}$ and partitioned conformally as in $\theta = (\theta_0 \theta_d)$, where θ_0 and θ_d are the subvectors of θ associated with states of having zero and non-zero node degrees, respectively. Let $\overline{nlff}$ denote the mean *nlff* value:

$$\overline{nlff} = \sum_{r=s_n+1}^{S_T} f(r) \times \frac{\theta_d(r)}{\|\theta_d\|_1} \tag{4.15}$$

where $f(r)$ is a function mapping the state index r into its actual *nlff* value which is defined only for a non-zero node degree (e.g., indexes 1 and 2 correspond to state (0,0) and (0,1), respectively, index s_n+1 corresponds to state (1,0), and so on). Here, s_n is the highest index having a zero node degree, hence, the states $r = 1, 2, \ldots, s_n$ are excluded from the calculation since their node degrees are equal to zero.

4.1.6 One Dimensional Comprehensive Markov Chain for Node Degree

The states in $P2$, which represent the same node degree but different number of new link departures, can be aggregated into a single state if only node degree is sought. Then, for a node z, with initially k available links, the probability of having $k+h$ available links in the next time unit is:

$$P_{k,k+h} = \sum_{l=0}^{k} P_{dap}(k,l) \times P_{ap}(K_u, l+h) \tag{4.16}$$

where $0 \leq k, k+h \leq K$. By substituting Eqs. (4.8) and (4.12) in Eq. (4.16), we obtain:

$$P_{k,k+h} = \sum_{l=0}^{k} \binom{k}{l} \times \binom{K_u}{l+h} \times P_{aa}^{k-l} \times P_{au}^{l} \times P_{ua}^{l+h} \times P_{uu}^{K_u-l-h} \tag{4.17}$$

Eq. (4.17) represents a new finite state Markov chain, P, where a state k is the number of available links for a specific node. Markov chain P is ergodic (i.e., finite, connected, and aperiodic), and hence possesses a stationary distribution. Suppose that the stationary

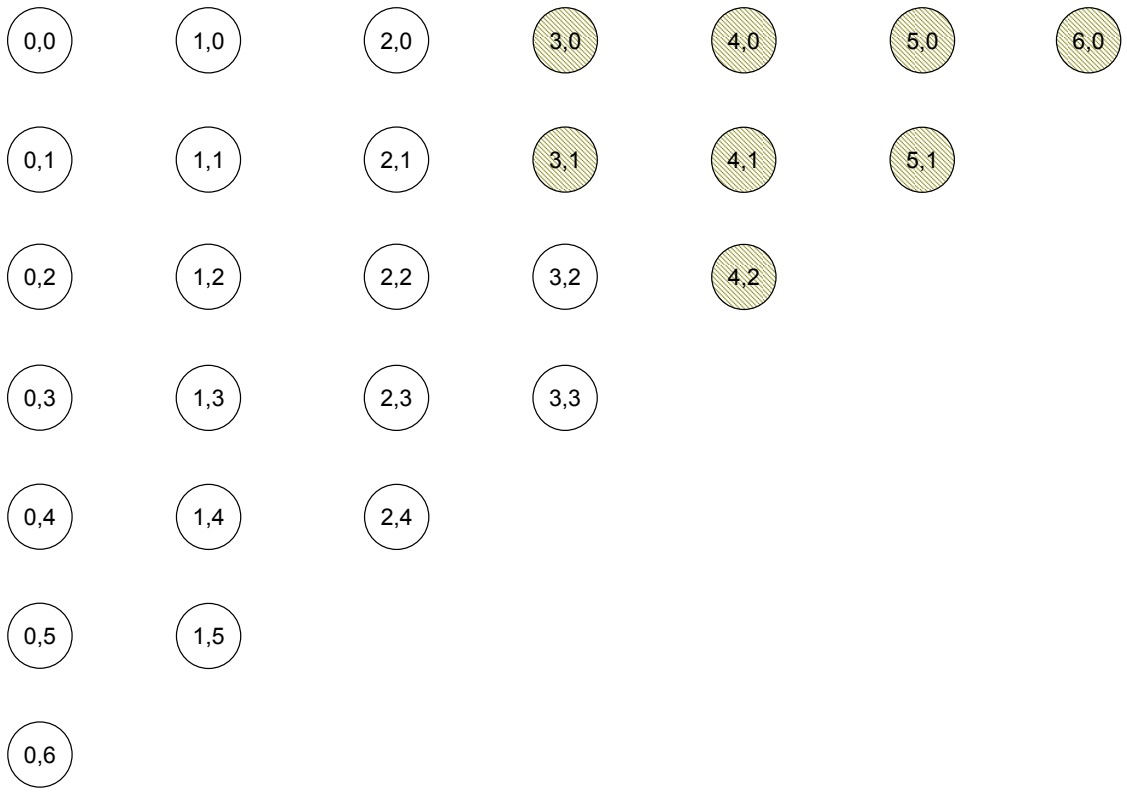

Figure 4.3: An example two-dimensional Markov chain representing node degree and *nlff* for 7-nodes network.

solution of this process is $\pi = \{\pi_k, k = 0, 1, \ldots, K\}$.. Let $\overline{N}$ denote the mean number of available links for a particular node, which can be calculated using the stationary solution π of P:

$$\overline{N} = \sum_{k=0}^{K} k \times \pi_k \tag{4.18}$$

4.2 Expected Time Calculation Using First Passage Times

When a node's degree is equal to or higher than a certain degree threshold (d_{thr}) and its *nlff* value is smaller than an *nlff* threshold ($nlff_{thr}$), this node will be *black*, otherwise *white*. In other words, *black nodes* represent stable nodes with a desired node degree.

Now, we will describe how $P2$ can be partitioned into *white* and *black* states. For a state (k, y) in $P2$, the node degree k and the *nlff* value y/k are compared with d_{thr} and $nlff_{thr}$, respectively. Based on the result of this comparison, the state is either *black* (b) if k is equal to or higher than d_{thr} and y/k is smaller than $nlff_{thr}$ or *white* (w) otherwise. Fig. 4.3 shows the states of $P2$ for a 7-nodes network. In this example, a particular node can have up to 6 neighbors and there are 28 different states for this node. If, for example, d_{thr} and $nlff_{thr}$ are set to 3 and 0.5, respectively, then 8 dark states represent the black states while the remaining 20 are the white states. The color of the state $(3, 1)$ is black since its node degree is equal to d_{thr} and its *nlff* value (i.e., 1/3) is less than $nlff_{thr}$=0.5 while the color of the state $(3, 2)$ is white since its *nlff* value (i.e., 2/3) is greater than $nlff_{thr}$=0.5. However, if we used the only node degree case, P, with d_{thr}=3, then there would be 3 white and 4 black states.

The node degree and *nlff* values of a node change with nodes' random movements, and hence the node's color may switch from one to another. The expected time it takes for a

node to change its color from black to white (or vice versa) provides a valuable indicator to determine node link stability in a wireless mobile network. We will calculate the expected first time moving from white states to a black state (or vice versa) by applying the first passage time analysis on $P2$ in the following sections.

4.2.1 First Passage Time from White to Black

First, we partition the states of $P2$ into *white* and *black* states:

$$P2 = \begin{pmatrix} P2_{w,w} & P2_{w,b} \\ P2_{b,w} & P2_{b,b} \end{pmatrix}$$

where w and b are respectively the subsets of *white* and *black* states; let the numbers of states in these subsets be respectively n_w and n_b. Here $P2_{w,w}$ is the square matrix of order n_w obtained from $P2$ by deleting the rows and columns that correspond to the states in b.

We seek to compute the expected time to move from white states to any *black* state for the first time. Therefore, all the black states of $P2$ can be aggregated into a single black state by modifying the associated probabilities as follows:

$$K = \begin{pmatrix} P2_{w,w} & P2_{w,b}e \\ \frac{\pi_b}{\|\pi_b\|_1} P2_{b,w} & \frac{\pi_b}{\|\pi_b\|_1} P2_{b,b}e \end{pmatrix}$$

where e is a column vector of ones with appropriate length. The number of states in $P2$ is $n_w + n_b$ before the state aggregation while the number of states is $n_w + 1$ in the modified version of $P2$. This aggregation of states will alleviate the cost of calculating the expected time, and hence result in a computationally efficient algorithm even for large number of states for realistic MANETs.

Using the algorithm given in Ref. [28], the mean first passage time from a *white* state i_m to the *black* state i_{n_w+1} is $f^{(1)}_{i_m,i_{n_w+1}}$ (i.e., all black states are represented by the black state i_{n_w+1}). Let p is the stationary vector of $P2$ and be partitioned conformally as in $p = (p_w p_b)$, where p_w and p_b are the subvectors of p associated with white and black states, respectively. Then, the expected time that a white node will become black:

$$\overline{T}_b = \sum_{m=1}^{n_w} \frac{p_w(i_m)}{\|p_w\|_1} \times f^{(1)}_{i_m,i_{n_w+1}} \tag{4.19}$$

where $p_w(i_m)$ is the probability that the white state is i_m initially for $1 \leq m \leq n_w$ and $\|p_w\|_1$ is used to normalize the length of the subvector p_w to 1. Here, there are n_w different white states. The expected times to become black are different for different white states when their node degrees are not the same. Therefore, the expected time moving from white states to black states for particular network parameters is calculated as the mean of the expected times for all initial white states.

4.2.2 First Passage Time from Black to White

In this case, the expected first time moving from black states to a white state is calculated. After partitioning $P2$ into white and black states as explained in the previous section, we combine all white states into a single white state and apply the first passage time analysis on the modified $P2$ for calculating the expected time to move from black states to a white state for the first time. The modified $P2$ has n_b+1 states. Using the algorithm given in Ref. [28], the mean first passage time from a black state i_m to the white state i_{n_b+1} is $f^{(1)}_{i_m,i_{n_b+1}}$ (i.e., all white states are represented by the white state i_{n_b+1}). Then, the expected time that a white node will become black:

$$\overline{T}_w = \sum_{m=1}^{n_b} \frac{p_b(i_m)}{\|p_b\|_1} \times f^{(1)}_{i_m,i_{n_b+1}} \tag{4.20}$$

where $p_b(i_m)$ is the probability that the black state is i_m initially for $1 \leq m \leq n_b$ and $\|p_b\|_1$ is used to normalize the length of the subvector p_b to 1.

4.3 Analytic Model Applications

4.3.1 Virtual Backbone Stability

For locating and registering available services within mobile ad hoc networks, a distributed service discovery architecture that relies on a virtual backbone (VB) can be effectively used [61]. A dynamic backbone is formed in a given network topology such that each node

in the network is either a part of the backbone or one hop away from at least one of the backbone nodes. VB formation algorithms apply two metrics: node degree [90], or both node degree and *nlff* [61], to resolve conflicts between the nodes competing to join the backbone. Similar to Refs. [61], CEDAR [90] builds a routing core such that each node that needs to find a dominator (i.e., backbone or core node) selects the highest-degree node with the maximum effective degree in its first neighborhood.

For example, consider Fig. 4.4, where nodes 1 and 6 join the backbone. Node 7 and nodes 2-5 select nodes 6 and 1 as their cluster heads, respectively. In the selection phase, all nodes are initially considered undecided, and exchange light-weight hello beacons to build up their own neighborhood-information tables. Each node runs the distributed virtual backbone algorithm in an asynchronous manner, where any node that satisfies the stability constraint (*nlff* less than a given threshold)) joins the VB if it has the highest number of white neighbors (effective degree). For example, in Fig. 4.4, node 5 has recently moved; therefore, node 1 rather than node 5 joins the backbone because it is more stable (i.e., the *nlff* value of node 5 may higher than $nlff_{thr}$).

Formally, we represent the snapshot of a network as a topology graph $G(V, E)$, where V and E correspond to the set of network nodes and symmetric links, respectively. Graph G can be expressed as the union of G_{nlff} and G^c_{nlff}, where G_{nlff} is the subgraph that contains vertices with *nlff* lower than a given threshold and the edges incident on them, and G^c_{nlff} is its complement. For each $k \in V$, let $N(k)$ and $deg(k)$ be the set of k's

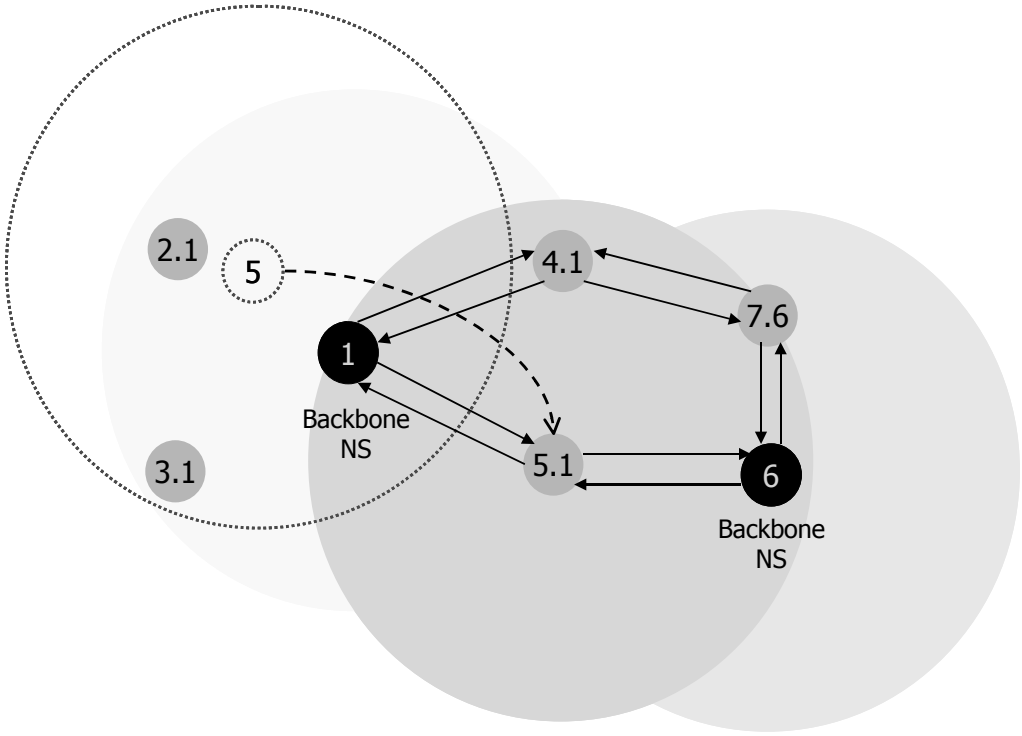

Figure 4.4: Virtual Backbone Formation.

neighbors and the degree of k in G, respectively. The subscript d refers to a subset of V containing decided nodes, i.e., the ones that have made a decision whether or not to join the backbone. Each such node $k \in V_d^c$ joins the backbone if it satisfies either of the following conditions:

CONDITION 1. $k \in V_{nlff} \wedge (\forall j \in V_d^c deg(j) \leq deg(k))$.

CONDITION 2. $k \notin V_{nlff} \wedge N(k) \cap V_{nlff} = \varnothing \wedge (\forall j \in V_d^c deg(j) \leq deg(k))$.

Our analytic model finds the average time it takes for not only the node degree but also the number of link changes (i.e., *nlff*) to either drop below or exceed given thresholds. Therefore, our model can be used as an estimation tool for the VB stability such that

the expected time to leave or join the backbone is approximated by certain node degree and *nlff* thresholds. Note that our analytic modeling characterizes a single node within a larger network: since degree/stability per node is just one factor in determining the backbone status, and the full VB algorithm involves message exchanges between subset of nodes, our approach can be considered the first step towards an accurate modeling of the VB stability.

4.3.2 Interference Analysis

In this section we demonstrate that using M we can analyze signal-to-interference ratio (SIR) in MANETs which is essential in studying network capacity. The fact that M represents link state transitions with respect to node movements it can be an effective tool to properly model SIR for wireless mobile networks.

A mobile node transmission is successful if the SIR at the receiver node is above a certain threshold value (SIR_{thr}). When and if the transmission power is kept constant for all mobile nodes, the SIR value at the receiver solely depends on the physical distances between the receiver and other mobile nodes. These distances changing with the dynamics of nodes' movements are available in our model, where a link state represents the physical distance between two mobile nodes. For example, for node z in cell (0,0) of a network with N nodes, there are $K = N - 1$ possible bi-directional links (available or unavailable) from z to all other nodes. Node i transmits data at rate R to node z if the SIR of the

node i at the node z is higher than the certain threshold value [41]:

$$\frac{P_i(t)\gamma_{iz}(t)}{N_0 + \frac{1}{L}\sum_{k\neq i} P_k(t)\gamma_{kz}(t)} > SIR_{thr} \tag{4.21}$$

where SIR_{thr} is SIR requirement for successful communication, N_0 is the background noise power, and L is the processing gain of the system. $\gamma_{iz}(t)$ represents the channel gain between nodes i and z at time t and is given by Ref. [41]:

$$\gamma_{iz}(t) = \frac{1}{d_{iz}^{\alpha}(t)} \tag{4.22}$$

where $d_{iz}(t)$ is the physical distance between nodes i and z and $\alpha > 2$ is a parameter. From our model, the stationary vector of M will be used to find the probability distribution of the physical distance between two mobile stations. Let f be the stationary vector of M and be partitioned conformally as in $f = (f_a f_u)$, where f_a and f_u are the subvectors of f associated with available and unavailable link states, respectively. Then,

$$\gamma_{iz} = \sum_{m=1}^{s_a} \frac{f_a(i_m)}{\|f_a\|_1} \times \frac{1}{d_{iz}^{\alpha}(i_m)} \tag{4.23}$$

where $d_{iz}(i_m)$ is the physical distance when the link state is i_m. Let R be the center-to-center distance between two neighboring hexagonal cells. In our model, for a link state i_m represented by $< x, y >$ in the state transition diagram in Fig. 3.3, $d_{iz}(i_m)$ can be calculated as:

$$d_{iz}(i_m) = \sqrt{x^2 + y^2} \times R \tag{4.24}$$

Due to the stationary vector f_a, the distances among mobile nodes in the next time unit

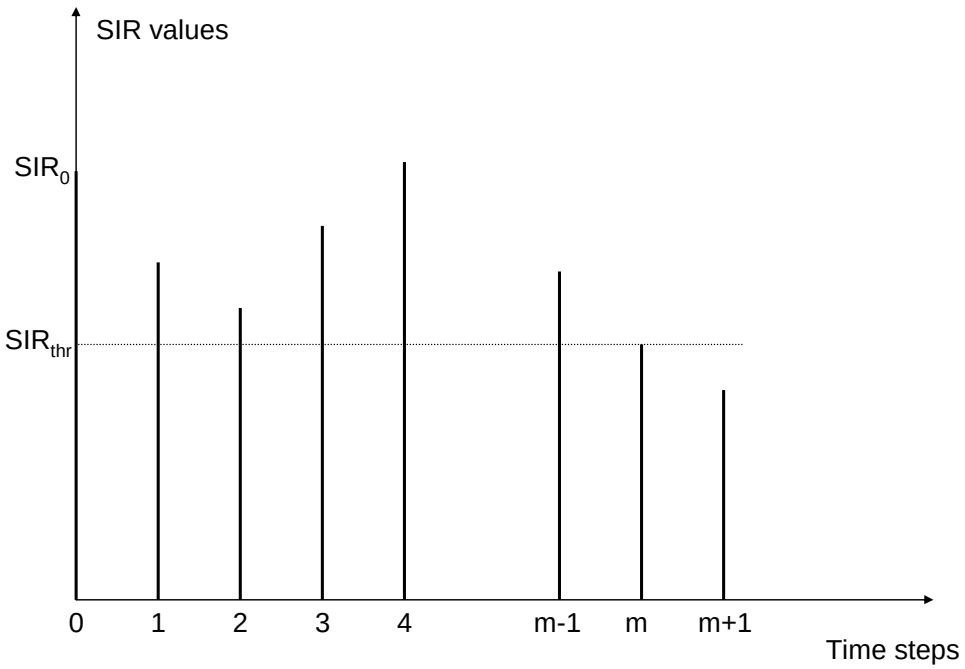

Figure 4.5: SIR change in each time step.

are independent and identically distributed (i.i.d.). If we assume that the transmission powers are kept constant (P) for all nodes, the steady state value of SIR can be calculated using Eq. (4.23) in Eq. (4.21):

$$SIR = \frac{\gamma_{iz}}{\frac{N_0}{P} + \frac{1}{L}\sum_{k \neq i} \gamma_{kz}} \tag{4.25}$$

Suppose a new random variable F represents the SIR value of a receiver node at the end of one time unit. Let $\overline{T}_{SIR}$ be the expected first time that a black node, whose initial SIR value SIR_0 is more than a threshold SIR_{thr}, will become a white node when its SIR value is equal to or less than SIR_{thr}. Suppose the first time that the SIR value of a node will meet or drop below SIR_{thr} happens at the m-th step, which implies that until the m-th step, the SIR value of this node remains more than SIR_{thr} (Fig. 4.5). Let a new

set of random variables $F_1, F_2, \ldots, F_m$ represent the SIR values for the 1-st, 2-nd, ..., and m-th steps, respectively. Then, a new random variable S_m represents the minimum SIR value among the SIR values encountered until the m-th step:

$$S_m = \min F_1, F_2, \ldots, F_m \tag{4.26}$$

B_m is the probability that the SIR value of a node will meet or drop below SIR_{thr} for the first time at the $m - th$ step:

$$B_m = \prod_{i=1}^{m} Pr(S_i \leq SIR_{thr} | (\forall j : 0 \leq j < i) S_j > SIR_{thr}) \tag{4.27}$$

From Eq. (4.27), the expected time that a receiver node with desirable SIR value will have undesirable SIR value can be calculated as:

$$\overline{T}_{SIR} = \sum_{m=0}^{\infty} m \times B_m \tag{4.28}$$

The expected first time $\overline{T}_{SIR}$ that a black node becomes a white node is defined by Eq. (4.28). To obtain a numerical value of $\overline{T}_{SIR}$, one should first calculate the probability mass function of the random variable F, and then solve Eqs. (4.26) and (4.27). These equations are hard to solve directly; therefore, we can use the probability distribution for the SIR value to create a new Markov chain representing the SIR values together with their transition probabilities (Eq. (4.23) needs to be modified slightly for this purpose; however, the details are omitted due to page limitations). The expected first time that the interference level is above or below the certain threshold value can be calculated by applying the first passage time analysis on the new Markov chain.

4.3.3 Network Connectivity and Convergence

Additional metrics that are directly available from our model include the expected link and route lifetimes, which can be applied to assessing the convergence of routing protocols by calculating the stability of neighbor and forwarding tables based on node mobility. The estimates of route lifetimes are useful in calculating the latency of on-demand routing protocols in ad hoc networks; they can also assist the routing protocol in selecting more stable routes.

Because the model is capable of computing the dynamics and the expected value of the number of a node's neighbors, it can also be used to estimate the time it takes for a node to lose connectivity to its routing domain. An example of optimal routing domains are the ones built by the Dynamic Domain Optimization Agent (DDOA) [40] that divides the network into routing domains (e.g., OSPF areas). The rate and expected values at which the nodes move in and out of domains characterize the rate of degradation of a network from its optimal partitioning into domains [67], and hence the achievable routing scalability and overhead.

Table 4.2: Network parameters for numerical experiments (N=46)

Network types	(n_{tot}, n_{av})	P_{aa}	P_{au}	P_{ua}	P_{uu}	$\overline{N}$	$\overline{nlff}$
Dense	(10,5)	0.8425	0.1575	0.0675	0.9325	13.5000	0.1710
Medium	(15,5)	0.8425	0.1575	0.0243	0.9757	6.0221	0.1959
Sparse	(20,5)	0.8425	0.1575	0.0128	0.9872	3.3936	0.2090
Sparsest	(30,5)	0.8425	0.1575	0.0055	0.9945	1.5111	0.1584

4.4 Numerical Results

In this section, we provide numerical results for characterizing $\overline{N}(n_{tot}, n_{av}, N)$ (the mean number of active neighbors), $\overline{nlff}(n_{tot}, n_{av}, N)$ (the mean normalized link failure frequency), $\overline{T}_b(n_{tot}, n_{av}, N)$ (the expected first time that a *white* node becomes a *black*) and $\overline{T}_w(n_{tot}, n_{av}, N)$ (the expected first time that a *black* node becomes a *white*). The results show that these metrics heavily depend on the network density which can be determined by varying a single parameter among n_{tot}, n_{av} and N. Table 4.2 shows the numerical values for the input parameters and the corresponding output metrics generated in our analysis. In Table 4.2, n_{tot} determines the geographic size of the network while n_{av} represents the communication range between two nodes, both in terms of layers. For all numerical results, N and n_{av} are fixed to 46 and 5, respectively. To represent mobile networks with different densities, n_{tot} is varied as 30, 20, 15, and 10 for *sparsest*, *sparse*, *medium density*, and *dense* networks, respectively.

In Table 4.2, P_{au} and P_{aa}, which denote the probabilities that an initially available wireless link will become unavailable and remain available in the next time unit, respectively, are calculated using Eqs. 4.6 and 4.7. Similarly, from Eqs. 4.10 and 4.11, P_{uu} and P_{ua} (the probabilities that an initially unavailable wireless link will remain unavailable and become available in the next time unit, respectively), are derived. From Eqs. 4.15 and 4.18, $\overline{nlff}$ and $\overline{N}$ are calculated as the mean *nlff* and the mean number of neighbors for particular network parameters, respectively.

The numerical results show that P_{aa} (and hence P_{au}) has the same value for different dense networks since it depends solely on n_{av} (i.e., n_{av} is 5 for all networks). However, P_{ua} (and hence P_{uu}) depends on n_{tot} and decreases (and hence P_{uu} increases) as the n_{tot} value increases with the network becoming sparser. $\overline{N}$, which has the highest value for the *dense* network, decreases as the network density decreases and reaches the lowest value for the *sparsest* network. Table 4.2 also shows that $\overline{nlff}$ values are close to each other for different dense networks, where it has the lowest value in the *sparsest* network and the highest value in the *sparse* network. Note that the *nlff* value is a non-linear function of both the number of link failures in the previous time window and the node degree, where both of their steady-state probability distributions change with respect to the network density. Therefore, it is predictable that we did not observe much differences in $\overline{nlff}$ values with respect to the network density (although we observed much differences in $\overline{N}$).

The expected first passage times using the values from Table 4.2 are discussed in detail

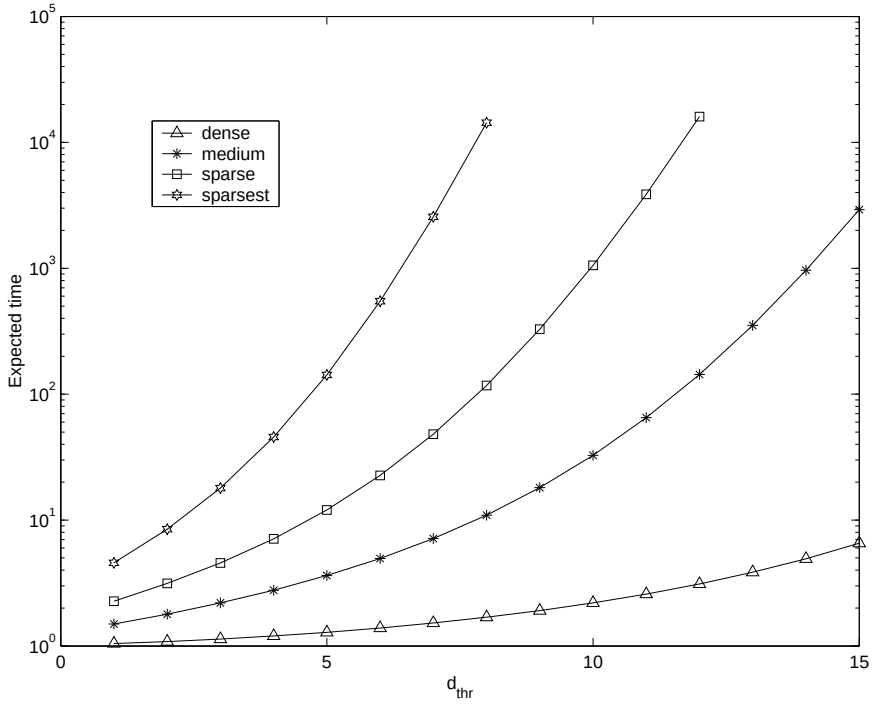

Figure 4.6: Analytic results of $\overline{T_b}(n_{tot}, n_{av}, N)$ versus d_{thr} for different density networks ($nlff_{thr}$=46)

below. In this study, one time unit represents the time for a mobile node to move from its current hexagonal cell to the next. This time unit depends on the size of the cell and the speed of the mobile node. The calculation of this time unit for different node speed and cell size distributions is beyond the scope of this study and will be handled as an extension of this work in the future.

4.4.1 Expected First Times from White to Black

In these experiments, $nlff_{thr}$ is kept constant and the effects of the node degree threshold (d_{thr}) and the network density over the expected *white* to *black* passage times are

studied. In Fig. 4.6, the numerical results for the expected passage times are presented using the logarithmic scale in the y-axis. Here, d_{thr} has been changed from 1 to 15 for four different network densities, whereas the $nlff_{thr}$ value is kept at N=46. The purpose of setting $nlff_{thr}$ to N is to observe the sole effects of d_{thr} over the passage times, without considering $nlff_{thr}$, since the *nlff* value cannot be greater than N-1 for $d_{thr} \geq 1$ (i.e., all states are considered as satisfying the *nlff* criteria).

In Fig. 4.6, for the same d_{thr} value, the expected passage time is always smaller for networks with high density than the sparse ones. In Table 4.2, P_{aa} (P_{au}) has the same value for different network densities, while P_{ua} (P_{uu}) decreases (increases) as the the network becomes sparser and n_{tot} increases. Hence, one can conclude that the probability of link failure is the same for all networks while the probability of new link arrivals is higher in denser networks and lowest in the sparsest. This result is intuitive since the expected passage times will be high if the passage probability is low, and vice versa.

As d_{thr} increases, the expected passage time increases but the amount of increase is not consistent for different network densities. This observation can be explained by the fact that networks with different densities have different mean node degrees. In Fig. 4.6, as d_{thr} exceeds the mean node degree for a given network, the expected passage time grows exponentially. For example, *sparsest* network with the mean node degree of $\overline{N}$=1.5111 starts growing exponentially for $d_{thr} \geq 2$ while the same behavior starts for *medium density* network with the mean node degree of $\overline{N}$=6.0221 when $d_{thr} \geq 6$. For *dense*

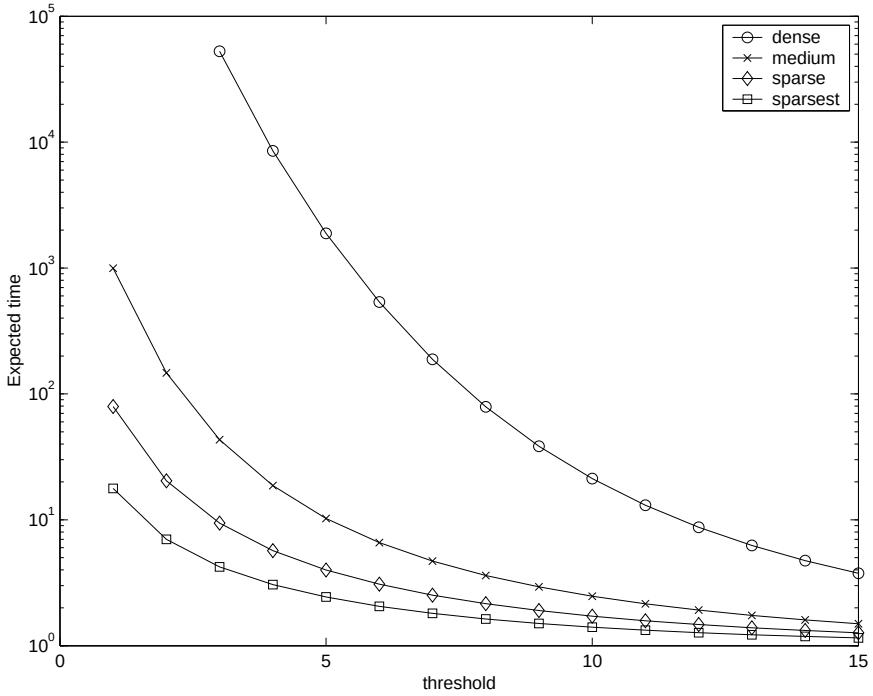

Figure 4.7: Analytic results of $\overline{T_w}(n_{tot}, n_{av}, N)$ versus d_{thr} for different density networks ($nlff_{thr}$=46)

network, the mean node degree is $\overline{N}$=13.5 and the exponential growth can not be observed since the maximum d_{thr} value is 15.

4.4.2 Expected First Times from Black to White

Using the same parameters in Section 4.4.1, the effects of d_{thr} and the network density over the expected *black* to *white* passage times are studied. In Fig. 4.7, for a given network, the expected first time to change a node's color from *black* to *white* decreases when d_{thr} increases. An exponential decline in the expected time is observed when d_{thr} is less than the mean node degree for all networks. For example, except for *dense* network, the mean

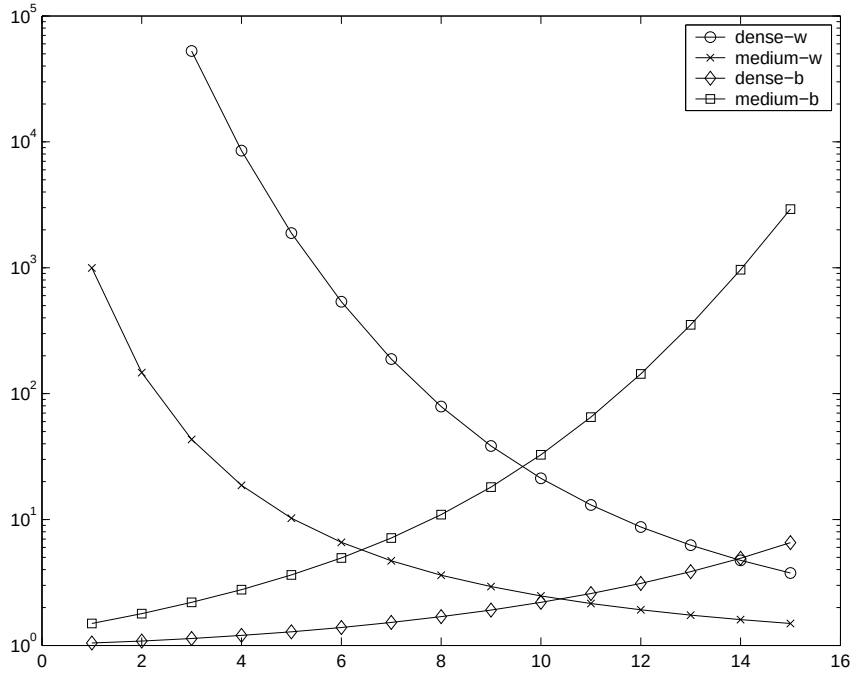

Figure 4.8: Analytic results of $\overline{T_b}(n_{tot}, n_{av}, N)$ and $\overline{T_w}(n_{tot}, n_{av}, N)$ versus d_{thr} ($nlff_{thr}$=46)

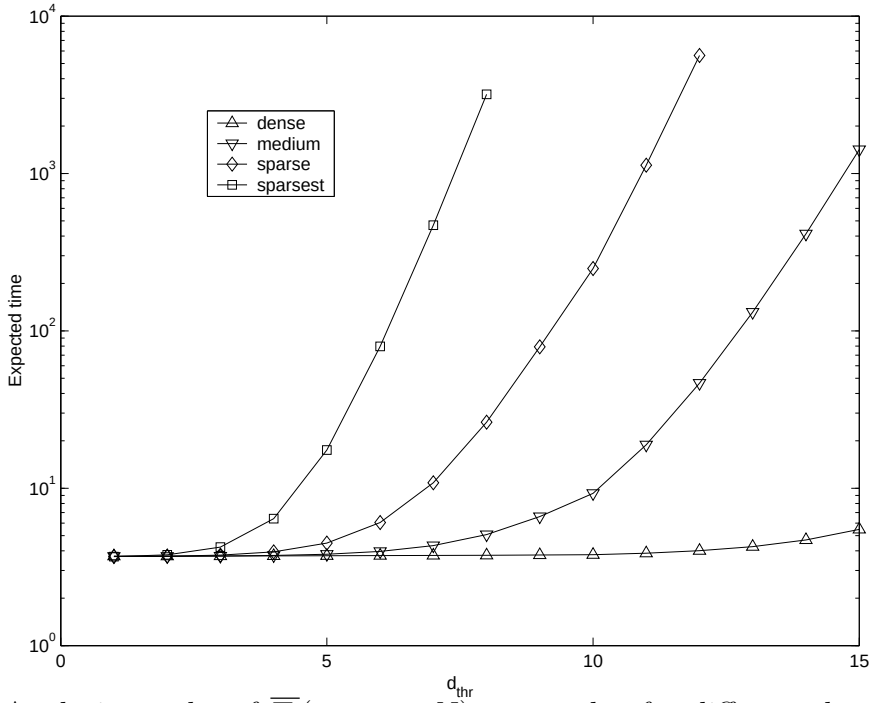

Figure 4.9: Analytic results of $\overline{T_b}(n_{tot}, n_{av}, N)$ versus d_{thr} for different density networks ($nlff_{thr} = 0.1$)

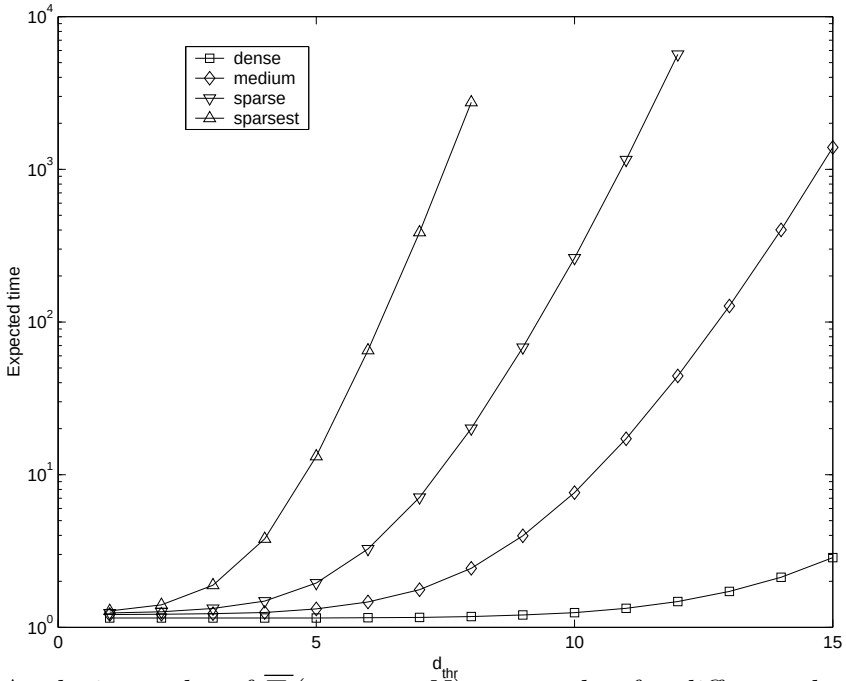

Figure 4.10: Analytic results of $\overline{T_b}(n_{tot}, n_{av}, N)$ versus d_{thr} for different density networks ($nlf f_{thr} = 0.3$)

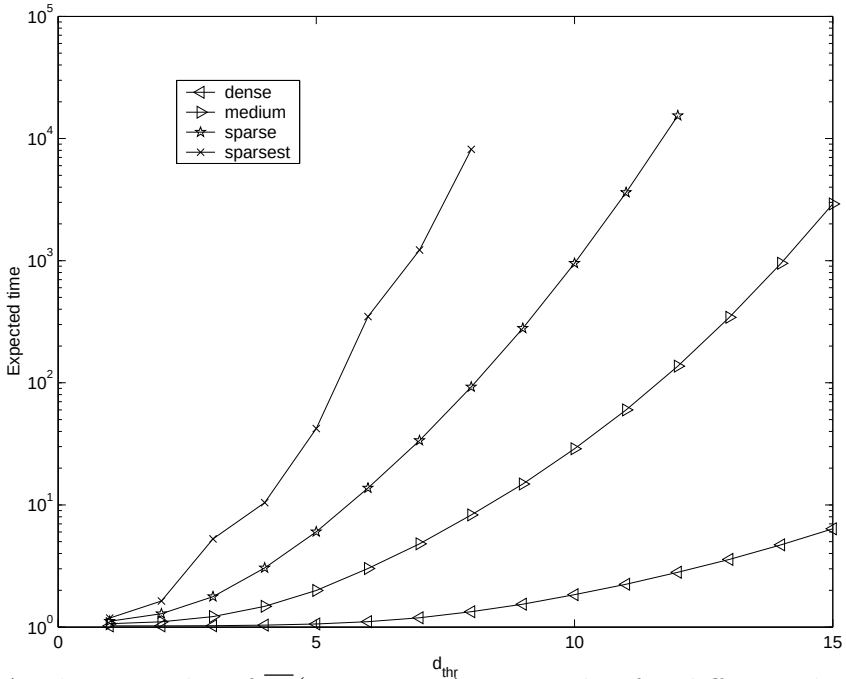

Figure 4.11: Analytic results of $\overline{T_b}(n_{tot}, n_{av}, N)$ versus d_{thr} for different density networks ($nlf f_{thr} = 0.5$)

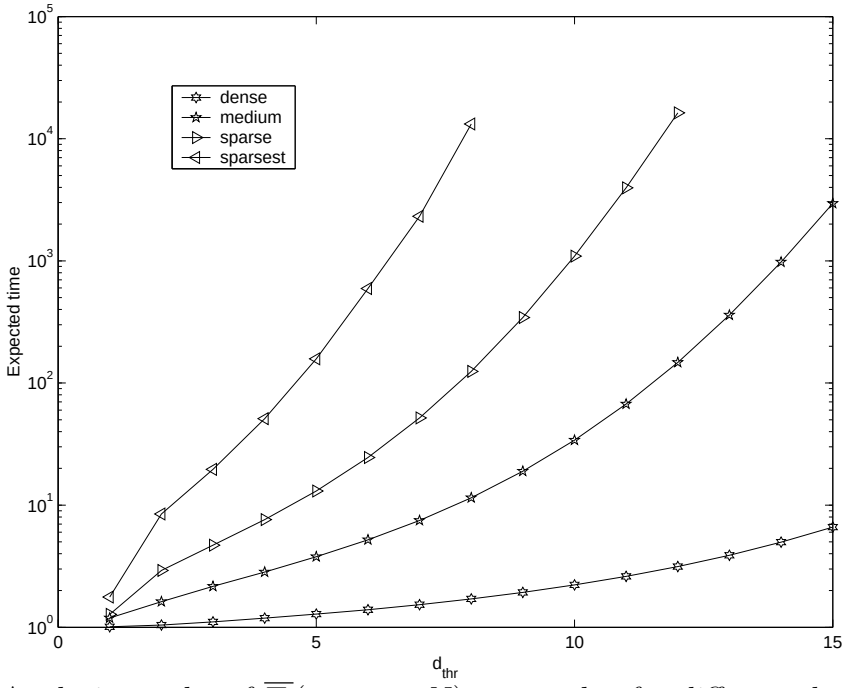

Figure 4.12: Analytic results of $\overline{T_b}(n_{tot}, n_{av}, N)$ versus d_{thr} for different density networks ($nlff_{thr} = 1.0$)

node degrees are less than 7 for the three networks for which the expected times do not decrease exponentially for $7 \leq d_{thr} \leq 15$.

An interesting observation is that the expected first times for becoming *white* and *black* for a given network are close when d_{thr} is around the mean node degree. In Fig. 4.8, the expected time for becoming white (black) in *dense* network is represented by `dense-w` (`dense-b`). Similarly, for *medium density* network, the expected time for becoming white (black) is represented by `medium-w` (`medium-b`). The mean node degrees are 13.5 and 6.0221 for *dense* and *medium density* networks, respectively and T_b and T_w have almost the same values when d_{thr} is 14 in *dense* network and 6 in *medium density* network. Another observation from Fig. 4.8 is that it takes almost the same number of steps to

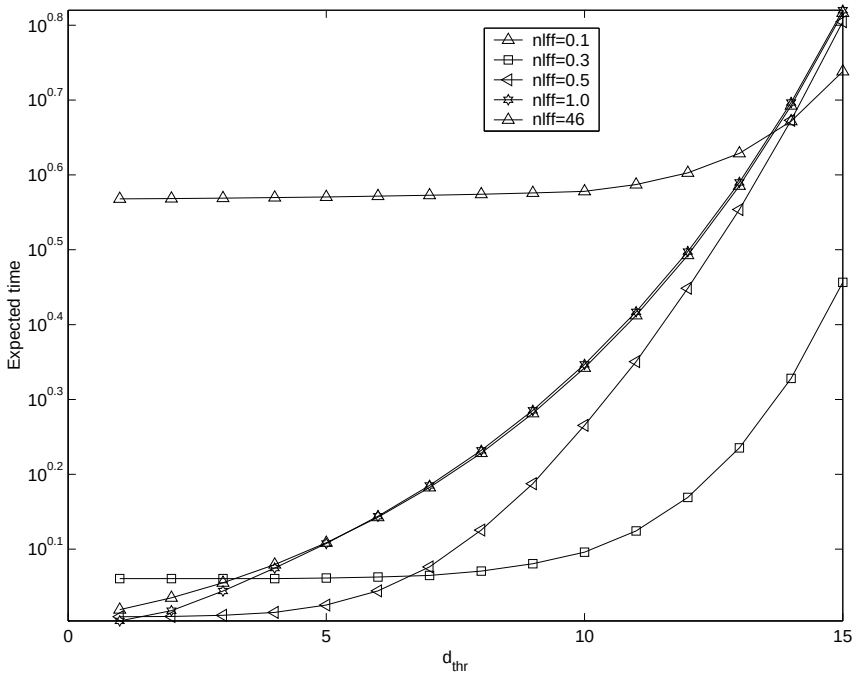

Figure 4.13: Numerical results of $\overline{T_b}(n_{tot}, n_{av}, N)$ for different d_{thr} and $nlff_{thr}$ values in *dense* network

reach the mean number of active links for both *dense* and *medium density* networks.

We conducted preliminary simulation experiments for virtual backbone (VB) formation and maintenance algorithms described in Section 4.3.1. The initial results showed that the expected number of neighbors in a VB node is higher than the expected number of nodes in nonVB node. Under this assumption, our analytic model predicts that the expected times will be close for a VB (nonVB) node to become a nonVB (VB) (i.e., two adjacent nodes cannot be VB at the same time, and hence one should become a nonVB node).

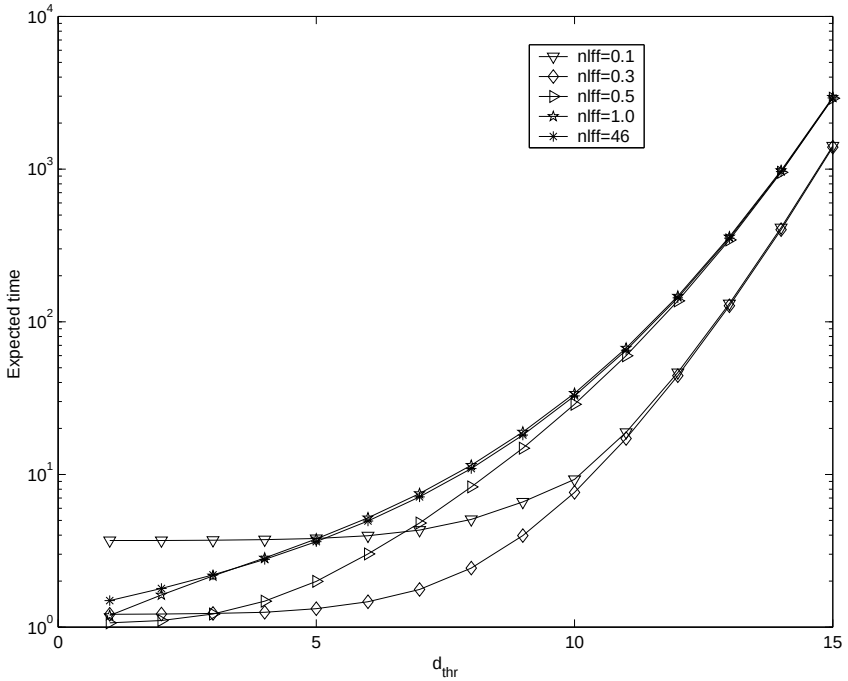

Figure 4.14: Numerical results of $\overline{T_b}(n_{tot}, n_{av}, N)$ for different d_{thr} and $nlff_{thr}$ values in *medium density* network

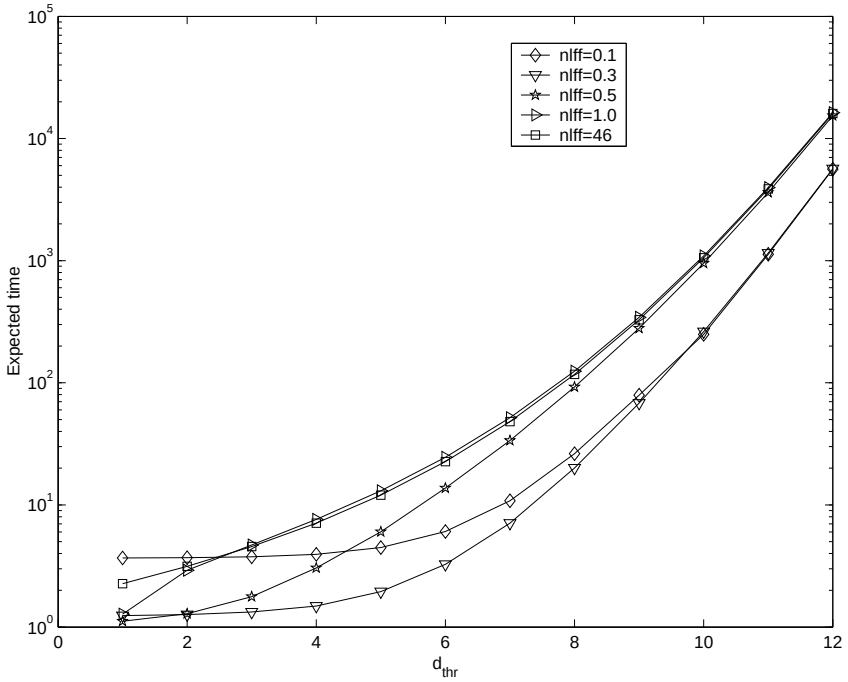

Figure 4.15: Numerical results of $\overline{T_b}(n_{tot}, n_{av}, N)$ for different d_{thr} and $nlff_{thr}$ values in *sparse* network

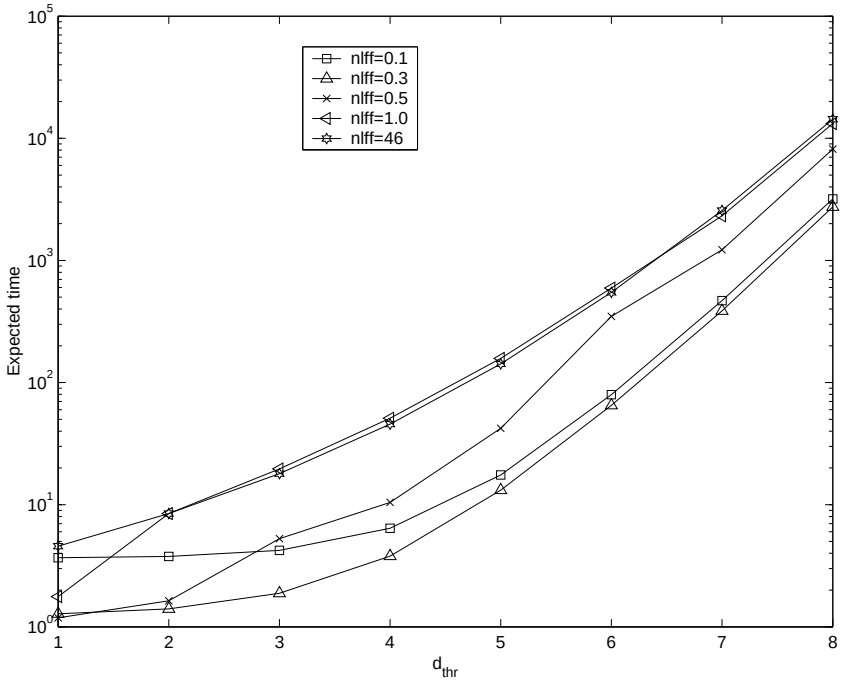

Figure 4.16: Numerical results of $\overline{T_b}(n_{tot}, n_{av}, N)$ for different d_{thr} and $nlff_{thr}$ values in *sparsest* network

4.4.3 Effects of *nlff*

In these experiments, in addition to d_{thr} and the network density, the *nlff* value is also varied to observe their combined effects over the expected first passage times. Table 4.3 presents the number of white and black states for different d_{thr} and $nlff_{thr}$ values for N=46.

In Figs. 4.9, 4.10, 4.11, and 4.12, the numerical results of the expected first passage times for different d_{thr} and four different network densities are presented for $nlff_{thr}$=0.1, 0.3, 0.5, and 1.0, respectively. We observe in Fig. 4.9 that, for $nlff_{thr}$=0.1, the expected passage times are almost the same for different network densities when $d_{thr} < 3$. In other

words, the small value of $nlff_{thr}$=0.1 has little effect on the first passage times since the number of black states will be small for different network densities. The same observation is evident from Table 4.3 for the row corresponding to $nlff_{thr}$=0.1 (e.g., the number of white and black states are 962 and 119 for d_{thr}=1, respectively).

As d_{thr} increases, we observe that sparser networks have higher expected first passage times compared to denser networks because sparse networks have smaller probability for a new link arrival (i.e., P_{ua}) than denser networks. A similar behavior can be observed for different $nlff_{thr}$ values shown in Figs. 4.10, 4.11, and 4.12.

Let us now consider the effects of increasing $nlff_{thr}$ values for different d_{thr} and network densities in Figs. 4.13, 4.14, 4.15, and 4.16. Except for *sparsest* network in Fig. 4.16, the expected first passage times have the highest values for $nlff_{thr}$=0.1 and $d_{thr} \leq 2$ in all networks. This is because in these cases the number of black states whose node degree is close to the mean node degree is less than the number of black states in *sparsest* network (recall that each *nlff* value is defined as the number of link failures normalized by its node degree). This observation is also supported by the fact that the mean node degree for *sparsest* network $\overline{N}(30, 5, 46)$ has a very small value of 1.51.

As d_{thr} increases to the mean node degree of the corresponding network, the effect of $nlff_{thr}$ becomes more evident. After d_{thr} becomes greater than the mean node degree, the expected passage times have the smallest values for $nlff_{thr}$=0.1 and the highest for $nlff_{thr}$=46. For example, in Fig. 4.14, where the mean node degree is 6.0221 from Ta-

Table 4.3: Number of white and black states for different d_{thr} and $nlff_{thr}$ values, (n_w,n_b)

	d_{thr}=1	d_{thr}=2	d_{thr}=3	d_{thr}=4	d_{thr}=5	d_{thr}=10	d_{thr}=15
$nlff_{thr}$=0.1	(962,119)	(963,118)	(964,117)	(965,116)	(966,115)	(971,110)	(981,100)
$nlff_{thr}$=0.3	(818,263)	(819,262)	(820,261)	(821,260)	(823,258)	(836,245)	(857,224)
$nlff_{thr}$=0.5	(706,375)	(707,374)	(709,372)	(711,370)	(714,367)	(735,346)	(769,312)
$nlff_{thr}$=1.0	(530,551)	(532,549)	(535,546)	(539,542)	(544,537)	(584,497)	(649,432)
$nlff_{thr}$=46	(46,1035)	(91,990)	(135,946)	(178,903)	(220,861)	(415,666)	(585,496)

ble 4.2, we observe this behavior when $d_{thr} > 6$.

4.5 Simulation Results

In our simulations, we place a mobile node on the center cell (0,0) while the remaining 45 nodes are distributed randomly on a geographic area partitioned into logical hexagonal cells. Each mobile node roams randomly following the same discrete-time random walk model described in Section 3.1. First, the mobile node located on the center cell (0,0) is roamed randomly into one of its neighboring cells and the coordinates of this new cell is set to (0,0), making the new cell as the new center cell. The remaining nodes roam based on the random walk model. Note that if a node is close to the geographic boundaries (i.e., its distance from the center cell is n_{tot}), there are less number of possible neighboring cells due to bouncing back described in Fig. 4.1.

In the simulation results, the mean node degree for a mobile node located on the center

cell (0,0) is observed and reported for 2×10^6 time units. The mean node degrees obtained from the simulation experiments are 12.33, 5.68, 3.26, and 1.47, while the corresponding mean node degrees obtained from our analytic model are 13.50, 6.02, 3.39 and 1.51 for *dense*, *medium density*, *sparse* and *sparsest* networks, respectively. These simulation results verify that our analytic models correctly capture the MANET behavior such that the mean node degree decreases as the network density increases.

Using the same network scenarios, we measured the expected first times to reach different d_{thr} values. The sole effects of d_{thr} over the passage times, without considering $nlff_{thr}$, are observed. For a node located on the center cell, the intervals starting when this node has a degree less than d_{thr} and ending when its degree is equal to or higher than d_{thr} for the first time are measured for 2×10^6 time units. We then averaged these intervals to find the expected first passage times. Fig. 4.17 shows the results measured from simulation experiments and calculated from analytic model for *medium density* and *sparse* networks. The simulation results support the numerical calculations from our model especially around the mean node degree, where the expected first time increases with d_{thr} in both cases.

The main difference between the numerical and simulation results stems from our approximation that, for all available links, we calculate only two probabilities namely, the probability of remaining available and becoming unavailable in the next time unit. This approximation was necessary to construct $P2$ with feasible number of states.

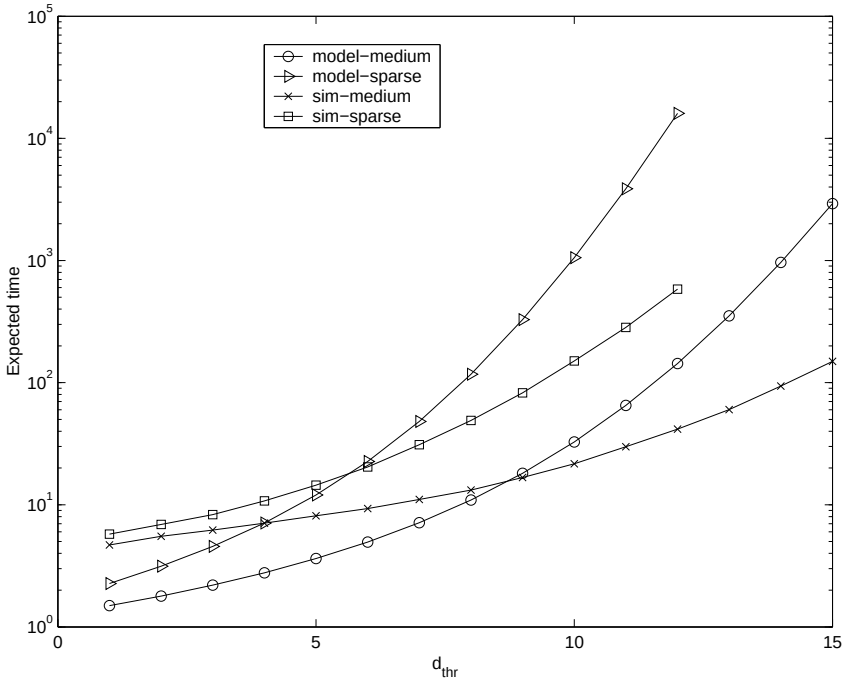

Figure 4.17: Analytic and simulation results for the expected times vs. d_{thr} in the medium density network.

The difference between the numerical and simulation results increases when d_{thr} is greater than 10 for *medium density* network and 7 for *sparse* network; however, having a node degree greater than 10 for *medium density* network and 7 for *sparse* network is not typical since their mean degrees are 6.02 and 3.39, respectively. Therefore, the d_{thr} range which our analytic model accurately predicts the expected first times is around the mean node degree. Note that the degrees which are very small or large compared to the mean node degree are typically less of an interest for network designers.

Chapter 5

Application to Controlled Dissemination Filter

The Proactive Integrated Link Selection for Network Robustness (PILSNER) program at Telcordia Technologies, Inc.[1] develops agent technologies that support automatic link selection over different and widely varying transmission paths to bypass network congestion and outages. To perform optimizations as the network changes dynamically, agents need to rely on information that is dispersed in the network. The *Integrated Link Selection Agent* (ILSA) architecture proposes that the data collected by the agents be published to the nodal information service in the CougaarNode agent, which allows the local agents

[1]I was a graduate co-op at Telcordia Technologies, Inc. when this work is done.

to share information organized according to a set of services: topology data, router data, needline data, radio data, and policy data. The *Controlled Dissemination Filter* (CDF) resides between the CougaarNode agent and the (future) WIN-T data distribution service and provides an efficient data dissemination service [56, 57] for the ILSA agents. The problem of optimal data dissemination can be divided into three main subproblems:

1. **Data grouping** is to combine the data types into a set of logical data groups (e.g., one group may be a link utilization data for a specific router). The challenge is to determine the granularity and hence the number of diverse data groups to minimize the overhead of data distribution.

2. **Channelization** is to find an optimal mapping of flows and data recipients to the set of data distribution groups (e.g., multicast trees) for a set of information flows and a fixed number of data distribution groups.

3. **Filter placement** is to define an optimal placement of filters at both the end nodes and the intermediate nodes of a distribution tree given the sets of distribution trees, each containing all members of a particular distribution group, and the filtering rules at the source and the destinations.

One can interpret the above problems as finding an optimum between two nave approaches. The first approach would be to create a single data group, say *CDF data*, build a multicast tree that includes all the potential recipients in the group, and then perform

all data filtering at end points. This would result in unnecessary data dissemination overhead and hence wasted bandwidth. The second approach is to create a multicast tree for every data type, which would lead to better bandwidth utilization, but would not scale due to complexity and overhead of maintaining a large number of multicast groups. The CDF uses both channelization and data filtering methods as complementary tools for obtaining maximally efficient data dissemination mechanism which can be adapted to application requirements.

In this study, new objective functions for the channelization approximation algorithms are proposed, and both centralized and distributed sub-optimal and optimal algorithms to perform filter placement in a polynomial time are developed. The CDF enhances a data-dissemination transport service by offering an efficient filtering functionality such as sifting, collection, aggregation, and scoping. Sifting, collection, and aggregation reduce the amount of data that gets published and disseminated in the network, whereas scoping function limits the set of nodes that receive a particular data object. All of the above data filtering transformations can be applied to the same data in sequence, e.g., it should be possible to disseminate only every fifth sample from an aggregated link-utilization for an interface on the router, to achieve a flexible filtering capability. We also present the CDF applications to the PILSNER Dynamic Domain Optimization Agent (DDOA) [40] and Unicast Routing Control Agent (URCA) [35] and outline a path towards application of Dynamic Survivable Resource Pooling (DSRP) [36] to further improve the CDF.

5.1 Channelization

The channelization problems are NP-hard [1], and thus solvable only through an exhaustive search or sub-optimal heuristics. The algorithms take as input a set of information flows S originating from a source, a set of nodes U with CDF instances, a set of multicast groups G, and the relevance matrix $W = (w_{ij})$, where flow $i \in S$ and user $j \in U$. The output of these algorithms is the flow-to-group mapping defined as matrix $X = (x_{im})$, where $i \in S$ and $m \in G$; and the user-to-group mapping defined as matrix $Y = (y_{jm})$, where $j \in U$ and $m \in G$.

Input:

1. A set of information flows S at the source. An information flow is defined on a per node per filter basis (i.e., multiple data objects that share the same filter can be grouped together as one flow). To execute the algorithm, the CDF agent needs the sets of flows at this source as vectors of flow information rates $(\lambda_1, \lambda_2, \ldots, \lambda_K)$ and the corresponding scoping filters $(F_{s1}, F_{s2}, \ldots, F_{sK})$.

2. A set of nodes U interested in a set of flows at the source.

3. A set of multicast groups G.

4. Network topology information per OSPF area. We assume that the full connectivity matrix $T(R, L)$ available at the source.

Output:

1. *Channelization problem:* Assignment of each information flow to one or more multicast groups. Information flows are distributed through multicast groups, with each flow being assigned to one or more multicast groups. The flow-to-group mapping is defined as matrix $X = (x_{im})$, where $i \in S$ and $m \in G$. If flow i is assigned to group m, $x_{im} = 1$; $x_{im} = 0$ otherwise.

2. *Subscription subproblem:* Assignment of each end user (i.e., the destination node for a data flow) to one or more multicast groups. The user-to-group mapping is defined as matrix $Y = (y_{jm})$, where $j \in U$ and $m \in G$. If a destination node j is assigned to group m, $y_{jm} = 1$; $y_{jm} = 0$ otherwise.

Algorithm:

1. Based on the scoping filters and topology information, derive the relevance matrix $W = (w_{ij})$, where $i \in S$ and $j \in U$. If a node $u \in U$ is among the destinations for i, $w_{ij} = 1$; $w_{ij} = 0$ otherwise.

2. Define the objective function as:

$$C(X, Y) = w_1 \sum_{i \in S} \sum_{j \in U} \sum_{m \in G} x_{im} y_{jm} d_{jm} \lambda_i + w_2 \sum_{m \in G} \sum_{i \in S} x_{im} \lambda_i \tag{5.1}$$

where w_1 and w_2 are weights such that $w_1 + w_2 = 1$. The cost associated with assigning user j to group m is proportional to the aggregate volume of CDF traffic

that j will receive and the minimum distance d_{jm} to other members of the group:

$$d_{jm} = \min_{k \neq j \wedge y_{km}=1} SP(n_j, n_k) \tag{5.2}$$

Alternatively, d_{jm} can be defined as the average distance d_m from all users in multicast group m to the source s.

$$d_{jm} = d_m = (\sum_{k:y_{km}=1} SP(s, n_k))/|m| \tag{5.3}$$

The *minimum-inclusion* requirement is defined as minimizing $C(X, Y)$ whereas the *no-false-exclusion* requirement, for each $i \in S$ and $j \in U$, corresponds to the following constraint:

$$\sum_{m \in G} x_{im} \times y_{jm} \geq w_{ij} \tag{5.4}$$

3. Choose the constrained or unconstrained version of mapping flows to multicast groups. The unconstrained version allows each flow to be assigned to multiple multicast groups and the constrained version requires that the multicast groups form a partition of all the information flows, i.e., that each flow be assigned to only one multicast group.

4. The constrained subscription problem can be solved in linear time. To solve other instances of these problems (unconstrained subscription, both constrained and unconstrained channelizations), which are NP-hard, apply one of the heuristic procedures defined in the literature (e.g., Adler et al. [1]).

The choice of using constrained or unconstrained mappings is determined by the nature of a problem or by considering a design tradeoff involving issues of system complexity, efficiency, flexibility, requirements of data consistency, etc. For example, using the constrained version of flow-to-group mapping instead of the unconstrained version may sacrifice system efficiency. This would happen if, in the unconstrained version of the channelization problem, the optimal configuration required that one flow be assigned to multiple multicast groups.

To illustrate this issue, consider three information flows s_1, s_2, and s_3 that are being disseminated to users u_1 and u_2 through multicast groups g_1 and g_2. The rates are $\lambda_1 = 1$ and $\lambda_2 = \lambda_3 = 100$. User u_1 is supposed to receive flows s_1 and s_2, whereas user u_2 is to receive flows s_1 and s_3. In the unconstrained problem, the optimal solution is $g_1 = s_1, s_2$ and $g_2 = s_1, s_3$. Thus, u_1 and u_2 need to be assigned only to g_1 and g_2, respectively. Both users receive only the relevant data with no need for end-point filtering. In the constrained problem, however, the best solution is $g_1 = s_1, s_2$ and $g_2 = s_3$. User u_1 will still receive only the relevant data, but u_2 will receive unwanted traffic at the rate of 100, which have to be filtered out.

To solve the channelization problem, we formulate several versions of the objective functions and the constraints defined above.

1. ***Unconstrained Channelization Problem:*** It uses the original objective func-

tion and the constraints defined above.

2. ***Constrained Channelization Problem:*** The weights w_1 and w_2 are set to 1 and 0, respectively, and Y is defined as a function of X and W, to obtain the objective function:

$$C(X,Y) = \sum_{i \in S} \sum_{j \in U} \sum_{m \in G} x_{im} y_{jm} (x_{im}, w_{ji}) d_{jm} \lambda_i \tag{5.5}$$

The value of y_{jm} becomes 1 when user j is interested in flow i that belongs to multicast group m (i.e., $w_{ji} = 1$ and $x_{im} = 1$). We also add the following constraint:

$$(\forall i \in S) \sum_{m \in G} x_{im} = 1 \tag{5.6}$$

3. ***Unconstrained Subscription Problem:*** Matrix X is obtained separately by using some heuristic methods, and the subscription is solved for each user j individually to further simplify the objective function in Eq. 5.5:

$$(\forall j \in U) C(Y) = \sum_{i \in S} \sum_{m \in G} x_{im} y_{jm} d_{jm} \lambda_i \tag{5.7}$$

4. ***Constrained Subscription Problem:*** The problem is defined similar to the unconstrained version except that we add one more constraint:

$$(\forall i \in S) \sum_{m \in G} x_{im} = 1 \tag{5.8}$$

5.2 Filter Placement

The filter-placement algorithms require processing and reconciliation of filtering rules in all nodes in a tree as well as explicit signaling among the nodes in the tree. The generic version of the distributed filter placement algorithm, which runs at each node v of a multicast tree, is described below.

Input:

1. The parent p and the children of a node in the multicast tree $M = (V; E)$, with vertex set V, edge set $E \subseteq VxV$, and an information flow i disseminated by the source CDF s assigned to this multicast group. The set V includes all nodes U assigned to the multicast group plus the intermediate nodes in M.

2. For each $v \in V$, sets of filters $F_t(v)$ and $F_r(v)$ that will be applied at v to transit (i.e., destined for v's children nodes) and received (i.e., destined for v) traffic, respectively. Initially, for each $v \in V-U$, $F_r(v) = \emptyset$; and, for each $v \in V-s$, $F_t(v) = \emptyset$. $F_{ra}(v)$ and $F_{ta}(v)$ represent the active filter rules for received and transit traffic, respectively. Sets $F_{ta}(v)$ and $F_{ra}(v)$ are initially set to $\emptyset$ for all nodes.

Output:

1. For each $v \in V$, active sets of filters $F_{ta}(v)$ and $F_{ra}(v)$

```
findFilterRule (at node v)

When F_t(s) received from the parent p
    if v ∈ U
      Find F_r(v) as the most restrictive superset of F_r(v) ∪ F_t(s)
      F = F_r(v)

When the updated F_r(u) and F_t(u) are received from u=child(v)
    if timer T_child not running, start T_child
    Add F_r(u) ∩ F_t(u) to F

When T_child expires
    Find the least restrictive subset of F_t(v)=F_t(v) ∩ F
    Send an updated F_t(v) to the parent node
    Send an updated F_t(v) to the children

When F_t(p) received from the parent p
    F_ra(v) = F_r(v) − F_t(p)
    F_ta(v) = F_t(v) − F_t(p)

When any rules are changed locally at v
    Add new rule F_r(v) to F
    Find the least restrictive subset of F_t(v) = F_t(v) ∩ F
    Send an updated F_t(v) to the parent node
    Send an updated F_t(v) to the children
```

Figure 5.1: Pseudocode for `findFilterRule` function at node v

Algorithm:

The distributed version of the placement algorithms is executed per information flow at each node in the tree. The execution at the source occurs any time the channelization algorithm completes (i.e., after mapping the information flow to a multicast group), or any time the filter rules at the source change, where the source will send its new filter

rules toward the downstream nodes; the execution at other nodes is triggered whenever a message with updated filters arrives from the children or the parent nodes. Each node runs the event-driven algorithm that reacts to the following events: message from a child, message from the parent, expiration of timers, and the reconfiguration of filtering rules by the local ILSA agents. The filter placement algorithm shown in Fig. 5.1 calculates the active filter rules at node v and has four main steps:

- wait for messages from all children nodes for a timeout period;
- calculate the least restrictive filter rules among the received filter rules from children;
- send the calculated filter rules to the parent; and
- modify the filter rules upon reception of a message from the parent.

The definitions of the *least restrictive subset* and the *most restrictive superset* in this algorithm depend on the application data flow and its rate requirement (e.g., the least restrictive subset is calculated by the modular operator for the sampling based on the number of data points while it is calculated by the less than ($<$) operator for the deviation based dissemination). The following two sections demonstrate two examples of the filter placement implementations in more details.

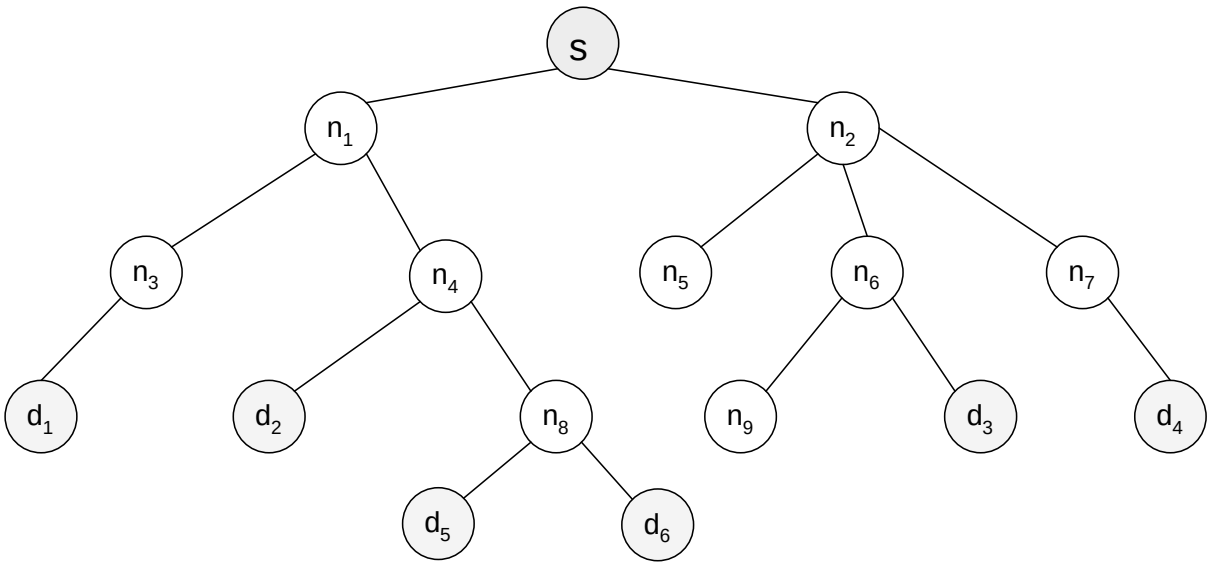

Figure 5.2: A multicast tree for a single source and multiple destinations

5.2.1 Sampling Based on Number of Data Points

We present an example to illustrate how our filter placement algorithm works on the multicast tree depicted in Fig. 5.2, where there are a single source s and multiple destinations $U = d_1, d_2, \ldots, d_6$. Suppose that the source has a collection of the link utilization data and disseminates this data after performing source filtering according to its filter rule, $F_t(s)$. In this case, the filter rule is based on the number of data points, e.g. the dissemination of a single data out of every n data objects. The filter rule *(string, sift, sampling) (integer, increment, 5)* defines dissemination of one out of every 5 data objects; we will use a shorthand notation of $F_t(s) = 5$ to represent this rule.

Initially, the intermediate nodes $n_1, n_2, \ldots$, and n_9 do not have any filter while the destination nodes have their local filter rules, $F_r(v)$ where $v \in U$. Note that local filter rules

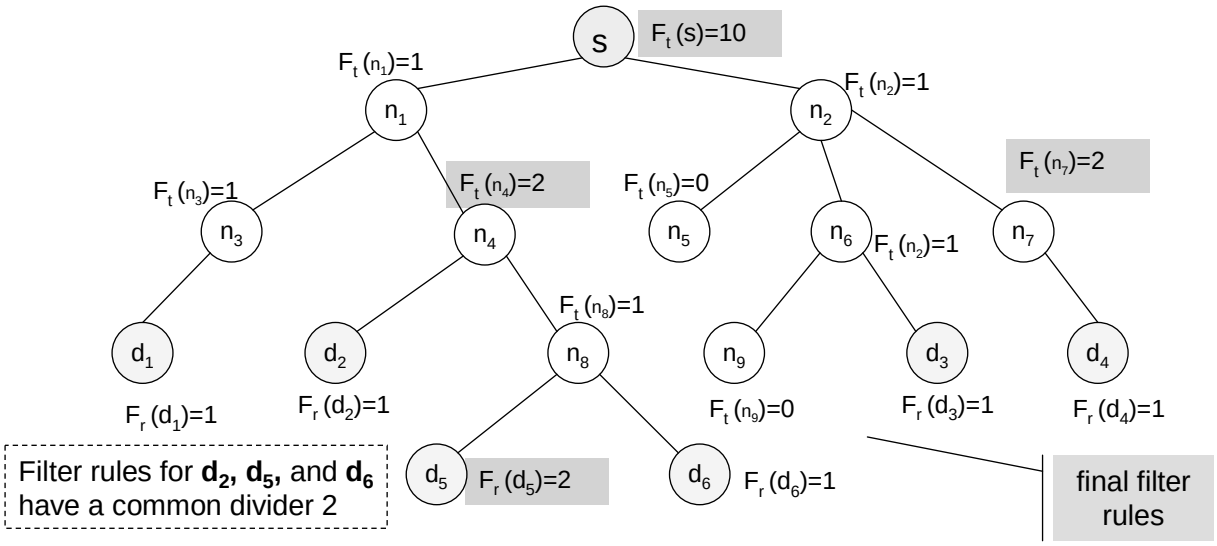

Figure 5.3: Final filter rules for the sampling based on number of data points

should be decided based on the local network conditions (e.g., link utilizations, congestion levels, etc.) together with application rate requirement. At the beginning, the algorithm modifies the filter rules of the destination nodes according to the source filter rule since a destination might ask for more data than the source can able to disseminate (e.g., due to bandwidth limitation). The final goal is to place the filter rules into the intermediate nodes so that unwanted data is filtered out at intermediate nodes of the multicast tree before delivering to the final recipients (e.g., either each receiver node requires different data rate or receiver's data rate requirement is adapted according to local congestion levels and available bandwidth conditions). Once the filter is placed into an intermediate node, this same filter rule will be applied to all the outgoing links of this node (i.e., data is multicasting).

Assume the following filter rules initially: $F_t(s) = 10$, $F_r(d_1) = 6$, $F_r(d_2) = 20$, $F_r(d_3) = 5$,

$F_r(d_4) = 20$, $F_r(d_5) = 40$, and $F_r(d_6) = 20$. After updating $F_r(v)$ as the most restrictive superset of filters $F_r(v) \cup F_t(s)$ for each $v \in V - s$ and applying the greatest common divisor of source and destination filter rules, the updated filter rules at the destination nodes become: $F_t(s) = 10$, $F_r(d_1) = 1$, $F_r(d_2) = 2$, $F_r(d_3) = 1$, $F_r(d_4) = 2$, $F_r(d_5) = 4$, and $F_r(d_6) = 2$. At this stage, the source does filtering before sending the data; therefore, the destination nodes update their filter rules by considering the source filter rule. The greatest common divisor of $F_t(s) = 10$ and $F_r(d_2) = 20$ is 2, therefore the updated filter rule for (d_2) becomes 2. The initial filter rule for (d_1) is 6 which is less than the source filter rule $F_t(s) = 10$, therefore $F_r(d_1)$ is set to 10 according to the most restrictive superset of filters $F_r(d_1) \cup F_t(s)$. After applying the greatest common divisor operator, the updated filter rule becomes 1 for d_1.

At the next stage the destination nodes propagate the filter rules toward the source; meanwhile the intermediate nodes collect the propagated rules to decide their own rules, which are further forwarded toward the source, recursively. After the first iteration: $F_t(n_8) = 2$, $F_r(d_5) = 2$, and $F_r(d_6) = 1$. After the second iteration: $F_t(n_3) = 1$, $F_t(n_4) = 2$, $F_t(n_8) = 1$, $F_r(d_2) = 1$, $F_t(n_6) = 1$, and $F_t(n_7) = 2$, $F_r(d_4) = 1$. After the third iteration: $F_t(n_1) = 1$ and $F_t(n_2) = 1$. The final filter rules for all the nodes in the multicast tree are shown in Fig. 5.3.

Using the final filter rules in the multicast tree shown in Fig. 5.3, a source s disseminates 100 data objects to the destination nodes and the total bandwidth usage is calculated for

three cases:

1. only source does filtering: **150**;
2. all nodes in the tree run the filter placement and reduction algorithm and the same filter is applied to all outgoing links of a particular node: **125**;
3. optimum case where destinations receive only the required data objects (different filters for each child in the tree) after source filtering performed: **92.5**.

We assume that each data object consumes one unit of bandwidth when it is forwarded to the next hop. The total bandwidth usage is 150 when only source does filtering whereas it is 125 when our filter placement algorithm is used. When a different filter rule can be applied for each child node in the tree, the total bandwidth usage can be further decreased to 92.5 but the multicasting can not be used in this case (i.e., data should be unicasted to each child node). Note that the amount of the CDF gain depends on the final filter rules and the size of the multicast tree, where the final filter rules depend on initial filter rules and the parent-children relations within the multicast tree. We propose methods for how to set local filter rules based on the triplets of local congestion levels, available bandwidths and application requirements for both the DDOA agent in Section 5.3 and the URCA agent in Section 5.5. The methods for setting local filter rules will likely be different for other agents depending on their applications' data rate requirements.

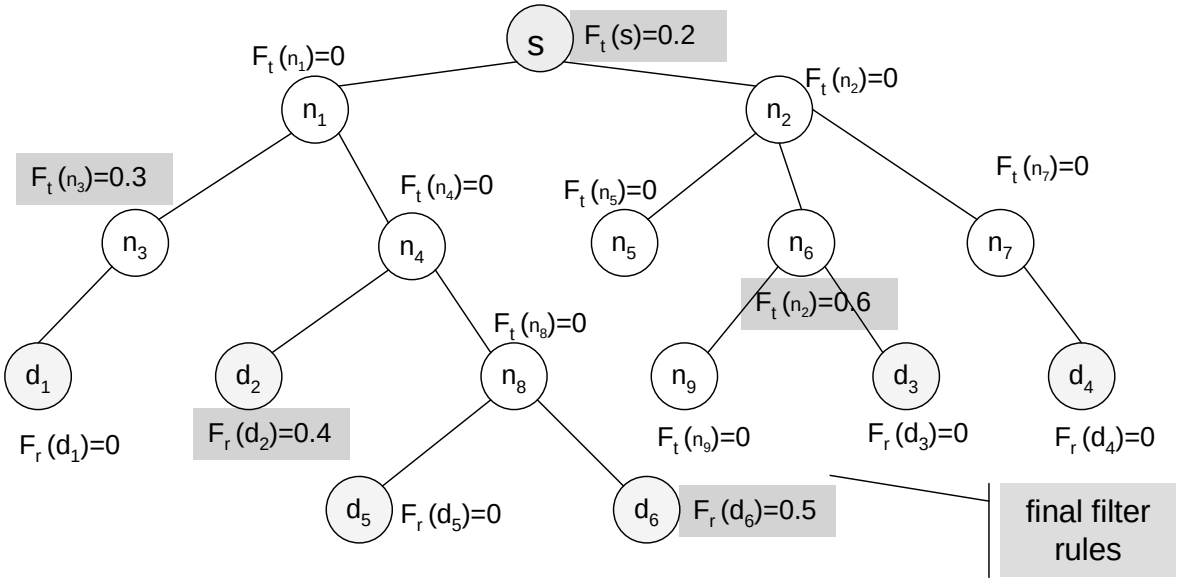

Figure 5.4: Final filter rules for the sampling based on deviation from the mean

5.2.2 Sampling Based on Deviation from Mean

The multicast tree shown in Fig. 5.2 is used to illustrate how our filter placement algorithm works for the deviation-based data dissemination from a single source s to multiple destinations $U = d_1, d_2, \ldots, d_6$. Suppose that the source has a collection of the link utilization data and disseminates this data using the deviation-based filter rule. The extra filtering will be performed at the intermediate nodes before the data items are delivered to their destination. In this case, the filter rule is deviation-based, e.g. the dissemination of data when the value of the data item is different than the average data value by at least a certain deviation constant. The filter rule *(string, sift, sampling) (float, delta, 0.2)* defines dissemination of a data if the link utilization deviates from the average link utilization by at least 0.2; we use a shorthand notation of $F_t(s) = 0.2$ to represent this

rule.

Assume the following filter rules initially: $F_t(s) = 0.2$, $F_r(d_1) = 0.3$, $F_r(d_2) = 0.4$, $F_r(d_3) = 0.6$, $F_r(d_4) = 0.2$, $F_r(d_5) = 0.1$, and $F_r(d_6) = 0.5$. After updating $F_r(v)$ as the most restrictive superset of filters $F_r(v) \cup F_t(s)$ for each $v \in V - s$, the filter rules at the destination nodes become as follows: $F_t(s) = 0.5$, $F_r(d_1) = 0.3$, $F_r(d_2) = 0.4$, $F_r(d_3) = 0.6$, $F_r(d_4) = 0.2$, $F_r(d_5) = 0.2$, and $F_r(d_6) = 0.5$. Note that $F_r(d_5)$ was 0.1 before calculating the most restrictive superset of filters $F_r(d_5) \cup F_t(s)$. This is not possible since the source node filters out a data sample whose deviation from the mean value is less than 0.2 due to its available bandwidth conditions. Therefore, the updated filter rule at node d_5 should be at least 0.2. The final filter rules for all the nodes in the multicast tree are shown in Fig. 5.4.

5.3 CDF Application to DDOA

The PILSNER program at Telcordia Technologies, Inc. proposes ILSA Dynamic Domain Optimization Agent (DDOA) [40] to partition a large size network into smaller routing domains for the scalability reason. Fig. 5.5 shows an example partition of a flat network into OSPF areas by DDOA. There are five areas after the partitioning, where the Area 0 is the backbone connecting the other areas with each other. For example, Area 1 includes nodes 1, 2, 3, 4, 5, and 6 where nodes 5 and 6 are Area Border Router (ABR)

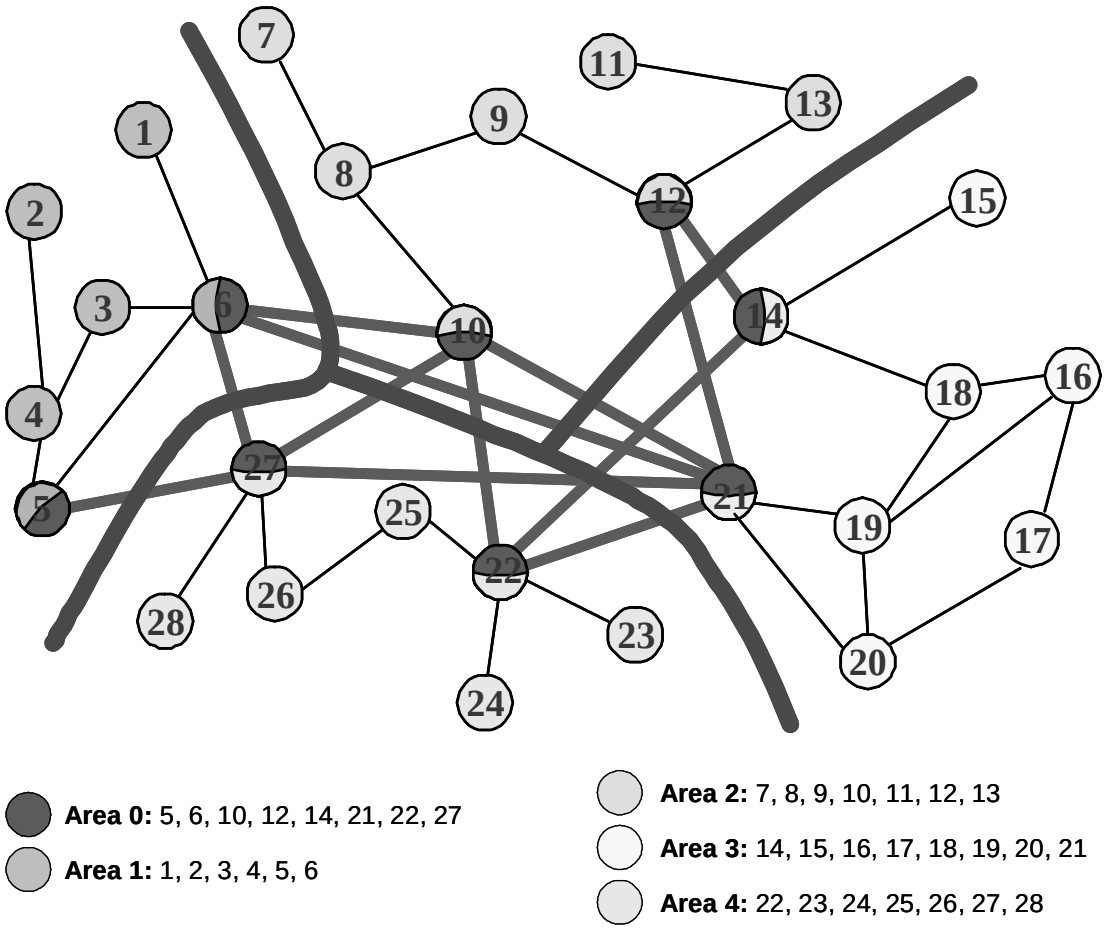

Figure 5.5: An example partition of a flat network into areas by DDOA agent (taken from the DDOA design document of the PILSNER program at Telcordia Technologies, Inc.)

nodes. Assume that node 6 is the primary ABR for the Area 1. We can classify the data dissemination to be performed by the CDF into two types: intra-area and inter-area dissemination. We will explain the procedure with example scenarios for the former type of data dissemination in more detail below. The latter case will be similar to the intra-area data dissemination in the sense that ABR nodes perform dissemination among themselves using the Area 0 (i.e., backbone area). For example, for inter-area traffic from node 2 in the Area 1 to node 17 in the Area 3, node 2 first delivers its data to node 6 (i.e., ABR node for the Area 1) using the intra-area dissemination mechanism within the Area 1. And then node 6 will deliver this data to node 21 (i.e., ABR node for the Area 3) using the intra-area dissemination within the Area 0. Finally, node 21 will deliver the data to its destination node 17 using the intra-area dissemination within the Area 3.

5.3.1 Dissemination Inside an Area

We further divide the data dissemination taking place inside the area for the DDOA application into two categories: *(i)* dissemination from all nodes to all other nodes in the area (*all-to-all dissemination*), and *(ii)* dissemination from all nodes within the area to only primary ABR of this area(*all-to-one dissemination*). As an example of the former type, each node should disseminate through the CDF its ABR priority and a flag indicating whether or not the node serves as an ABR (*periodic dissemination*). This data needs to be received by all nodes within the area since each node asynchronously runs the DDOA

algorithm to decide its ABR role. As an example of the latter type, in the case where a partitioning algorithm uses link loading as the metric to minimize, each DDOA agent disseminates its traffic vector to the area lead, from which the lead DDOA assembles a traffic matrix (*deviation-based dissemination*).

5.3.2 All-to-one Dissemination

In this case, all nodes inside a DDOA area (apart from primary ABR node) need to disseminate their traffic matrix to the primary ABR node. A data-distribution tree, whose root is the primary ABR, is formed such that all nodes within the DDOA area are members of this tree. The deviation-based dissemination functionality of the CDF is used for an efficient data dissemination of traffic vector of each node to the primary ABR node. Each node maintains the current and mean traffic rate per destination node and adapts their filter rules based on their local traffic and available bandwidth conditions. The distributed filter placement algorithm running in each node yields the final filter rule to be applied to the transit traffic in this particular node. When and if the deviation between the current traffic rate and the mean traffic rate is less than the active filter rule (i.e., the filter rule is the deviation constant), this data will be filtered out at this node.

Suppose that the node n_2 disseminates its traffic vector to the primary ABR node n_6 in Fig. 5.5. For example, n_2 has the following initial traffic vector T_2, where the first entry (n_1, 5.03, 4.70) means that the current traffic rate between n_2 and n_1 is 5.03 and the

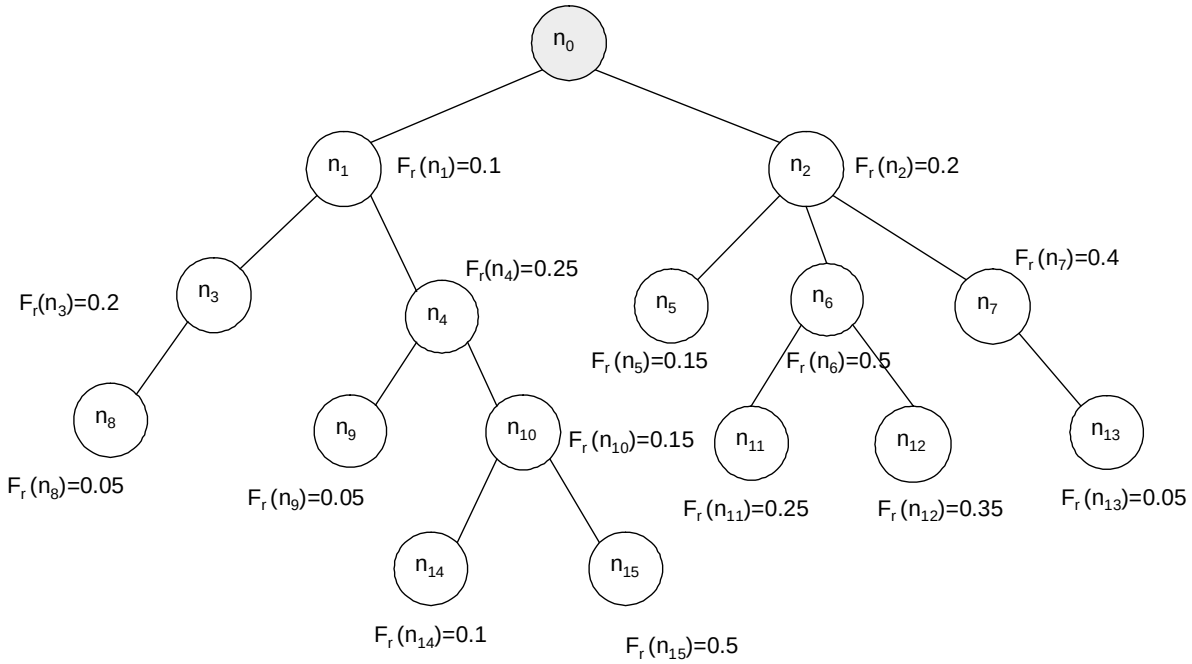

Figure 5.6: Initial filter rules for deviation-based dissemination

mean traffic rate between n_2 and n_1 is 4.70:

$T_2 = [(n_1, 5.03, 4.70), (n_3, 13.65, 13.81), (n_4, 12.23, 11.43), (n_5, 11.34, 12.21), (n_6, 7.21, 6.5),$
$(n_7, 5, 5), (n_8, 5, 5), (n_9, 5, 5), \ldots, (n_{26}, 5, 5), (n_{27}, 5, 5), (n_{28}, 5, 5)]$

T_2 has a separate entry for each node in the other areas. However, the primary ABR only needs the traffic matrix among the nodes within the area and all the inter-area traffic is sent to the primary ABR node; therefore, the node n_2 will aggregate its traffic vector by combining the entries belonging to other areas with the entry belonging to the primary ABR node n_6. After the aggregation is performed, a new vector T_{2new} to be sent to the primary ABR is as follows: $T_{2new} =$
$[(n_1, 5.03, 4.70), (n_3, 13.65, 13.81), (n_4, 12.23, 11.43), (n_5, 11.34, 12.21), (n_6, 117.21, 116.5)]$

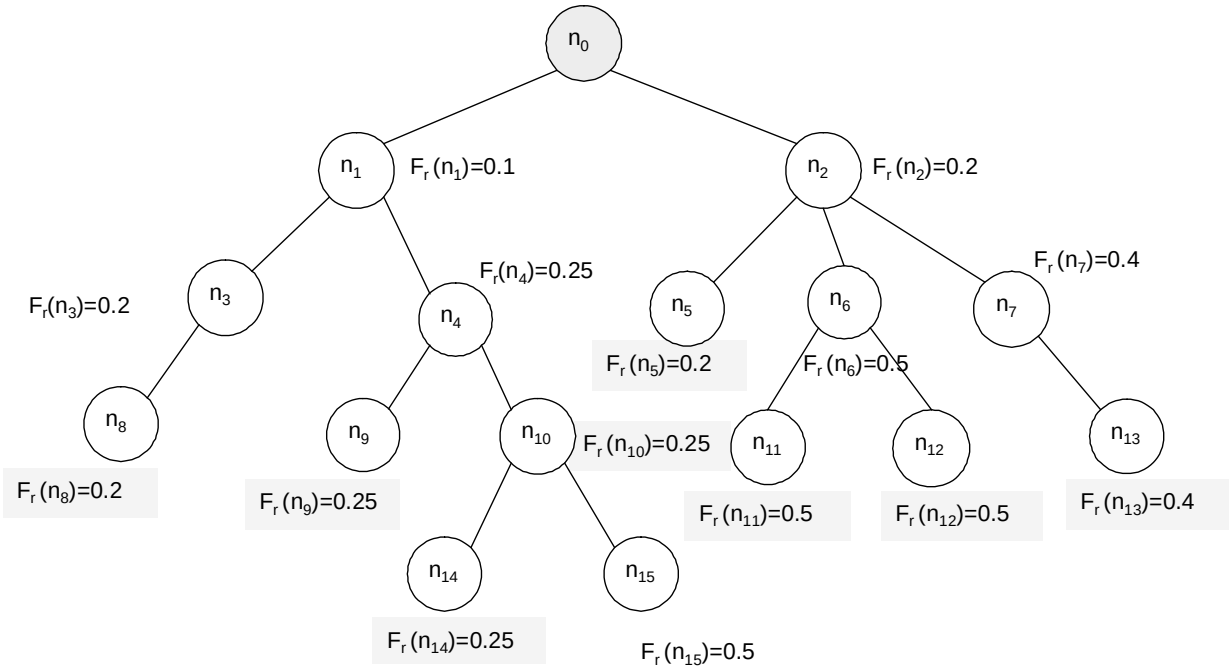

Figure 5.7: Final filter rules for deviation-based dissemination

Fig. 5.6 shows the initial filter rules for deviation-based dissemination obtained through local traffic and bandwidth conditions. In this example, suppose that the node n_{15} has a severe congestion and therefore its filter rule (i.e., deviation constant) is set to 0.5. When and if the deviation between the current traffic rate and the mean traffic rate is higher than 0.5, this data will be disseminated toward the upstream nodes. The higher the deviation constant, the less number of traffic vector samples is disseminated. Each node adapts its filter rule dynamically when its local traffic and available bandwidth conditions change. The distributed filter placement algorithms running at the background continually may converge to a new set of final filter rules whenever a change occurs in one of local filter rules.

The filter placement algorithm calculates the highest filter rule from a data source to the

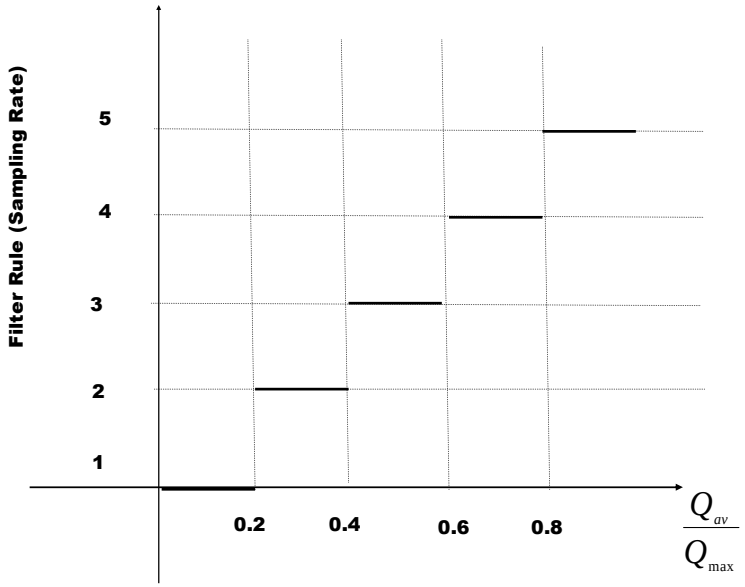

Figure 5.8: Filter rules determined by average queue size

primary ABR node in a distributed way and this filter rule is placed into the data source node. Fig. 5.7 shows the final filter rules calculated in this manner. The motivation is to filter out a data sample, whose deviation is less than the active filter rule of the data source (i.e., the highest deviation constant from a data source towards the primary ABR), because otherwise this sample will be filtered out at one of intermediate nodes due to their congestion levels (e.g., the case which the highest local filter rule is at one of intermediate nodes rather than the data source). Filtering this sample at the data source will prevent unnecessary bandwidth consumption which occurs during the delivery of this sample to the intermediate node which has a higher local filter rule than the data source (i.e., unnecessary bandwidth consumption when we do not use the CDF).

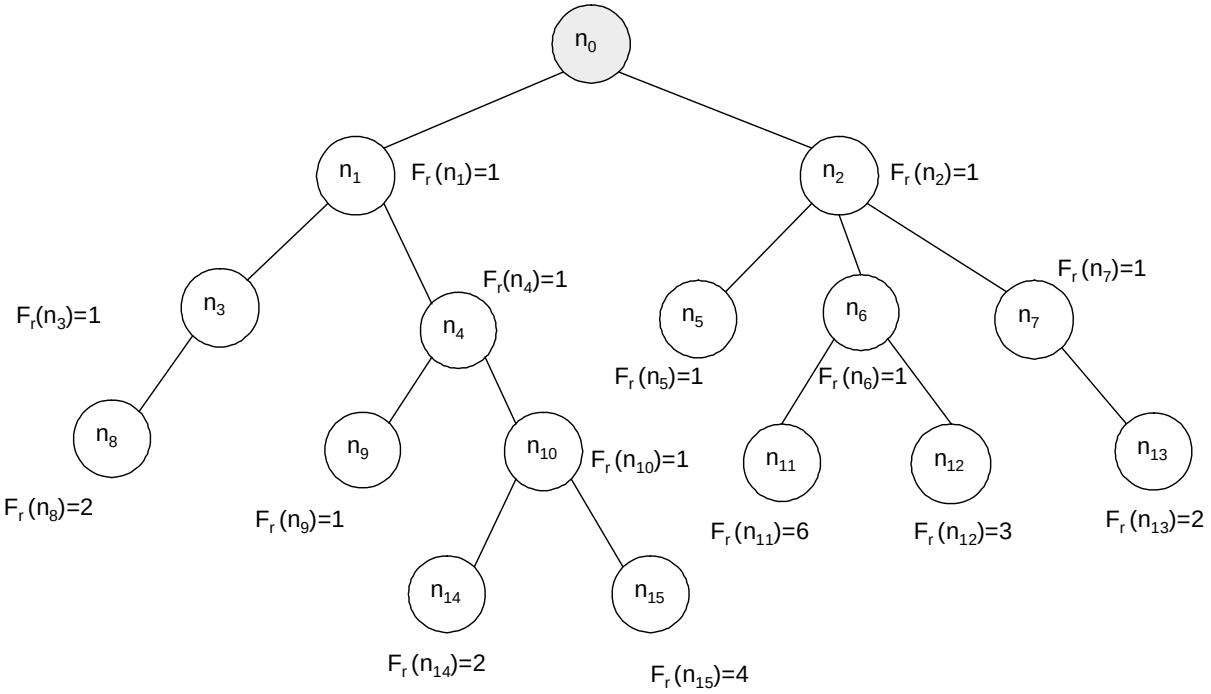

Figure 5.9: Initial filter rules (sampling rates) obtained by average queue size

5.3.3 All-to-all Dissemination

For this case, each node disseminates its ABR priority and a flag indicating whether or not the node serves as an ABR to all other nodes within the area. The dissemination procedure for a single node will be described, where other nodes follow the same procedure. For example, assume that the node n_0 is the primary ABR and the node n_6 is the source node for the data dissemination in Fig. 5.9. The node n_6 first unicasts its data towards the primary ABR n_0 by sending its data to the parent node n_2 and multicasts it to the children nodes n_{11} and n_{12}. When the node n_2 receives the data from n_6, it will further disseminate the data to the primary ABR n_0 and its children nodes n_5 and n_7, where n_7 will further disseminate the data to its child node n_{13}. Since the node n_0 received the

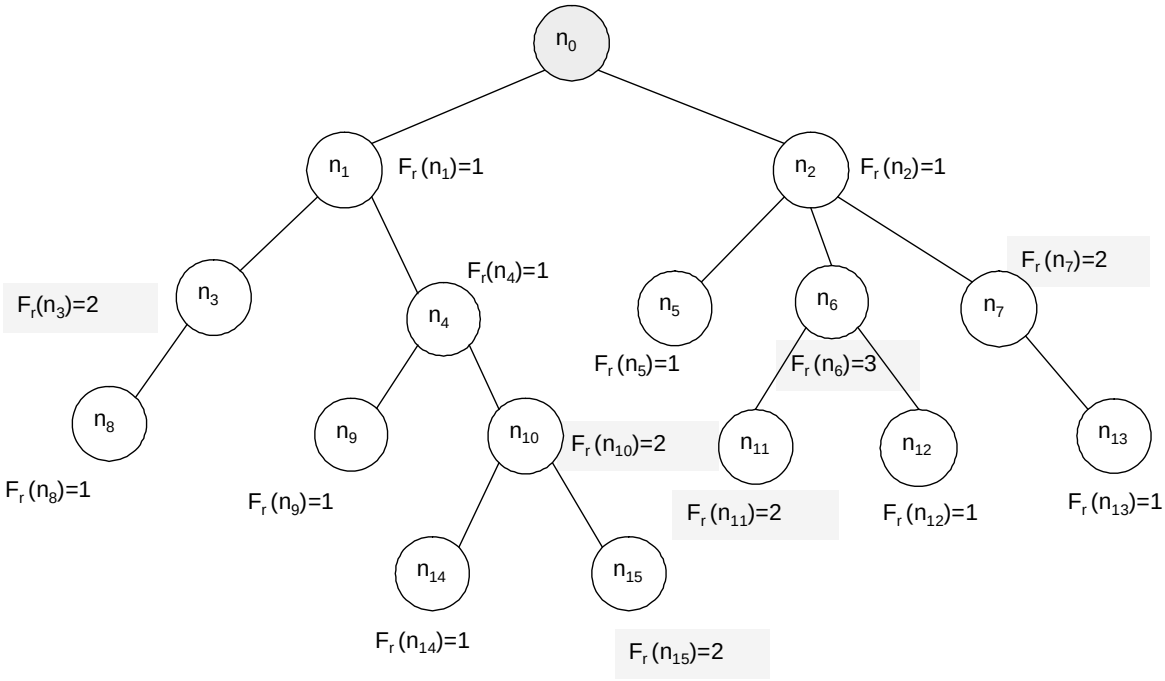

Figure 5.10: Final filter rules for the sampling-based dissemination

data from n_2, n_0 only disseminates the data towards n_1. At the end of this procedure, all nodes within the area will receive the data from n_6 without any duplication. The filtering mechanism is used when the data is disseminated from a parent towards its children nodes.

Fig. 5.9 shows the initial filter rules, where local filter rules are determined by the maximum of average queue sizes of local links (e.g., the maximum average queue size for the node n_{15} is higher than the node n_9; therefore the filter rule (sampling rate) for the node n_{15} is 4 while the filter rule for n_9 is 1. An example function which maps an average queue size to its corresponding filter rule is shown in Fig. 5.8. When the average queue size for a particular node changes, this node will adapt its filter rule dynamically. For example, if the average queue size of the node n_{15} decreases, the node n_{15} may change

its filter rule from 4 to 2. Independent of the local filter rule assignment, each node runs the distributed filter placement algorithm, which requires only local information exchange between parent-children nodes, described in Section 5.2.1. Fig. 5.10 shows the final filter rules. The data coming from the source node n_0 will be disseminated to children nodes after applying these final filter rules.

5.4 CDF to DDOA Implementation

A CDF simulator, which shows the proof-of-concept implementation of the CDF, was developed using the Java programming language with XML and XPATH libraries. The simulator is shown in Fig. 5.11, where there are 16 CDF agents (i.e., default multicast tree). The user can either select the default tree from the "topology" menu or enter its own tree using the *Node* button which creates a new CDF agent when clicked. A communication link between two agents can be established (with parent-child relation) by the mouse. Six CDF functions, which are shown under the *Mode* menu, are implemented, where *Sampling, Deviation and Averaging* are from one source to multiple destinations while *Traffic vector* is from multiple sources to a single destination (e.g., all-to-one dissemination). The *ABR Selection Traffic* function corresponds to the all-to-all dissemination inside a DDOA area.

Fig. 5.11 shows the deviation-based dissemination from one source (node 0) to multiple

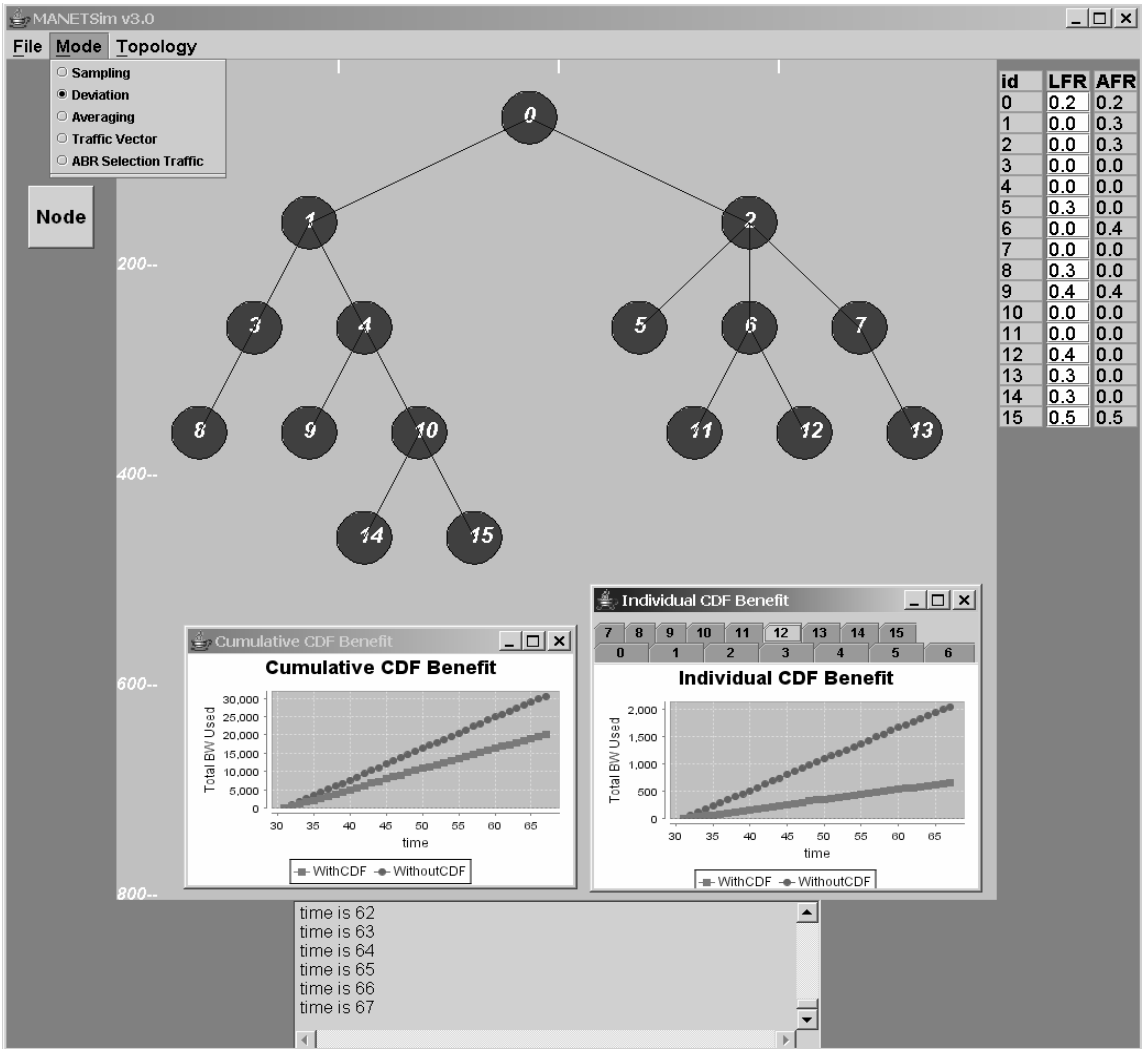

Figure 5.11: CDF simulator

destinations. Three text-field columns are shown on the right side, where *id, LFR* and *AFR* represent the node id, local and active filter rules, respectively. Note that *LFRs* are input to the distributed filter placement algorithm and editable text-field, where the user may change their values at any time. The *AFR* text-fields are initially not displayed and their values are shown after the filter placement algorithm converges. For example, the filter rule file in XML format is created at node 1 using the active filter rule value 0.3. The source node 0 starts disseminating its data towards the downstream nodes after 30 time units passed. In each time unit, 100 data samples, where each sample is uniformly distributed between 0 and 100, are created in the XML format and disseminated after the source filter rule is applied to these data objects. The *XPATHAPI eval* function is used to filter out the unwanted data samples dynamically. The *Cumulative CDF Benefit* plot on the left side shows the network-wide total bandwidth used with and without CDF while the *Individual CDF Benefit* plot on the right side shows the total bandwidth used with and without CDF when the data is delivered to node 12 (the individual benefit can be displayed for any node by selecting the corresponding chart). Note that the total bandwidth used without CDF is calculated after the source filtering and the incoming data is filtered at nodes 2 (*AFR=0.3*) and 6 (*AFR=0.4*) before delivered to node 12. Since the active filter rules 0.3 and 0.4 are relatively high, the CDF benefit for node 12 is relatively high compared to the cumulative case.

In Fig. 5.12, the multicast tree, which consists of 7 CDF agents, is entered by the user. The

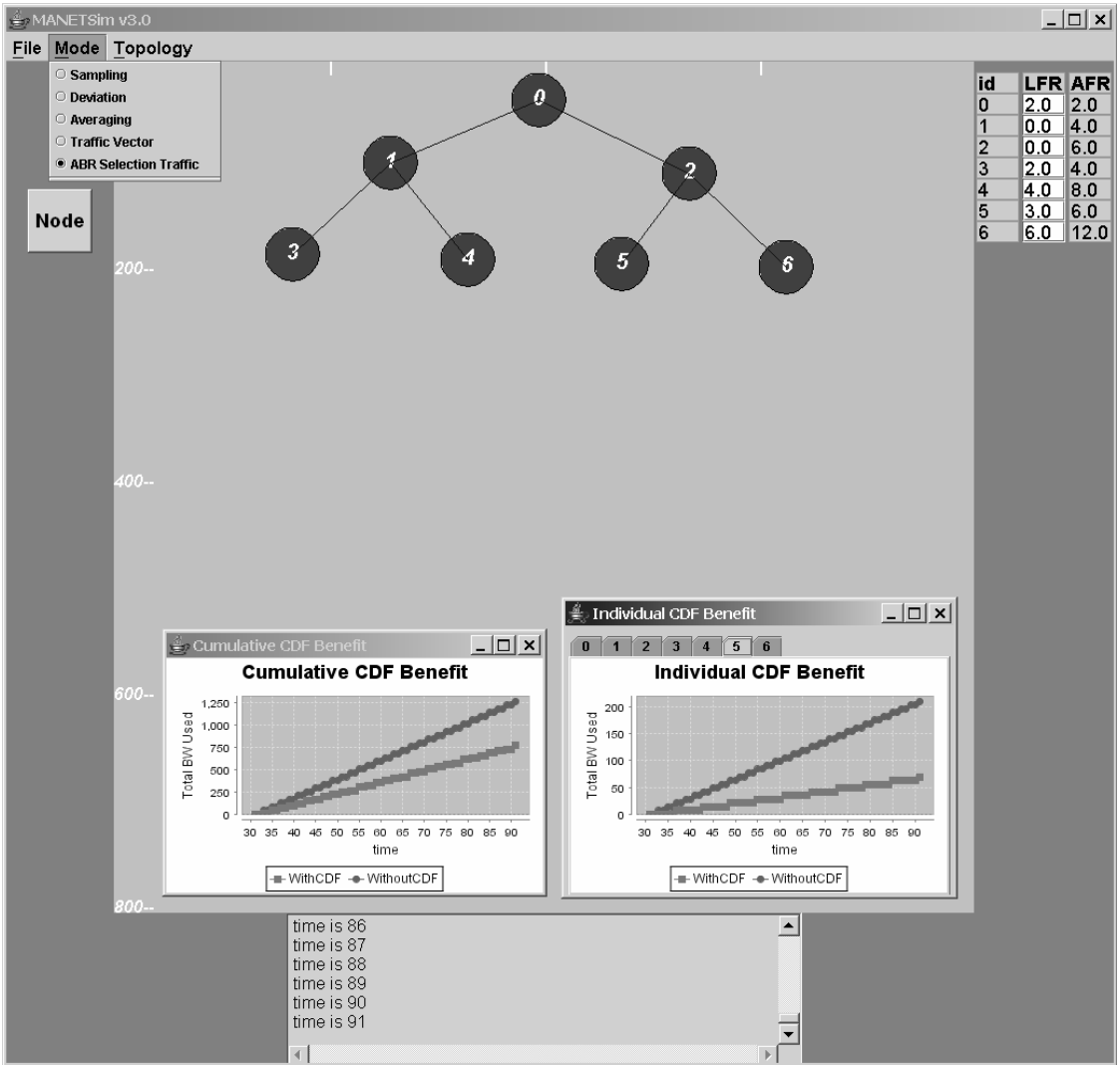

Figure 5.12: CDF simulator

CDF function is *ABR Selection Traffic*, where each node periodically disseminates its ABR priority and a flag indicating whether or not the node serves as an ABR to all other nodes inside this DDOA area (assume node 0 is the primary ABR node). Each node first unicasts its data object to the primary ABR node, where data objects coming from different sources are aggregated to an XML file to be further disseminated towards the downstream nodes using the sampling-based dissemination. Each source node assigns its own sequence number (SN) to its data object, where SN together with the modular operator is used to apply the filter rule. For example, the primary ABR node will disseminate the data objects if their sequence numbers are even since the active filter rule is 2.0 (i.e., *SN mod AFR = 0*). Note that the local filter rules are determined by excluding the source filter rule. For example, LFR for node 5 is 3.0, which means node 5 needs every 6th data disseminated by any source node.

5.5 CDF Application to URCA

In this section, we present another CDF application for ILSA Unicast Routing Control Agent (URCA), which involves finding a set of the OSPF link metrics that distributes traffic evenly among links within the network, reroutes the traffic away from the congested links, and utilizes the preferred routes. The choice is influenced by the network topology and the traffic demand between each source and destination pair within the network. The

goal is to find a set of link metrics that not only minimizes the possibility of congestion across a link in the network but also works well across a wide range of varying network conditions. During the operation, the network conditions may change due to several factors (e.g., soft and hard node/link failures, variations in traffic demand and link capacity, and topology change due to addition/subtraction of node/link and mobility). Therefore, the URCA agent should execute its algorithm periodically (or immediately in case there is a sudden significant change) so that it recalculates its new link weights (hence its new shortest path routes) according to new network conditions.

The URCA algoritm can run either in a distributed manner on each local URCA platform or in a centralized manner on the lead URCA. In the former case, the local URCA algorithm will continuously periodically monitor the local platform to determine whether one of the events triggering the local URCA algorithm took place (i.e., local congestions) and react locally to these events (e.g., changing local link weights). This approach does not require information exchange among the platforms, which minimizes not only additional signalling overhead but also response time to react the link congestion events. However, the route change decided by the local URCA algorithm on the local platform may affect the other parts of the network, where the local URCA algorithm on other platforms may be triggered and the convergence to consistent link metrics may take a longer time.

In the centralized case, the lead URCA runs the global URCA algorithm to determine the optimum link weights (hence corresponding shortest path routes) for the overall network.

The centralized approach guarantees that the URCA algorithm converges to one set of consistent link metrics. However, the URCA lead needs up-to-date network information (e.g., traffic and capacity matrix) which increases not only the signaling overhead but also the delay for reacting to the link congestion events. There are three components for this delay: *(i)* dissemination of the traffic and capacity change information to the lead URCA, *(ii)* finding a new set of link metrics by the global URCA algorithm, and *(iii)* shortest path calculations using new link weights. Therefore, in the centralized version, it is critical to provide accurate and up-to-date network information to the lead URCA continually in order to decrease this delay (e.g., components *(i)* and *(ii)*) while the network information dissemination should be performed in a controlled manner in order to decrease the additional signaling overhead. CDF is responsible for efficient network information dissemination to the URCA lead.

5.5.1 URCA Overview

The URCA lead obtains periodically the network information through the CDF and then invoke a heuristic([39]) to compute the OSPF link weights either adaptively according to the received network information (e.g., detection of new link congestions) or periodically (e.g., every 30 secs). The network information consists of:

(i) Traffic Demand: The aggregate traffic flowing between each source and destination pair. If a significant change from the current value is detected, the traffic matrix is

updated locally and the information is propagated to the URCA lead through the CDF.

(ii) Topology Information: The URCA algorithm needs the up-to-date network topology (e.g., connectivity matrix). The URCA uses the network topology information disseminated through OSPF link state advertisements.

(iii) Link Characteristics: Link characteristics such as availability, bandwidth, utilization, and type and node mobility information need to be disseminated to the URCA lead as well. The URCA heuristic attempts to divert traffic away from links that have low availability and bandwidth to links that have higher availability and bandwidth. If a link has consistently high utilization and mobility, the URCA algorithm should move traffic away from this link to other links which have lower utilization and mobility. All these link characteristics and node mobility is reflected to a single parameter called the *normalized link capacity*.

The normalized link capacity of a link j is substituted for c_j in computing the network wide objective function $\sum_l f(u_l/c_l)$. f' represents the derivative of f which is defined

as (Ref. [39]):

$$f' = \begin{cases} 1, & 0 \leq u_l/c_l < 1/3 \\ 3, & 1/3 \leq u_l/c_l < 2/3 \\ 10, & 2/3 \leq u_l/c_l < 9/10 \\ 70, & 9/10 \leq u_l/c_l < 1 \\ 500, & 1 \leq u_l/c_l < 11/10 \\ 5000, & 11/10 \leq u_l/c_l < \infty \end{cases}$$

where u_l is the total load on the link l (the total load is calculated by using traffic demand and topology information) and c_l is the normalized capacity of link l. $f()$ is a convex function that penalizes solutions that have heavily-loaded links. Next, the value of w_j is determined that minimizes f. Different values of w_j are iterated and f is computed for each iteration. The value of w_j is chosen that minimizes f.

5.5.2 CDF Functionality for URCA

The relevant questions are that when is the URCA algorithm triggered at the lead node and how accurate and frequent traffic, topology and capacity information are needed at the lead URCA? When more samples of the network information were delivered to the lead URCA, the global network information would be more accurate at the lead node but the signaling overhead would increase correspondingly. The goal of the CDF is to minimize the data dissemination overhead without jeopardizing the accuracy of the network information

needed by the lead URCA. The other nodes do not know when the lead URCA executes its global algorithm; therefore, the other nodes need to disseminate their traffic demand (T) and link capacity (C) information continuously. In order to minimize the overhead, new samples should be reported when there are significant deviations from the previously disseminated samples.

Note that the proposed URCA algorithm executes only in case of high link utilization on one or more link(s) (e.g., link utilization threshold may be 0.9) to remove the congestion (even the URCA algorithm is invoked, it may not calculate new link weights if none of the links is congested). Without CDF all local URCA nodes send the information to the lead node even though the information may not be used by the lead URCA as none of the links would exceed the URCA threshold. One task of the CDF is to disseminate traffic demand and capacity information in a distributed and timely manner only when this information is required at the lead URCA node thereby saving on unnecessary overhead.

To prevent delay from occurring when links are detected as congested and then sending completely accurate T and C information to the lead URCA, CDF provides continuous information dissemination with lower frequency (i.e., higher deviation constant) even if there is no congestion in any part of the network. Suppose that the deviation constants for T and C are represented by T_{dev} and C_{dev}, respectively. The CDF changes dynamically T_{dev} and C_{dev} according to the link utilization levels in order to adapt its dissemination frequency. For example, if a link utilization on any link in the network independent of

the multicast tree is 0.9 then T_{dev} and C_{dev} might be set to 0.05 (a sample with a small deviation will be disseminated); on the other hand, if the link utilization is around 0.6 then T_{dev} and C_{dev} might be set to 0.3. Depending on the characteristics of T and C (their short term variations), the number of samples for two cases might be very different (hence the overhead might be very different). Note that all nodes disseminate their information using the smallest deviation constant calculated by considering the most congested link in the network and the distributed filter placement algorithm is used to propagate this smallest deviation constant to all other nodes. The reason for this approach is that the URCA algorithm is triggered when one or more links in the network get congested and the complete and accurate T and C information from all nodes (i.e., not only from the nodes whose links are congested) within the area are needed just before invoking the global URCA algorithm at the URCA lead.

In Fig. 5.13, the node n_0 is the URCA lead and all other nodes in the multicast tree apart from the URCA lead disseminate their traffic matrix to the lead URCA (e.g., there are 14*15=210 different source-destination pairs; hence, 210 different traffic demand items need to be disseminated continuously). The deviation-based dissemination functionality of the CDF is used for an efficient data dissemination of traffic demand matrix to the lead URCA. Each node maintains the traffic demand rates (e.g., current and previously disseminated rates) per destination node.

Fig. 5.13 shows the initial filter rules obtained through the maximum of local link utiliza-

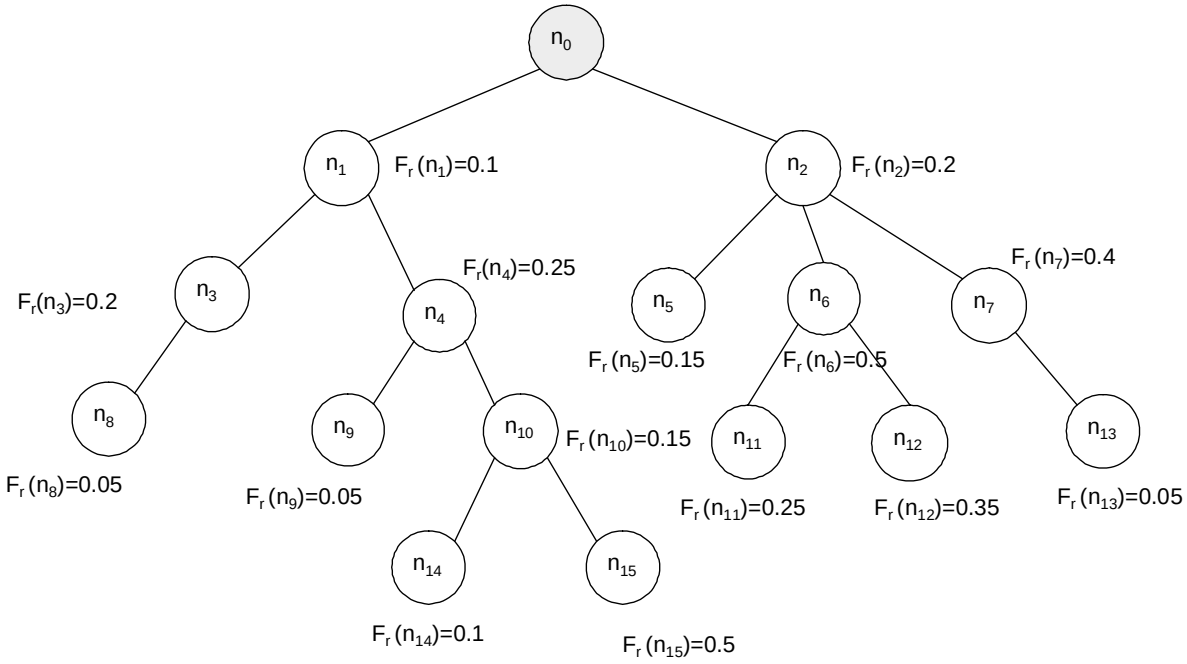

Figure 5.13: Initial filter rules (deviation constants) obtained from link utilizations

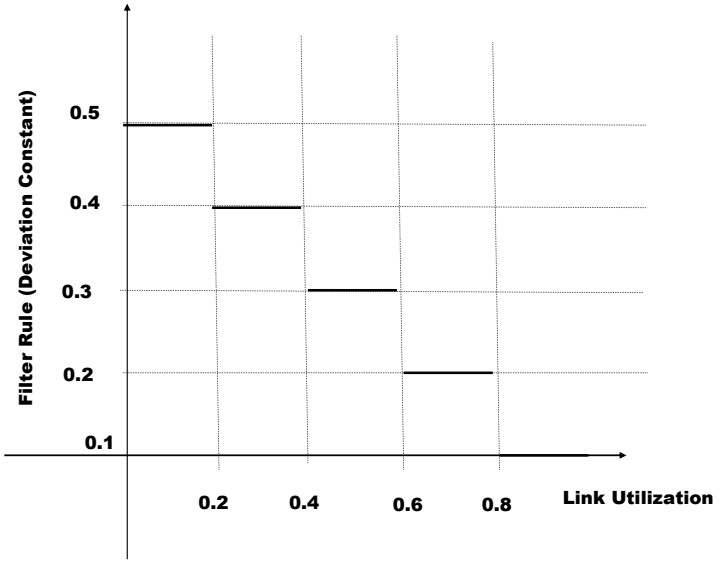

Figure 5.14: Filter rules determined by link utilizations

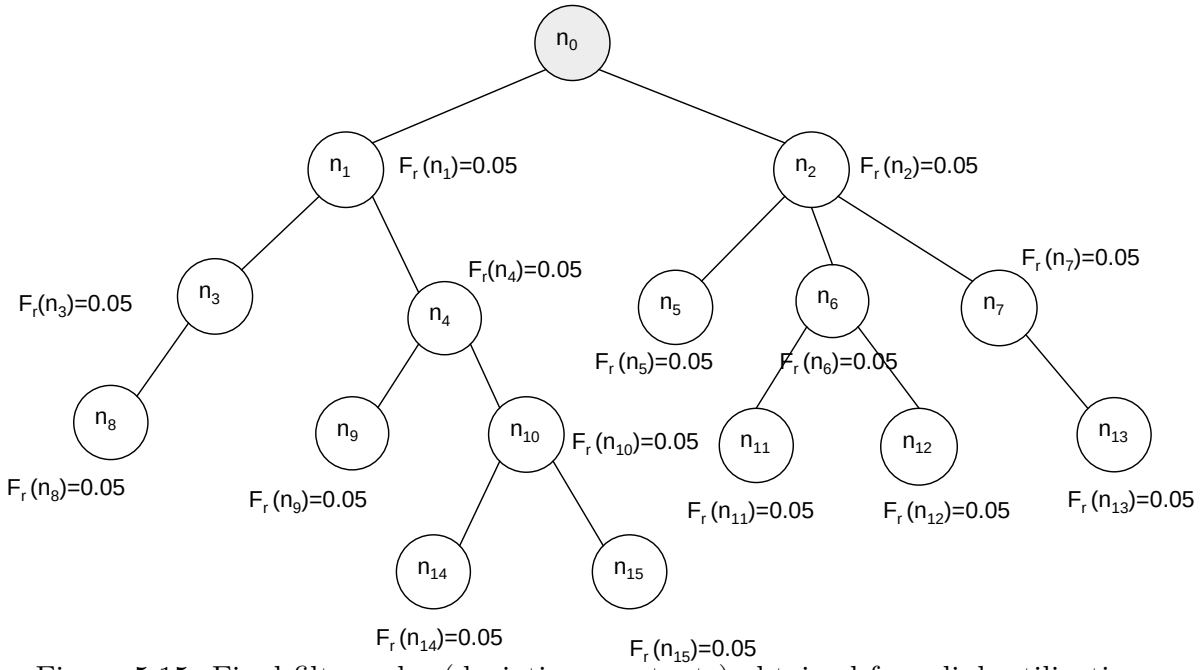

Figure 5.15: Final filter rules (deviation constants) obtained from link utilizations

tions for the deviation-based dissemination. In this example, the node n_{15} has lower link utilization and therefore its filter rule (i.e., deviation constant) is set to 0.5. When and if the deviation between the current traffic rate and the previously disseminated traffic rate is higher than 0.5, this data will be disseminated towards the URCA lead. An example function, which maps a local link utilization to a local filter rule, is shown in Fig. 5.14. The link utilization, which is between 0.8 and 0.9, sets the deviation constant to 0.1 whereas the link utilization, which is between 0.2 and 0.4, sets the deviation constant to 0.4 in Fig. 5.14. The higher the deviation constant is, the more the number of samples is to be disseminated.

The distributed filter placement algorithm performs the following steps:

1. Each node measures its outgoing links' utilizations and determines its local filter rule.

2. Each node propagates its minimum filter rule, which is calculated by using its local and all recursive descendents' filter rules, to its parent node.

3. When the smallest filter rule is reached to the URCA lead, it is disseminated towards the downstream nodes.

Fig. 5.15 shows the final filter rules after the distributed filter placement algorithm converges. Note that the smallest filter rule is 0.05 in Fig. 5.13; therefore, the distributed filter placement algorithm places an active filter of 0.05 into all nodes of the multicast tree. When and if the deviation between the current measured traffic rate and the previously disseminated traffic rate is higher than 0.05, the current measured traffic rate information will be further disseminated, otherwise filtered out at each node. The dissemination frequency is higher in this case since the deviation constant 0.05 indicates that the global URCA algorithm will be executed to divert traffic away from the congested link(s); therefore, the accurate traffic demand matrix information is needed at the URCA lead.

5.6 CDF to URCA Implementation

A packet-based event-driven CDF simulator, which shows the proof-of-concept implementation of the CDF, was developed from the scratch using the Java programming language. The simulator allows us to specify a network topology and the traffic demand matrix for all source destination pairs. The simulator is shown in Fig. 5.16, where there are 15 CDF agents. The default multicast tree, which is used for the CDF algorithms and traffic vector dissemination, consists of dark links. Five bi-directional links between 1 and 2, 2 and 12, 6 and 11, 12 and 13, and 12 and 14 do not belong to this tree and are used for the URCA routing purpose. The user can either select two default topologies from the *Topology* menu or enter its own topology using the *Node* button which creates a new CDF agent when clicked. A communication link between two agents can be established (with parent-child relation) by the mouse. The black node 0 represents the URCA lead agent which optimizes the shortest path routes among source-destination pairs by varying OSPF link weights. The traffic vector and capacity information are input to the URCA agent and disseminated by the CDF agents to the lead URCA periodically and efficiently.

Fig. 5.16 shows the deviation-based dissemination from the CDF agents to the lead URCA (node 0). Three text-field columns are shown on the right side, where *id*, *LFR* and *AFR* represent the node id, local and active filter rules, respectively. LFRs are input to the distributed filter placement algorithm and determined by the local link utilization levels as described in Section 5.5.2. For example, if one of the local link utilizations for a particular

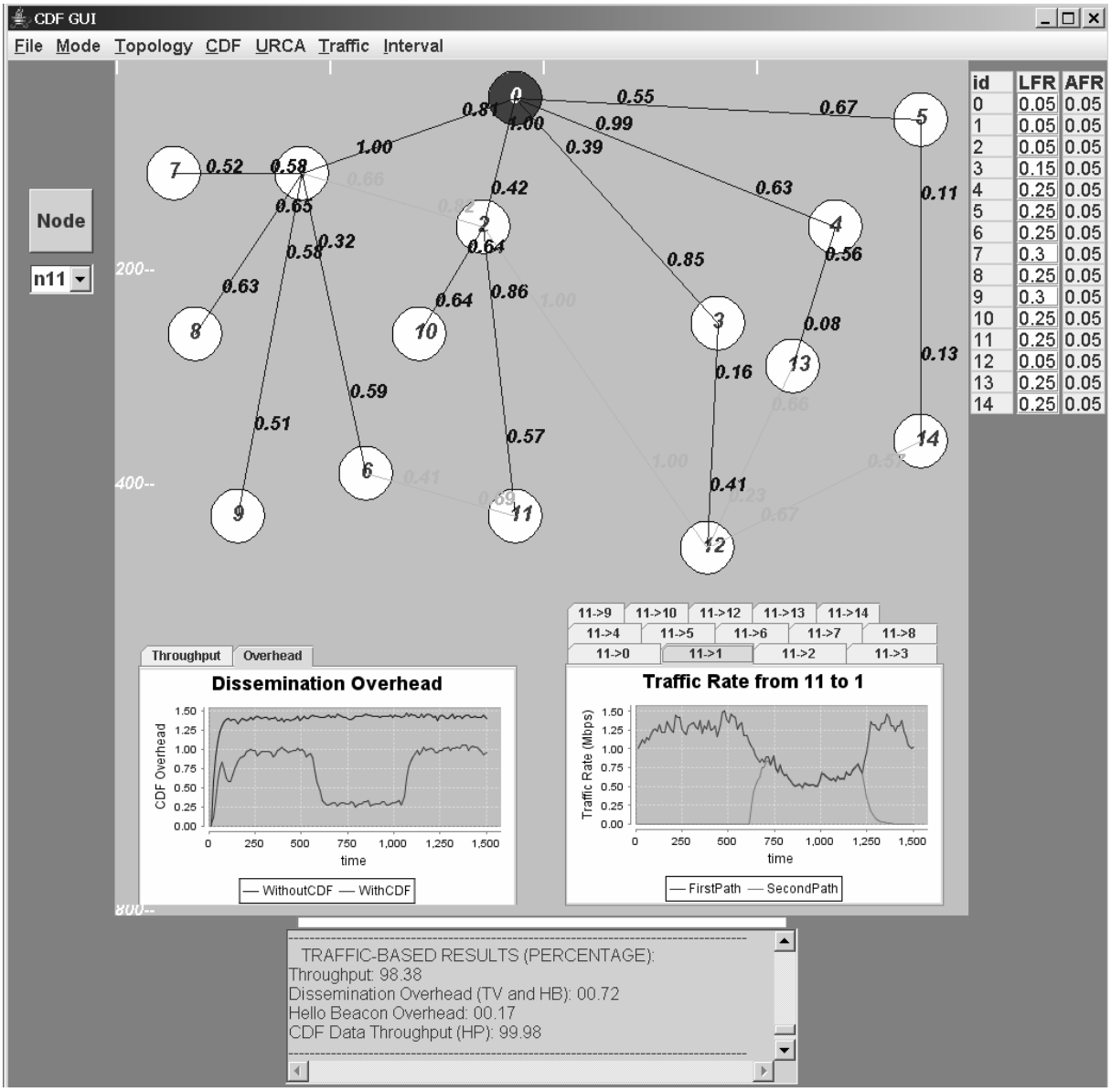

Figure 5.16: CDF Simulator

node is higher than 0.95, the LFR for this node is set to a lower value (e.g., 0.05). AFRs are the output of the distributed filter placement algorithm (e.g., AFR are the same for all nodes due to the URCA design requirement).

In our simulation experiments, the UDP flows are generated between each source-destination pair according to the mean traffic rate (e.g., 1.10 Mbps) specified by the user as an input to the simulator. For example, Fig. 5.17 shows the current traffic rate between nodes 0 and 1, where the target mean traffic rate is 1.10 Mbps and the average traffic rate generated by the simulator is reported as 1.113 Mbps. The generated traffic consist of 64 Kbps UDP flows which indicates that there are 17 active UDP flows on the average for the 1.10 Mbps traffic demand. Each flow consists of 40 packets, where each packet contains 128 bytes of data payload. The flow arrivals are deterministic such that after creating 40 packets from an active flow and a new flow is scheduled deterministically. The inter-packet arrivals within a flow are either exponentially distributed or deterministic.

The user can change the following parameters before the simulation experiment: traffic characteristic from the *Traffic* menu and dissemination interval from the *Interval* menu. Traffic can be generated either by random or deterministic inter-packet arrivals within a flow. The default dissemination interval is 12; it can be changed to either 6 or 24 to observe the effects of dissemination interval on the results. This interval is also used as a time window for calculating the traffic rate values based-on the measurements of

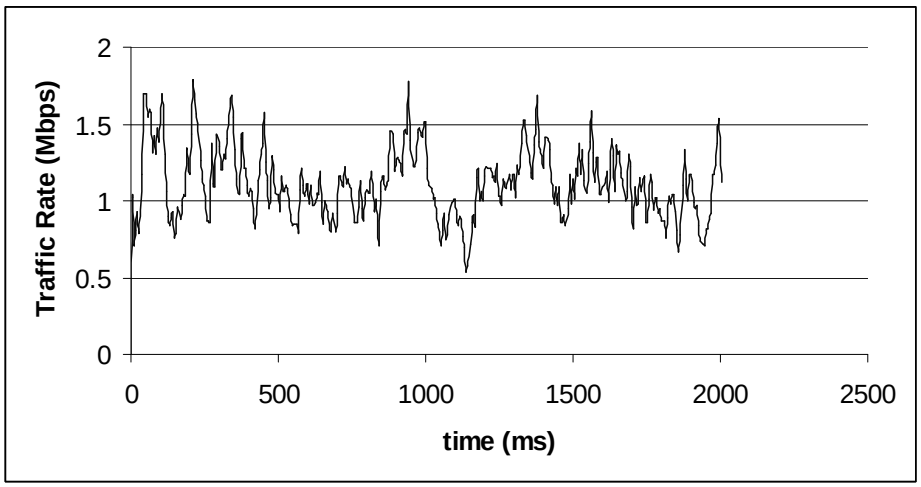

Figure 5.17: Traffic rate from 0 to 1. The user specified mean traffic rate is 1.1 Mbps and the average is calculated as 1.113 Mbps from the above plot.

outgoing traffic from this source to all other destinations. For example, the number of bytes generated during this time window is counted and the traffic rate value is calculated by dividing this sum of bytes to the time window. URCA needs a mean traffic value; therefore we keep the running average of the traffic rate value, where the weight for the current value is set to 0.3 while the previous running average value is weighted by 0.7. These samples are disseminated either using CDF (i.e., if the current traffic rate is deviated from the previously disseminated value by at least the amount of active filter rule) or without CDF (i.e., every sample calculated for each time window is disseminated). The performance metrics are the dissemination overhead and the accuracy of URCA algorithm (i.e., throughput) with and without CDF cases. We conjecture that the data dissemination overhead is reduced using the CDF and the performance of the URCA is not affected adversely from the CDF filtering.

The connection between two nodes has a certain capacity and the packet is transmitted from one node to another using this capacity information. Initially, each link capacity is set to 30 Mbps, which is not adequate to carry all the traffic (i.e., this case represents the existence of the congestion). The default simulation time is set to 1500 and all the link capacities are changed to 50 Mbps when the simulation time is between 500 and 1000, and 30 Mbps between 1000 and 1500. The reason is to observe the transient behavior of the CDF during the transitions between congestion and non-congestion states in the network. When the link capacities are 30 Mbps, the AFR is reported as 0.05 while the link capacities are 50 Mbps, the AFR is 0.2. When the AFR is 0.2, the CDF overhead is reduced considerably as shown in Fig. 5.16 (left plot). Note that during this period the URCA agent is triggered (periodically) but it did not perform its optimization since there is no congested link in the network. Therefore the dissemination overhead is reduced more by setting LFR to higher values when no congestion is available in the network.

There are two queues for each node, one is for application traffic and the other one is for CDF traffic such as traffic demand matrix and local CDF control packets (i.e., node identifiers and filter rules). The CDF traffic have a strict priority over the application traffic since it includes the critical information for network-optimization. Providing the strict priority for the control packets helps to detect the network congestion promptly so that the URCA optimization tool can be invoked earlier.

Each packet generated by a source node is transmitted using the shortest path routing

until it reaches the destination node. The shortest path routes are calculated according to the OSPF link weights. The URCA agent is triggered periodically with a period of 300, where it uses the traffic and capacity information delivered by CDF to optimize the shortest path routes. The traffic is routed over the new shortest paths after the URCA calculates new link weights. For example, after the URCA route optimization agent is run, the traffic from node 11 to 1 is sent using two equal-cost paths when the time is between 600 and 1250 (the right-side plot in Fig. 5.16).

We report the network-wide throughput and dissemination overhead for four different combinations of CDF and URCA, where each can be ON or OFF. The throughput and overhead are reported in two ways: one is the percentage of the network-wide delivered traffic to the overall network-wide generated traffic (for throughput) and the second one is the absolute value of the total consumed bandwidth (in bytes). The *Network-wide Results* plot on the left side shows the network-wide throughput and overhead while the *Per Node Results* plot on the right side shows the traffic rate between source-destination pairs (the traffic rate between a source to any destination can be displayed by selecting the corresponding destination chart after selecting the source node using the selectable choice menu located under the *Node* button). 15-node simulation results are summarized in Tables 5.1, 5.2, and 5.3.

	W/o CDF and w/o URCA		*W/ CDF and w/ URCA*	
	(Percentage)	(in bytes)	(Percentage)	(in bytes)
Overall Throughput	97.39	55,450,368	98.41	55,782,528
Dissemination Overhead	1.37	782,464	0.97	547,456
Hello Beacon Overhead	0	0	0.17	96,480
BW Usage for Application	-	127,746,312	-	130,567,304
BW Usage for Dissemination	-	1,198,976	-	784,160
BW Usage for Hello Beacon	-	0	-	47,744

Table 5.1: 15-Node simulation results using time interval 12

	W/o CDF and w/o URCA		*W/ CDF and w/ URCA*	
	(Percentage)	(in bytes)	(Percentage)	(in bytes)
Overall Throughput	97.53	55,114,144	98.18	55,239,296
Dissemination Overhead	0.71	403,232	0.28	157,184
Hello Beacon Overhead	0	0	0.09	52,128
BW Usage for Application	-	126,211,336	-	127,732,648
BW Usage for Dissemination	-	611,072	-	189,408
BW Usage for Hello Beacon	-	0	-	25,984

Table 5.2: 15-Node simulation results using time interval 24

	W/o CDF and w/o URCA		*W/ CDF and w/ URCA*	
	(Percentage)	(in bytes)	(Percentage)	(in bytes)
Overall Throughput	97.18	55,948,320	98.02	55,928,352
Dissemination Overhead	2.56	1,472,544	1.67	953,120
Hello Beacon Overhead	0	0	0.30	169,952
BW Usage for Application	-	128,667,593	-	129,785,761
BW Usage for Dissemination	-	2,258,560	-	1,305,408
BW Usage for Hello Beacon	-	0	-	84,960

Table 5.3: 15-Node simulation results using time interval 6

5.7 Application of DSRP in CDF Design

We also formulated an initial approach to exploiting the DSRP's *virtual backbone (VB)* component to reduce the problem space. Additionally, the amount of processing and signaling involved in the filter placement phase can also be reduced by utilizing VB. Although the details still need to be designed in the future, the approach can be outlined as follows.

Consider the network shown in Fig. 5.18. The source n_0 distributes data to a subset of nodes, which may include some or all nodes n_1 through n_{26}. The channelization algorithm executed at n_0 uses the interest matrix, whose number of columns may be as high as 26; flow-to-group matrix X, whose number of columns is equal to the number of multicast groups $|G|$, and user-to-group matrix Y, whose number of rows may be as high as 26.

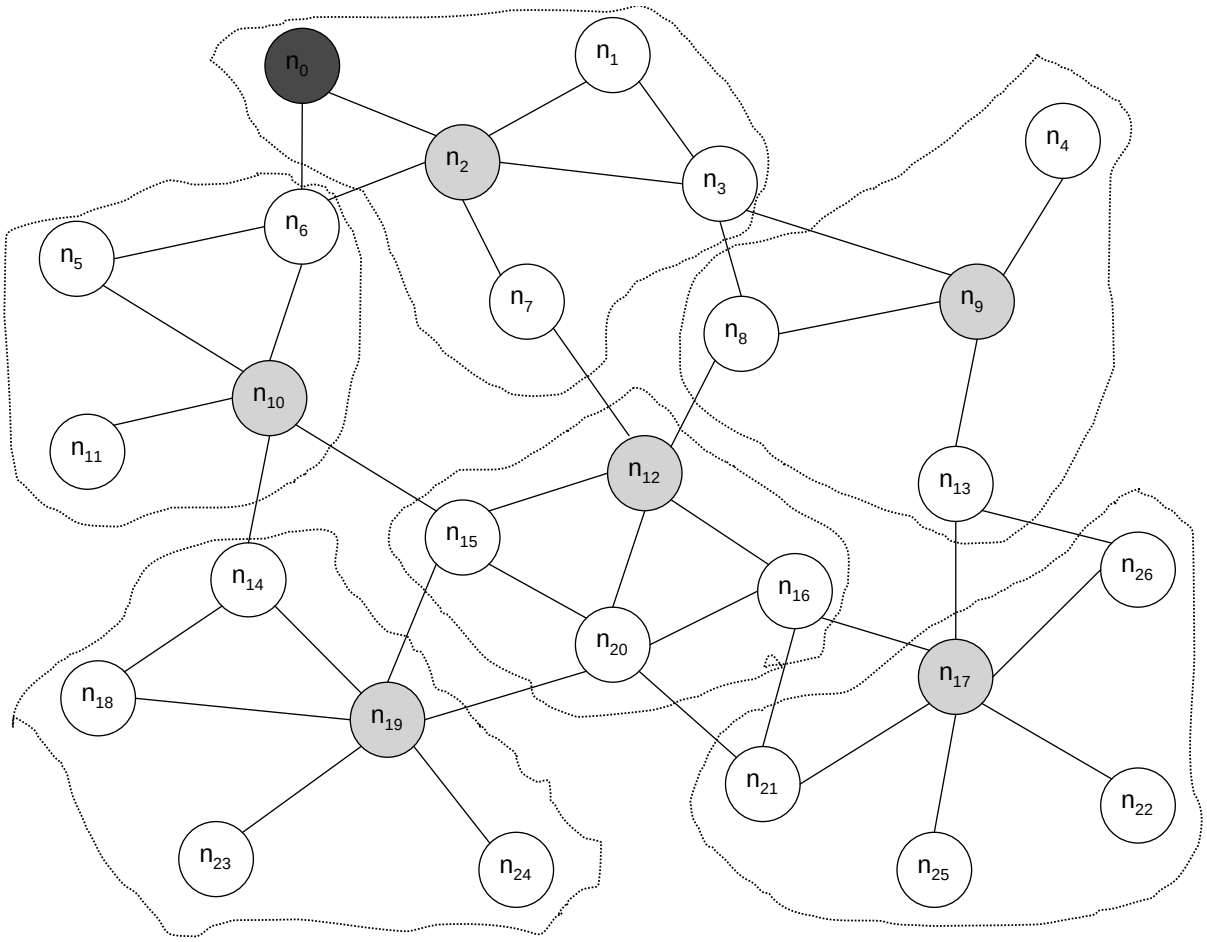

Figure 5.18: Virtual backbone structure

The parameter $|G|$ increases as the number of nodes in the network grows.

Suppose that the following approach is used to reduce the problem space and the amount of CDF traffic:

- A virtual backbone (VB) is formed as a connected dominating set and maintained as the topology changes. The six shaded nodes become the members of the VB.
- The destination nodes send their interests in particular flows only to their Primary

VB (PVB) nodes, e.g., nodes n_5, n_6, and n_{11} contact their PVB on node n_{10}.

- Each VB node sends the summarized interest of its subordinate nodes to the source's PVB, essentially acting as their proxy for data reception. This action may be triggered by a query from the source's VB relayed through the mesh of VB nodes. At this stage, the source builds the interest matrix W for the interested VBs. The size of W is reduced from as many as 26 columns to the maximum of 6.

- The source runs the channelization algorithm to form the multicast groups. The required number of multicast groups $|G|$ can be now much smaller because it should cover 6 instead of up to 26 members. Additionally, one dimension of matrices X and Y is reduced from 26 to 6.

- Using the output of the channelization algorithm, the source finds the VB nodes for each multicast group (multiple multicast groups can be formed for a particular flow).

- The multicast tree is formed for each multicast group on top of the VB. Each VB-based multicast tree is likely to be smaller than that built for the entire network.

- Once the multicast tree has been formed, the users send their filter rules to their PVB node.

- The filter reduction and placement algorithm is run on a set of smaller VB-based trees.

- The data dissemination takes place from the source n0 to the leaf VB nodes (push model), e.g., PVB on node n_{10} will be the multicast destination for the actual destination nodes n_5, n_6, and n_{11}.

- The end users ask their PVBs to relay data to them (pull model). They can switch back and forth between ON and OFF state for data reception without impacting the rest of the data dissemination process.

Chapter 6

Distributed Robotics Testbed Implementation

In a distributed robotics system, wireless connections among mobile robots can become broken due to severe network stress, robot or link failures, and constant mobility in the system. Typical applications of distributed robotics systems, such as real-time transactions, disaster recovery, and search-and-rescue operations, include exchanging large amounts of data among robots. As the wireless connections are frequently broken during a session, the traditional abort-and-restart approach often results in long delays and puts a heavy burden on end users for these applications.

System reliability can be achieved in different levels, from data transfer mechanisms in the

transport and lower layers, where hardware, firmware or software units ensure successful data transmission, to upper layers (above transport) where an application is desired to continue its operation without (or with minimum) interruptions due to failures, mobility, or QoS degradation. The *reliable server pooling (RSP)* [99, 100, 23, 66, 102] mechanism is designed to deal with the reliability using the latter approach. For high level protocols (above transport layer) the RSP framework proposed by the IETF is called the *RSerPool* [99, 100]. Since RSerPool has been mainly targeted for wired networks, serious performance problems have been found in the suggested protocols implementing the RSerPool for MANETs [102]. *Dynamic survivable resource pooling (DSRP)* [36, 37, 38] provides survivable access to resources and services in mobile robotics networks. The PEs accessed by PUs are pooled together for higher availability and failover. In the DSRP scheme, NSs are placed on a virtual backbone (VB) [61]: a highly distributed, scalable, and survivable network formed and maintained through one-hop beacons.

In distributed robotics framework, each robot that provides a service is a *pool element (PE)* and the user that receives the service is a *pool user (PU)*. In the DSRP, the *Name Servers (NSs)* are responsible for maintaining PE pools, load balancing, and PE discovery. The PU resolves the mapping from a server-pool handle to the addresses of PEs registered in this pool by querying its NS. Under this scheme, whenever a PE fails, PUs that utilize that PE should transparently switch over to another PE in the pool, possibly migrating the session to the new PE [5, 94]. In this study, we present the implementa-

tion of the DSRP framework, together with its architecture and protocols, in distributed robotics environment. The robotic nodes are referred to as smart tiny auto-configurable robots (STARs). In this adaptation of the DSRP, a pool of STARs is viewed as a single service endpoint and therefore is able to provide reliable services for distributed robotics applications. Then we introduce a search-and-rescue image retrieval application, as a proof-of-concept for using DSRP successfully in a distributed robotics environment. New analytic models are presented for calculating the expected time for an available wireless link between the PU and a mobile PE robot to become unavailable. These models are used to evaluate the performance of the proposed DSRP framework.

In Section 6.1, we describe the VB algorithms. Section 6.2 introduces DSRP in distributed robotics with its implementation in the FPGA-based system. In Section 6.3, we present a search-and-rescue image retrieval application. Analytic models are presented in Section 6.4. In Section 6.5, we present the measurement results obtained from our distributed robotics testbed.

6.1 Virtual Backbone Algorithm

In the DSRP architecture the NSs are dynamically placed on a VB, and the set of NSs change in response to network events. The NSs in VB constitute a dominating set (i.e., each node in the network is either a member of this set or is only one-hop away from a

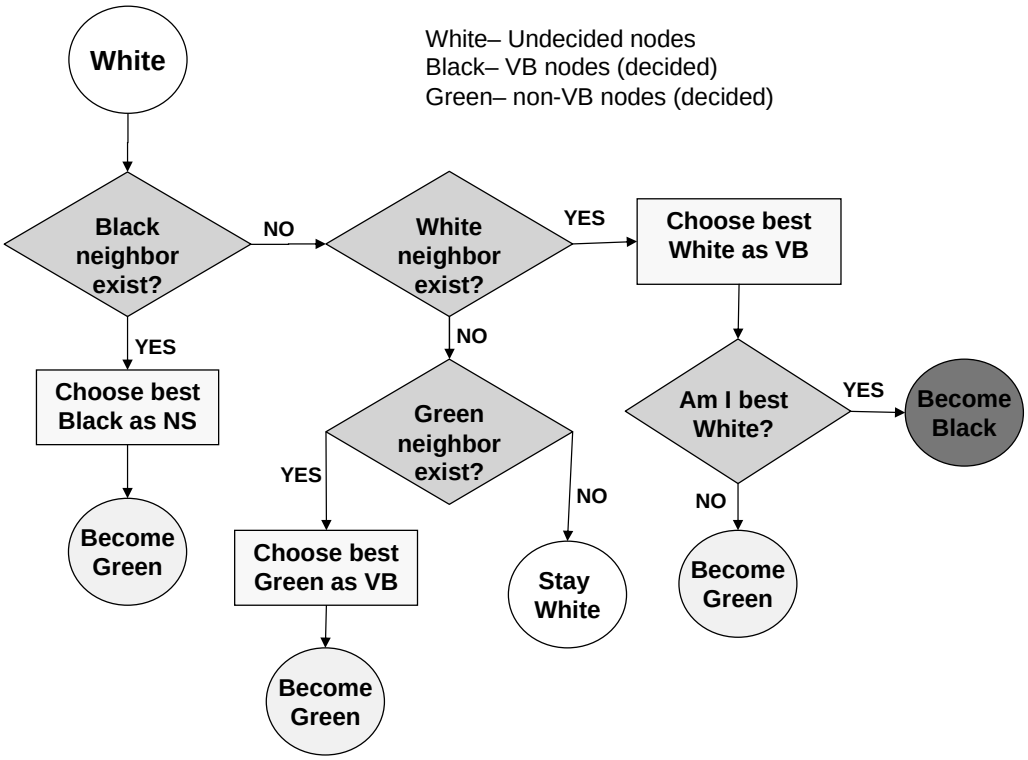

Figure 6.1: VB algorithm for finding *the best* color for a white node

member) and thus provides a fast name-resolution for a resource request.

In DSRP framework, autonomous nodes (1) form a VB by creating a mesh structure that consists of stable nodes acting as service brokers and a subset of paths (also called virtual links) connecting them; and (2) distribute service registrations, requests, and replies over the VB in a controlled scope. There are three phases for forming and maintaining the VB, namely backbone-node selection, mesh formation, and maintenance phases. These phases are all accomplished by 1-hop lightweight hello beacons (50 bytes). A hello beacon of a particular node includes *id, degree, color, nlff, NS* for this node. *Degree* is defined as the total number of neighbors for a node. *Color* indicates a node's role in VB: it is either a NS (black), has another node as its NS (green), or not decided yet (white). The

normalized link failure frequency (nlff) is the number of link losses within a time-window normalized by the number of total links at the end of the time-window.

During all three phases of the VB algorithm, nodes broadcast hello beacons periodically which are received by only 1-hop neighbor nodes. Each node maintains a Network Information Table (NIT) to keep track of neighbors' degree, nlff, color, and NS. Upon receipt of a hello beacon, each node updates its NIT information to be used by the VB algorithm.

A simplified version of the VB formation algorithm [61] is shown in Figs. 6.1, 6.2, and 6.3. In this version the decisions of choosing backbone nodes are made solely on the node degree. All nodes are initially white and go through the backbone-node selection phase shown in Fig. 6.1. The role of a node changes in response to changing network topology. To update the role of each node, the VB maintenance phase shown in Figs. 6.2 and 6.3 is deployed, where green and black nodes check their decisions periodically so that a consistent VB is maintained.

As depicted in Fig. 6.1, a white node checks whether it has any black neighbor(s). In case of a single black neighbor, this black node will be selected as the NS making the white node turn into green. If there are multiple black neighbors, *the best* black neighbor will be chosen as the NS. Nodes selected in the VB as NSs are preferred to have high connectivity and stability. High connectivity is required to have a backbone size as small as possible to reduce the communication overhead from forming and maintaining VB. The duration of being a backbone node (stability of a backbone node) should be maximized

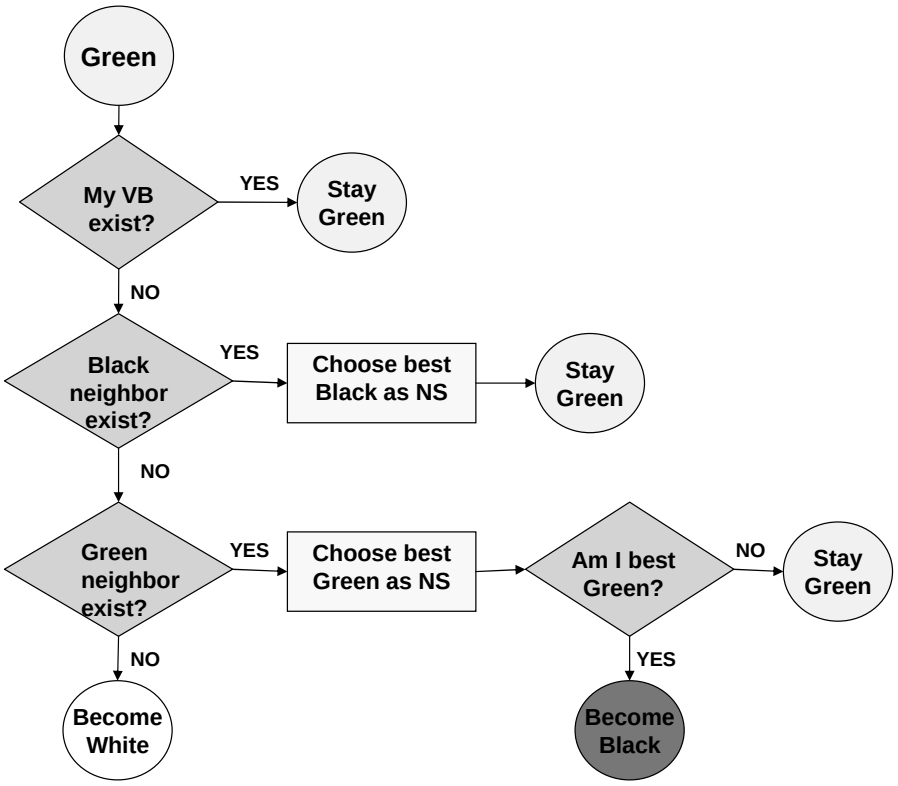

Figure 6.2: VB algorithm for finding *the best* color for a green node

since the underlying network cannot function until a new backbone node is selected. High connectivity is judged based on the number of neighbors, while the nlff metric is used to indicate the node stability. In this simplified version, *the best* black (green, white) node is defined as the node having the highest degree among the black (green, white) candidates. If two nodes have the same degree, then the strict priority for being the best node will be given to the node with the highest ID. If there is no black neighbor, then the white node will check for white neighbors (including itself) and choose *the best* white node as its NS. The same procedure of finding *the best* node will be repeated for green neighbors in case there is no white neighbor. At the end of this phase, a white node will be either green or black (except for the case that it has no neighbor).

Since each node gives its decision autonomously, some conflicts may arise between a node's decision and its neighbors' decisions. In these cases, the maintenance phase algorithms shown in Figs. 6.2 and 6.3 will be used to resolve such conflicts among the neighbors. Moreover, these algorithms reorganize a VB in response to topology changes due to the node mobility or failures.

In Fig. 6.2, a green node checks the availability of its NS. If its NS is no longer available, then the green node chooses another neighbor as its new NS by giving strict priority to black nodes such that *the best* black neighbor will be selected as its NS.

Fig. 6.3 shows the VB algorithm for a black node (NS). If a black node is deserted by its green nodes, it becomes a white node to be treated as if it were an undecided node. In another situation, if a black node has a neighbor that is also a black node, it sends a hello beacon to its green nodes indicating that it will change from black to green. This rule eliminates any redundancy of VB nodes in a close proximity. All black and green nodes that receive this message determine *the best* black node to assign as their NS so that only one of the black nodes will remain as black. An example of forming a VB is shown in Fig. 4.4. Each green node is numbered as x.y where x is the node identifier and y is its NS. For example, the node numbered as 2.1 represents the node 2 whose NS is node 1. In Fig. 4.4, node 5 has recently moved; therefore, node 1 rather than node 5 joins the VB because it is more stable. Each node runs the VB formation algorithm autonomously, and, after completing the selection phase, it immediately proceeds to the mesh formation

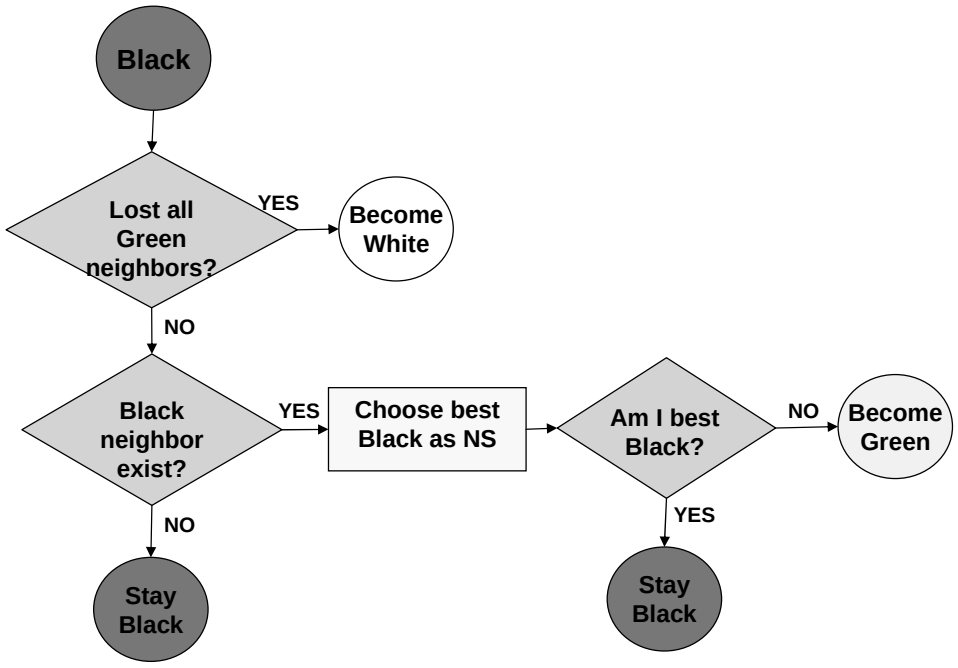

Figure 6.3: VB algorithm for finding *the best* color for a black node

and maintenance phases.

During the mesh formation, 2-hop or 3-hop virtual links are formed between the black nodes using hello beacons. After the backbone selection phase, there are one or two green nodes between each two black nodes. If there is one node (i.e., two hops), the green node chooses one of the two black nodes as its NS. In this case, the black (VB) nodes have enough information to communicate with each other as a result of receiving the green node hello beacon including its NS. In Fig. 4.4, this case is illustrated by black nodes 1 and 6 communicating through green node 5, which chooses node 1 as its NS. If there are two nodes (i.e., three hops), the green nodes recognize that they have different NSs, and include this information in the hello beacon to their NSs. Consequently, the two NSs find out about each other from the hello beacons from their respective green nodes, and know

that they can communicate using these intermediate green nodes. This case is shown in Fig. 4.4, where nodes 1 and 6 can also communicate through nodes 4 and 7.

Once the VB is formed, a server has to register with its NS. If it wants to register with more NSs, then a multicast or broadcast mechanism is needed to distribute the registration messages. Similarly, a client requests a service by contacting its NS. If the NS does not have a registration for this service, then the service should be requested from other NSs by multicasting or broadcasting.

6.2 DSRP in Distributed Robotics

We utilize the DSRP mechanism to create a reliable distributed robotics environment. An example of such an environment is an earthquake rescue mission where hundreds of robots, each equipped with cameras and/or some sensing capabilities are sent into a disaster site. Typically, these mobile robots are required to autonomously disperse into the disaster site, continuously taking pictures, measuring the level of CO2 in their spots, which can give indication of human life, and similar tasks. The measurements taken by each robot can then be processed by the robot, and, upon the request of a user, can regularly or on-demand be sent to one or more collection points to be further processed. Each robot is equipped with a low-power short-range wireless network interface, which only allows direct communication with its immediate neighbors; the overall connectivity among the

robots that are far away from each other is provided over a mobile ad hoc (multihop) wireless network. For this type of applications, it is crucial to provide the time-critical reliable service to the collection point(s) so that the rescuers can determine whether there are people who need help in this region. The DSRP architecture hence becomes a suitable choice for providing high communication reliability in such environments.

Fig. 6.4 shows a possible DSRP usage scenario where 9 mobile robots sent to a disaster region. They form three different reliable server pools, each with three robots. The three robots in a given pool continuously take pictures of their surrounding, and then they share the pictures among each other. This way, each robot has the same set of pictures for different spots to be uploaded to the collection point. One of the PEs can save all the pictures in one file or can stitch several pictures together based on geo-referenced mosaicing [63], or can compress the pictures based on a defined quality-rate tradeoff. This PE can then send the processed pictures to all other PEs in the pool so that PU can utilize the PEs in the pool interchangeably. The robots discard the old pictures when they receive the processed ones from the elected PE. When there are large numbers of robots participating in a rescue mission, there can be multiple nodes elected (i.e., region heads) to process the pictures. These region heads can communicate to coordinate their efforts.

When the user at a collection point (i.e., the PU) would like to engage the service of receiving processed pictures from the pool it invokes the DSRP mechanism for picture

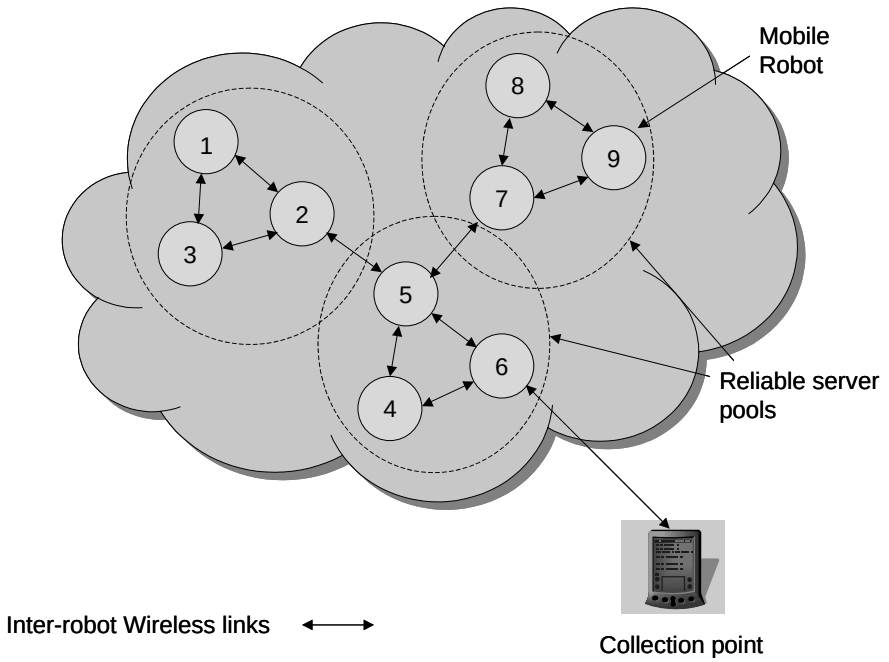

Figure 6.4: A simplified example of DSRP with 9 robots and 3 pools

collection. Transparent to the PU, the DSRP at the collection point selects one of the PEs to download the processed pictures. In other words, the PU does not have any knowledge of the PEs, but only requests a service from the DSRP pool. One of the PEs from the pool will be assigned for providing the processed pictures to the PU. During the transmission, if the connection error rate worsens due to mobility, interference or the lost connectivity, DSRP automatically switches to one of the other PE robots in the pool and continues to download the pictures from the point left from the previous PE. DSRP mechanism accomplishes this fail-over transparent to the user at the collection point. DSRP will then remove the unavailable robot from the pool list, sharing this information with the other PE robots in the pool. When the robot, which left the pool, returns back to the pool area or the communication obstruction is no longer there, it will be added into the

pool membership list as one of the potential PEs.

Fig. 6.5 shows the architecture of our distributed robotic nodes, referred to as STAR, which implement the DSRP mechanism. The STAR nodes are built on Xilinx ML310 development boards powered by Virtex-II Pro FPGA device which have two on-chip 400 MHz embedded IBM PowerPC 405 (PPC405) processor cores [107]. The processors run Wind River VxWorks 5.5 Real Time Operating System (RTOS) [105] which provides deterministic timing required for time sensitive applications, as well as a small foot print suitable for embedded systems. Wind River also provides a built-in communication protocol stack.

The design employs a processor-centric architecture, where one PPC405 core is used to realize vision functions, another one is used for control and communication, and the FPGA fabric is used for custom logic and interfaces. The blocks inside the FPGA device consist of Xilinx Intellectual Property (IP) Cores and user IP cores. The Xilinx IP cores enable us to use pre-verified, pre-optimized design blocks to implement commonly used functions such as memory management, PCI bus, UART, and JTAG interfaces, thus facilitate the design procedure. The hardware flexibility of FPGA also enables us to design on-chip custom circuit (user IP) for Pulse Width Modulation (PWM) and quadrature encoder modules used to drive motors and thus provide the mobility to STAR nodes. The PPC405 processor cores interconnected on-chip with other re-programmable IP cores through the IBM CoreConnectTM bus architecture which consists of a high-speed PLB

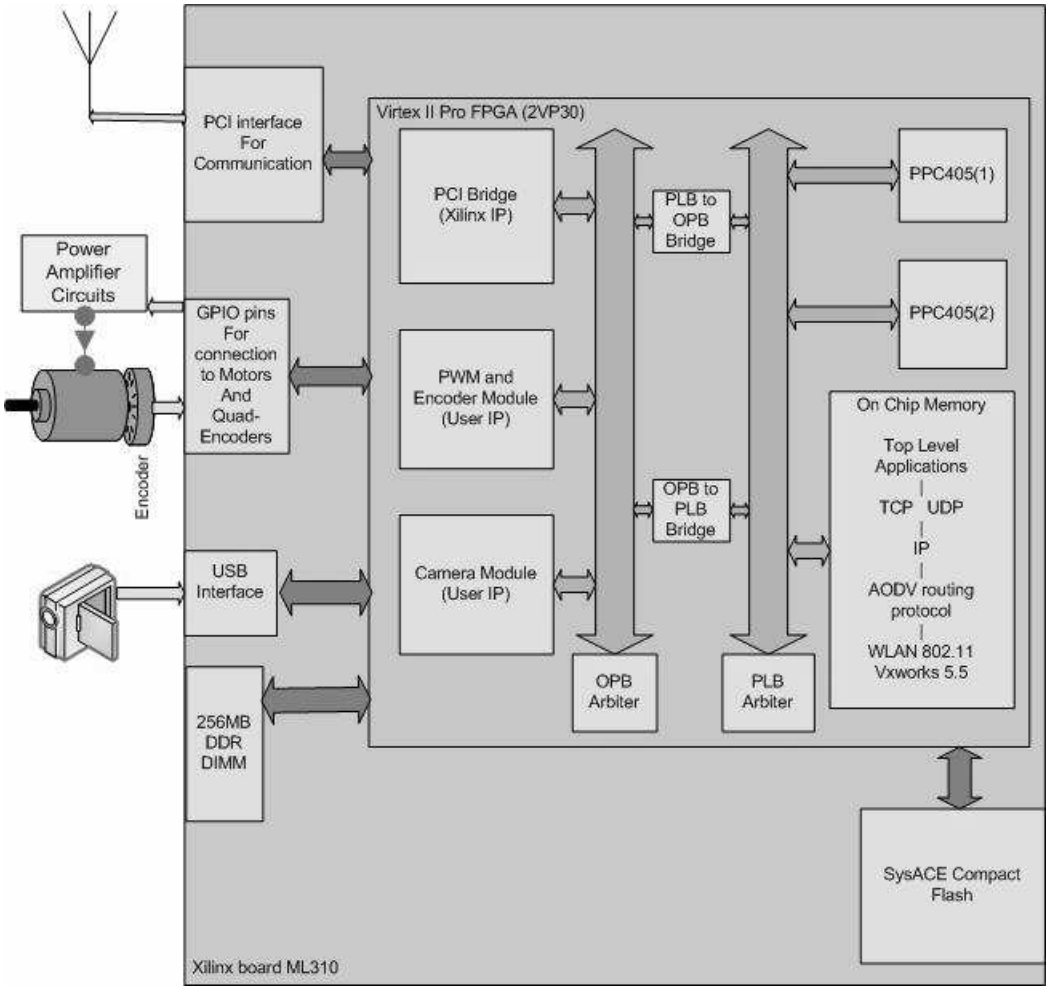

Figure 6.5: Block diagram of FPGA-based STAR architecture

(Processor Local Bus), a general-purpose OPB (On-chip Peripheral Bus), a bus bridge, and two arbiters. The function blocks outside the FPGA chip but within the ML310 board represent the hardware circuit and on-board connectors for interfacing with peripherals and devices, such as compact flash memory, motors, cameras, and wireless communication card.

For the wireless communication module, we currently use the existing resource at the

ML310 board and use the PCI card to establish the wireless communication. For inter-robot communications and for communications between the robots and the collection points several alternatives are possible ranging from a low-power short-range communication technology, such as Bluetooth [45], to a high-power long-range technology such as the IEEE 802.11 wireless LAN [27] protocol. Bluetooth's key features are low complexity, low power and low cost, which operates in the unlicensed ISM band at 2.4GHz, avoiding interference from other signals by hopping to a new frequency after transmitting or receiving a packet. To the best of our knowledge, there is no industry-wide standard hardware and protocol stack currently available to be utilized as a multihop routing mechanism for either Bluetooth or IEEE 802.15.4 in mature embedded platforms such as Xilinx ML310 [107]. For these reasons, we use the IEEE 802.11 wireless LAN protocol for the wireless communication among the robots, in spite of its disadvantages such as relatively larger size and higher power consumption. The current testbed implementation only supports one-hop communication among the mobile robots. We will extend the geographical coverage of our testbed using multihop routing such as *Ad hoc On-Demand Distance Vector (AODV)* [19]. As Bluetooth hardware providing a multihop communication becomes commercially available, we plan to investigate utilizing the price and low consumption features of Bluetooth for communication among the robots.

Fig. 6.6 illustrates an overview of our network protocol stack. In our current design, the VB and DSRP mechanisms are implemented as a sublayer between the transport

and the application layers. All VB and DSRP control messages are transported over User Datagram Protocol (UDP) since these control messages are broadcasted and the UDP is more suitable for this purpose. The application messages are also transported over UDP since the dynamics of Transmission Control Protocol (TCP) is not suitable for the image retrieval application (e.g., the image retrieval application can tolerate packet lost but can not tolerate long delays). As reliability of the periodic control messages is mandatory for the success of the VB formation and DSRP service registrations and requests, we added a TCP-like retransmission mechanism to make sure that the packet is delivered to the destination node which is in the communication range. Unlike TCP, our retransmission mechanism does not include a complex rate control and is controlled by the DSRP mechanism (i.e., the UDP is used as it is but the upper layer provides reliable delivery).

6.3 Image Retrieval Application

To illustrate DSRP capabilities to achieve reliable and survivable service delivery in distributed robotics, a proof-of-concept for a search-and-rescue application described in Section III is implemented in our FPGA-based robotics system.

Fig. 6.7 shows the high level steps of this image retrieval application. Each PE, implemented using Xilinx ML310 FPGA board, runs the DSRP program, as shown in Fig.6.6,

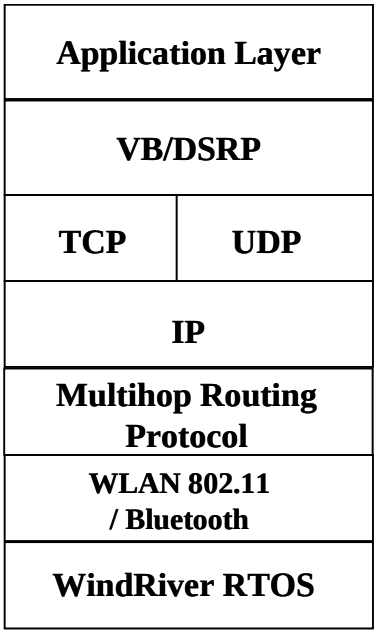

Figure 6.6: Network protocol stack implementing DSRP

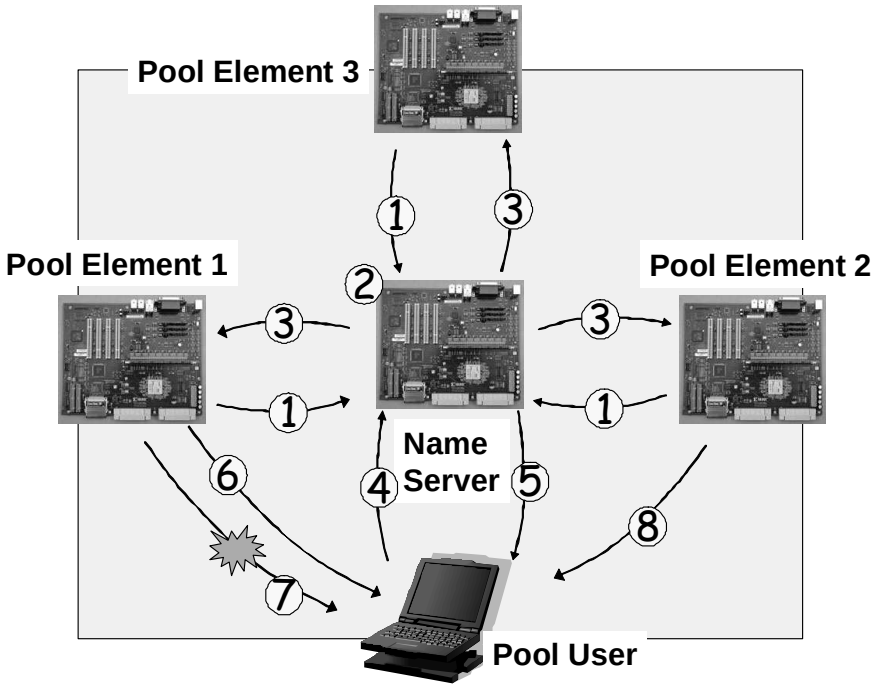

Figure 6.7: Image retrieval application

written in C++ using Wind River VxWorks 5.5 real-time operating system. To demonstrate the feasibility of using DSRP mechanism, first an image retrieval application is started by the PU at the collection point using one of the PEs (PE1 in Fig. 6.7). Then the communication between the PU at the collection point and PE1 is disrupted, emulating an obstacle. As part of the reliable and survivable service, the image retrieval application continues to download images using another pool member (PE2 in Fig. 6.7) without re-starting the application.

In this implementation, first the VB is formed, NS is found, and the PEs register to the NS. Then the image retrieval application using DSRP proceeds as follows:

1. PEs take pictures and upload them to NS
2. NS combined the pictures into a single file
3. NS sends the compressed file to PEs
4. PU requests a list of PEs
5. PU gets a list of PEs
6. PU starts downloading from PE1
7. The link between PU and PE1 is broken (implemented by blocking the packets between PE1 and PU)
8. PU switches over to PE2 and downloads the remaining part of the compressed file

Figure 6.8: Partial views of a street - 4 out of 24 pictures shown

Figure 6.9: Stitched images form a panoramic view of the rescue site

Fig. 6.8 shows the individual pictures taken by the PE robots that represent the STAR nodes, each with a partial view of the scene. The PU can form a panoramic view of the scene only after receiving the pictures taken by all three PEs in the pool. If DSRP mechanism was not used in this example, either the downloading will be restarted several times, depending on the mobility of the PE robots in the scene or the obstructions in the field that prevent wireless communications. Another significant advantage of DSRP is the ability for a PU to select the best PE satisfying the QoS requirements. Otherwise, the PE with the worst connection to the PU will constitute a bottleneck for the entire application.

In our example, the panoramic view formed by the PU combining the pictures received from the image retrieval pool is shown in Fig. 6.9. A commercial image stitching software [33] is used in the PU to construct this panoramic view.

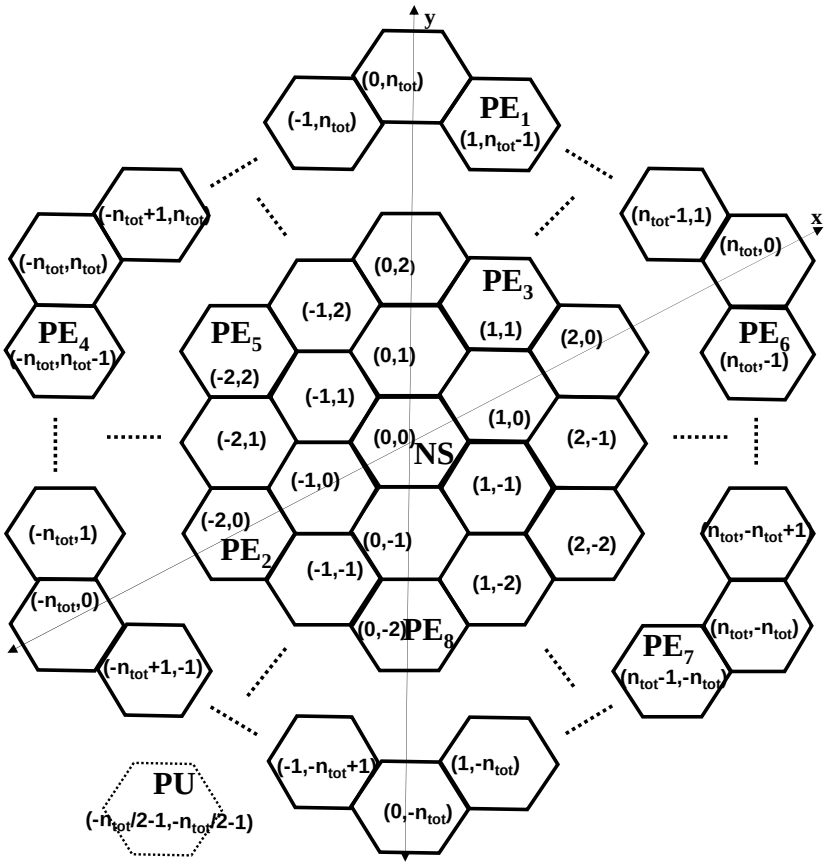

Figure 6.10: Hexagonal geographic area with 1 NS, 1 PU, and 8 mobile PEs

6.4 Analytic Models

In a discrete-time random walk model, we introduced new models to predict the state of a wireless link between two nodes in MANETs in Chapters 3 and 4, where an element of a state transition matrix represents the probability to transit from one link state to another within one time unit. Here, we adapt our earlier generic MANET models to an image retrieval application in distributed robotics systems.

In our approach, all PEs are mobile and follow the discrete-time random walk mobility

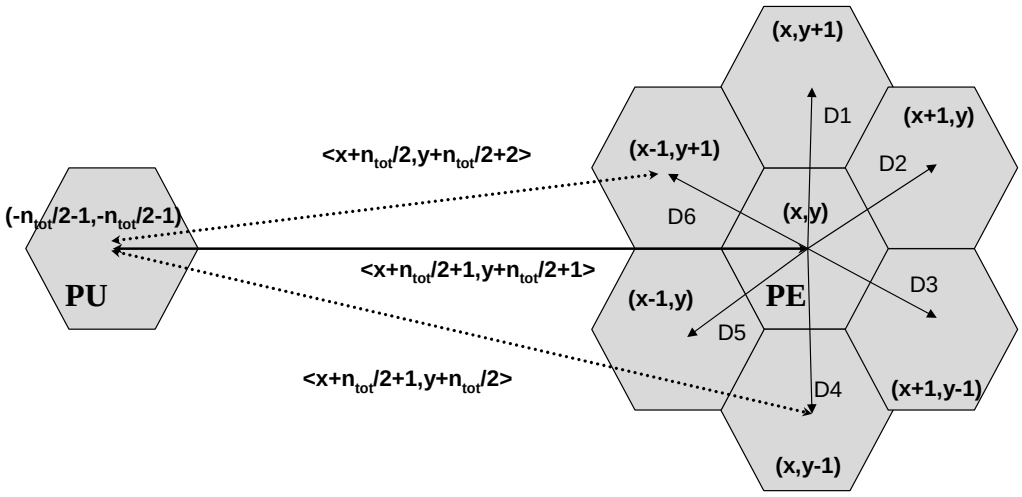

Figure 6.11: Link state change

model while a single PU is static and link availability from this PU to any mobile PE is calculated. For example, a PE can move into six different directions within a certain geographic area which is partitioned into logical hexagonal cells (Fig. 6.10), where n_{tot} determines the network size (i.e., the radius of the hexagonal area in terms of layers). A coordinate pair (x,y) is assigned to each hexagonal cell similar to the cartesian coordinate system (i.e., the coordinates of the center cell is $(0, 0)$). A vector representing a wireless link between a static PU and a mobile PE is called a *state*. In Fig. 6.11, for a PU in location $(-n_{tot}/2-1, -n_{tot}/2-1)$ and a mobile PE in location (x,y), the link state between these nodes is $< x + n_{tot}/2 + 1, y + n_{tot}/2 + 1 >$ (i.e., $< x - (-n_{tot}/2 - 1), y - (-n_{tot}/2 - 1) >$). A random walk model is applied to formulate how a wireless link changes states, where PE moves into one of its six neighbor cells with probability of 1/6 for each direction after one time unit (i.e., discrete-time random walk). For example, if a PE moves into the

neighboring cell in the direction of $D4$, the wireless link state between these nodes will be $< x + n_{tot}/2 + 1, y + n_{tot}/2 >$ in the next time unit. However, if a PE moves into the direction of $D6$, the wireless link state $< x, y >$ will become $< x + n_{tot}/2, y + n_{tot}/2 + 2 >$. If a PE is located in one of the boundary cells, there is less than six possible neighboring cells (i.e., a PE can move only within a given geographic area) and this fact is taken into account during the construction of the link state transition matrix M, where each element $M_{i,j}$ represents the probability to transit from the i-th to the j-th link state within one time unit.

```
Inputs: x, y
Outputs: L_{x,y}

1  function [L_{x,y}] = findLayer(x, y)
2     if ((x < 0 & y < 0) || (x >= 0 & y >= 0)) {
3        x_e = max(abs(x), abs(y))
4        y_e = min(abs(x), abs(y))
5     }
6     else {
7          x_e = max(abs(x + y), min(abs(x), abs(y)))
8          y_e = min(abs(x + y), min(abs(x), abs(y)))
9     }
10    L_{x,y} = x_e + y_e
11 end
```

Figure 6.12: Pseudocode for `findLayer` function

The transmission range of the PU, currently resident in cell $(-n_{tot}/2 - 1, -n_{tot}/2 - 1)$, will be modeled by the number of layers that it can reach. A wireless link state from the PU to a mobile PE is *available* if $L_{x,y} \leq n_{av}$, otherwise it is *unavailable*, where $L_{x,y}$

represents the corresponding length, in terms of layers, of a link state $< x, y >$ and n_{av} represents the maximum number of layers for the communication range. $L_{x,y}$ is calculated using the `findLayer` function given in Fig. 6.12. For example, if n_{tot} and n_{av} are equal to 4 in Fig. 6.12 (n_{tot} and n_{av} are the same due to the negative coordinates), the link state between the PU and PE_8 is available (i.e., this link state is $< 3, -1 >$, where $L(3, -1) = 3 < n_{av} = 4$) whereas the link state between the PU and PE_4 is unavailable (i.e., this link state is $< -1, 6 >$, where $L(-1, 6) = 6 > n_{av} = 4$).

A link state from the PU to a mobile PE changes with PE's random movements, and hence the availability and unavailability of this link may switch from one to another. The expected time it takes for a node to change its status from available to unavailable (or vice versa) provides a valuable intermediate result to be used for determining the download completion time for the image retrieval types of applications. The first passage time analysis is applied to M for calculating two important metrics: *(i)* the expected first time that an available link becomes unavailable, and *(ii)* the expected time that an unavailable link becomes available. If the DSRP mechanism is utilized, the PU downloads 8 pictures from 8 PEs if at least one PE is within one-hop communication range of the PU (i.e., at least one link is available from PU to PEs). If the DSRP is not used, all 8 PEs must be within the communication range of the PU at some point in time until their pictures are downloaded individually (only one-hop communication is possible). Our analytic models can be used to evaluate these types of applications, where a PU downloads the pictures

from PEs with and without using DSRP.

6.4.1 First passage time from available link to unavailable

First, we assume that the states of M partitioned into *available* and *unavailable* states:

$$M = \begin{pmatrix} M_{a,a} & M_{a,u} \\ M_{u,a} & M_{u,u} \end{pmatrix}$$

where a and u are respectively the subsets of *available* and *unavailable* states; let the numbers of states in these subsets be respectively n_a and n_u. Here $M_{a,a}$ is the square matrix of order n_a obtained from M by deleting the rows and columns that correspond to the states in u.

We seek to compute the expected time to move from *available* states to any *unavailable* state for the first time. Therefore, all the unavailable states of M can be aggregated into a single unavailable state by modifying the associated probabilities as follows:

$$K = \begin{pmatrix} M_{a,a} & M_{a,u}e \\ \frac{\pi_u}{\|\pi_u\|_1} M_{u,a} & \frac{\pi_u}{\|\pi_u\|_1} M_{u,u}e \end{pmatrix}$$

where e is a column vector of ones with appropriate length. The number of states in M is $n_a + n_u$ before the state aggregation while the number of states is $n_a + 1$ in the modified version of M. This aggregation of states will alleviate the cost of calculating the

expected time, and hence result in a computationally efficient algorithm even for large number of states for realistic distributed robotics systems. Using the algorithm given in Ref. [28], the mean first passage time from an *available* state i_m to the *unavailable* state i_{n_a+1} is $f^{(1)}_{i_m,i_{n_a+1}}$. Let p is the stationary vector of M and be partitioned conformally as in $p = (p_a p_u)$, where p_a and p_u are the subvectors of p associated with available and unavailable states, respectively. The expected time that an available link becomes unavailable:

$$\overline{T}_{unav} = \sum_{m=1}^{n_a} \frac{p_a(i_m)}{\|p_a\|_1} \times f^{(1)}_{i_m,i_{n_a+1}} \tag{6.1}$$

where $p_a(i_m)$ is the probability that the available link state is i_m initially for $1 \leq m \leq n_a$ and $\|p_a\|_1$ normalizes the length of the subvector p_a to 1. Here, there are n_a different available link states (i.e., i_m) and $f^{(1)}_{i_m,i_{n_a+1}}$ is most likely different for each i_m. Therefore, the weighted average of the expected first times moving from available link states to any unavailable link state for particular network parameters is calculated by considering the steady state distributions of available states in Eq. (6.1).

6.4.2 First passage time from unavailable link to available

In this case, the expected first time moving from unavailable link states to an available link state is calculated. For M which partitioned into available and unavailable link states, we combine all available link states into a single available state and apply the first passage time analysis on the modified M for calculating the expected time to move from

Table 6.1: Expected times that available and unavailable link becomes unavailable (T_{unav}) and available (T_{av}), respectively, for different n_{tot} values

n_{tot}	4	6	8	10	12	14	16	18	20	22
$\overline{T}_{unav}$	8.25	19.40	34.97	54.98	79.40	108.24	141.49	179.15	221.22	267.71
$\overline{T}_{av}$	37.09	61.96	93.48	131.60	176.29	227.54	285.35	349.70	420.61	498.06

n_{tot}	24	26	28	30	32	34	36	38	40
$\overline{T}_{unav}$	318.60	373.90	433.61	497.72	566.25	639.18	716.52	798.26	884.41
$\overline{T}_{av}$	582.06	672.61	769.70	873.34	983.52	1100.25	1223.52	1353.34	1489.70

unavailable link states to an available link state for the first time. The modified M has $n_u + 1$ states. Using the algorithm given in Ref. [28], the mean first passage time from an unavailable state i_m to the available state i_{n_u+1} is $f^{(1)}_{i_m,i_{n_u+1}}$ (i.e., all available states are represented by the available state i_{n_u+1}). Then, the expected time that an unavailable link state becomes available:

$$\overline{T}_{av} = \sum_{m=1}^{n_u} \frac{p_u(i_m)}{\|p_u\|_1} \times f^{(1)}_{i_m,i_{n_u+1}} \tag{6.2}$$

where $p_u(i_m)$ is the probability that the unavailable link state is i_m initially for $1 \leq m \leq n_u$ and $\|p_u\|_1$ normalizes the length of the subvector p_u to 1.

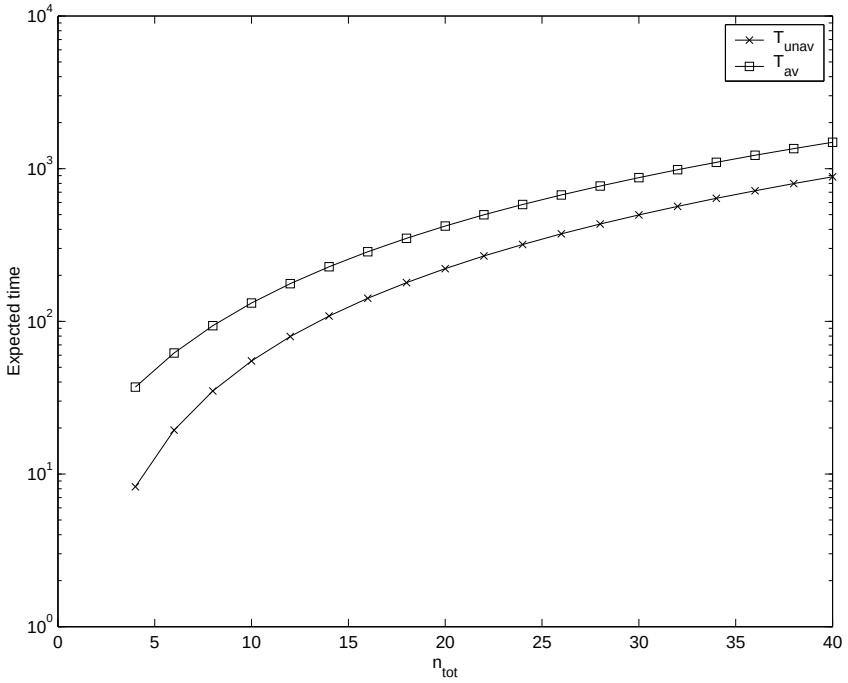

Figure 6.13: Numerical results of $\overline{T}_{av}$ and $\overline{T}_{unav}$ for different n_{tot} values

6.4.3 Numerical Results

In this section, we provide numerical results for characterizing $\overline{T}_{av}(n_{tot}, n_{av})$ (the expected first time that an *unavailable* link becomes *available*) and $\overline{T}_{unav}(n_{tot}, n_{av})$ (the expected first time that an *available* link becomes *unavailable*). $\overline{T}_{av}$ and $\overline{T}_{unav}$ are shown for different n_{tot} values in Table 6.1 and Fig. 6.13, where the numerical results for the expected passage times are presented using the logarithmic scale in the y-axis. Here, n_{tot} has been changed from 4 to 40 for even numbers, whereas the n_{av} value is set to n_{tot} for all experiments.

The results show that the expected passage times heavily depend on the network size

(i.e., n_{tot}), where both T_{av} and T_{unav} increases by the network size. For example, from Table 6.1, $\overline{T}_{av}$ values are 131.60, 420.61, and 1489.70 whereas $\overline{T}_{unav}$ are 54.98, 221.22, and 884.41 for n_{tot} values of 10, 20, and 40, respectively.

Fig. 6.13 shows that T_{av} is always higher than T_{unav} for all n_{tot} values. This behavior can be explained by the fact that the number of unavailable link states is always higher than the number of available states for any n_{tot} value in Fig. 6.10. The probability of remaining unavailable is higher for an unavailable link state than the probability of remaining available for an available link state due to the number of desired link states in both cases (e.g., the desired states are unavailable links for an initially unavailable link and available links for an initially available link). Therefore, it is expected that moving from an unavailable link state to available takes longer than moving from an available link state to unavailable. The ratio of T_{av} to T_{unav} decreases when n_{tot} increases (i.e., the ratio is greater than 4 when $n_{tot} = 4$ whereas it is less than 2 when $n_{tot} = 40$). This result can be explained by the fact that the ratio of the number of available link states to the number of unavailable link states changes when n_{tot} changes.

6.5 Testbed Measurements

We implemented the DSRP framework, together with its architecture and protocols, in a distributed robotics environment at the City College of the City University of New York

(CCNY). The current testbed implementation includes a total of 22 nodes emulating real-life mobile robots, where two nodes are FPGA boards and the remaining ones are laptop and desktop PCs. The measurements collected from the testbed confirm that the DSRP framework significantly improve the system reliability for the image retrieval application.

The image retrieval application uses two algorithms:

(i) Using DSRP: To improve the system reliability of an image retrieval application in distributed robotics environment as described in Section 6.3, the same compressed file containing all pictures from all PEs is created in each PE. The PU starts downloading the compressed file from the best PE among its PE neighbors. The optimal allocation of the pooled servers is studied in [58]. In our study, the best PE is defined as the PE having the closest distance to the PU since all the wireless cards are equipped with the same capabilities. If the wireless cards have different capabilities, the PE providing the highest bandwidth can be selected as the best PE. Whenever the selected PE moves away from the PU's communication range, the PU will switch over transparently to another PE if there is one. The most significant two advantages for using DSRP in this types of image retrieval applications are: *(i)* using the best PE for downloading to the PU, which eliminates the waiting time for the bottleneck PE since all pictures are needed for a panoramic view of the disaster site, and *(ii)* significantly increasing the system reliability by creating a pool of the PEs that can upload the same compressed file.

(ii) Without using DSRP: In this case, the PU downloads the picture file of a PE

when this PE is within its communication range. The downloading task is complete when all the picture files from all PEs are downloaded individually. This case requires that each PE should be within the PU's communication range for at least the amount of time necessary for downloading an individual picture file.

An instant of our mobile testbed topology is shown in Fig. 6.14, where one static NS (e.g., a laptop) is located in the center, one static PU (e.g., a desktop) is located in the left corner, and eight mobile PEs (e.g., laptops, desktops, and FPGA boards) are distributed randomly over the given geographic area. The virtual locations of these nodes are assigned from the hexagonal network shown in Fig. 6.10, where the center cell is (0,0), the PU's location is $(-n_{tot}/2-1, -n_{tot}/2-1)$, and each PE's location is randomly selected as one from the cells within this area. In this configuration, NS has always direct communication to all PEs whereas PU is not within the NS's communication range. A mobile PE has an active communication with the PU only if the distance between them is within the communication range. For example, in the instant of our mobile testbed topology shown in Fig. 6.14, PU can only directly communicate with PE2. The PE mobility is realized by roaming PEs randomly following the discrete-time random walk model described in Section 6.4 (Fig. 6.11). Depending on the distance between the PU and a PE, the wireless communication between these two nodes is either enabled or disabled.

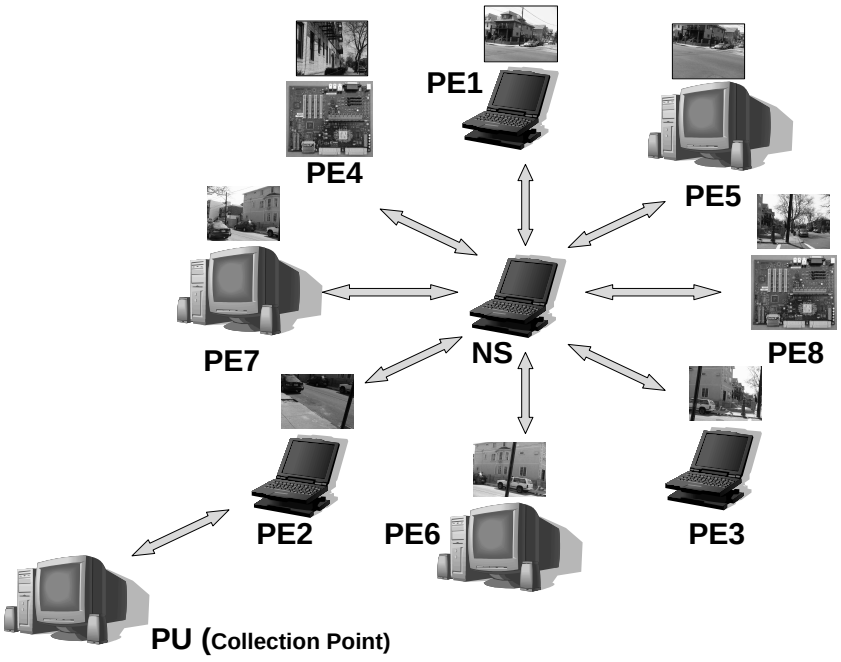

Figure 6.14: An instant of CCNY mobile testbed topology with one NS, one PU, and eight PEs

6.6 Testbed Measurements and Interpretations

In our experiments, we measure the download completion times for the cases with and without the DSRP. The download completion time is defined as the interval starting when the PU sends its request for the set of picture files and ending at the completion of the downloading. The download completion times are obtained for different speeds, network sizes, number of PEs, and data payload sizes.

Table 6.2: Pool setup times for DSRP (Each file size is 28K)

Number of PEs	1	2	4	6	8
Pool setup times (seconds)	62	95	152	227	289

6.6.1 Pool Setup Times

Table 6.2 shows the pool setup times for the DSRP case, where it takes longer to create the PE pool when the number of PEs is higher. For example, the pool setup time is 152 seconds for 4 PEs whereas it takes 289 seconds to setup the pool for 8 PEs. This result is expected since the compressed file contains the number of picture files as many as the number of PEs (i.e., the compressed file size is larger when the pool have larger number of PEs). Another observation is that, when each individual file size increases, the pool setup time for the same number of PEs increases. For example, for the DSRP mechanism with 8 PEs, the pool setup time is 289 seconds when each file size is 28K whereas it takes 791 and 1106 seconds for 70K and 100K, respectively.

6.6.2 Download Completion Time with and without DSRP

The download completion times for the cases with and without DSRP are shown in Fig. 6.15, where the PU downloaded the same compressed file 85 times using these two methods. There are high variations among the download completion times for the case without DSRP, whereas the download time remains steady with the DSRP. The reason

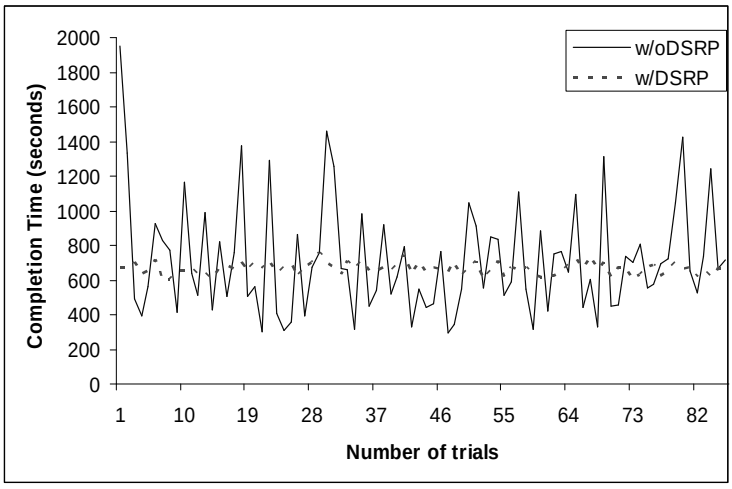

Figure 6.15: Download completion time (seconds) with and without DSRP

for large differences in the completion time in the cases without DSRP is that all PEs at some point are required to be within the PU's communication range to complete the download task. Since the initial PE locations and their movement patterns are randomly selected, it may take a long time for a PE to be within the PU's communication range and, therefore the download completion time can be high.

The mean completion times (i.e., the mean result of 85 experiments) are 664.8 and 714.2 seconds for the case with and without DSRP, respectively. The mean completion time for the DSRP cases includes the pool setup time, which is 289 seconds. For an image retrieval application, where a set of updated pictures is needed periodically, the DSRP mechanism will setup a new pool with updated pictures simultaneously as the PU is downloading the current pool's compressed file. Hence, the presented results are an upper bound for the DSRP cases.

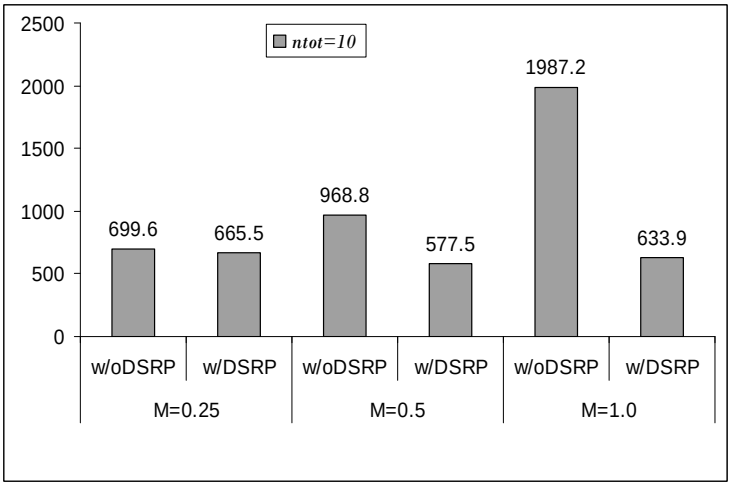

Figure 6.16: Speed effect on mean download completion time (seconds) from 20 experiments with and without DSRP for n_{tot}=10

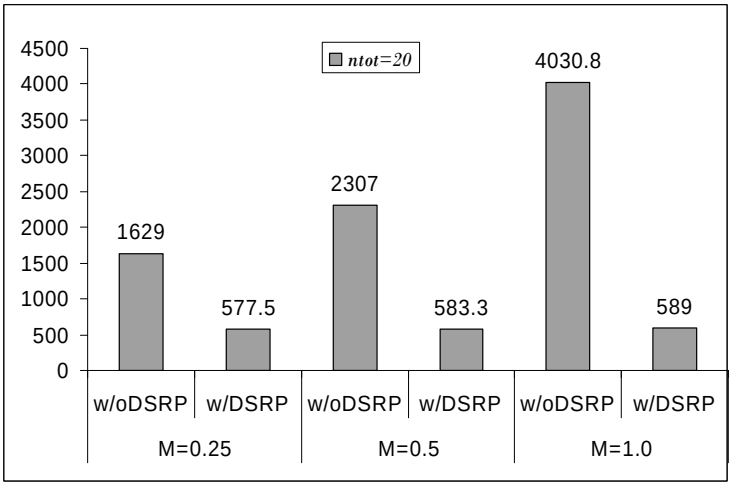

Figure 6.17: Speed effect on mean download completion time (seconds) from 20 experiments with and without DSRP for n_{tot}=20

6.6.3 Effect of Mobility

In this section, the effect of the PE speeds on the mean download completion time over 20 repeated experiments is measured for three different movement intervals, namely M=0.25,

0.5, and, 1.0. One single movement of a PE represents a movement from PE's current cell to one of the neighboring cells according to the random walk described in Section 6.4 (Fig. 6.11). $M = 0.25$ means that a PE moves to the neighboring cell in 0.25 seconds, which implies that four such movements are done within one second time interval. Similarly, M=1.0 represents the case of PEs moving four times slower than the case of M=0.25.

We observe that, without DSRP, the PU completion time is directly proportional to their speeds. For example, in Fig. 6.16, for PEs with M=0.25 (i.e., faster PEs), the downloading takes 699.6 seconds, whereas it takes 1987.2 seconds for PEs with M=1.0 (i.e., slower PEs). This observation is expected since faster (slower) moving PEs will be more (less) likely to come back to the PU's range (i.e., one-hop connectivity) after they loose connectivity with the PU.

With the DSRP, we observe a very different behavior. The first observation is that the system performance is much improved compared to the case without the DSRP. In addition, we do not observe a significant difference in system performance due to different PE speeds as was the case without the DSRP. Fig. 6.16 shows that all PEs with different speeds complete the task within 665.5 seconds. In other words, it takes almost the same amount of time to complete the download for all PEs regardless of their speeds.

Both observations are direct results of the DSRP mechanism which allows that even if one PE is within the range of the PU, the downloading task can proceed. These measurements clearly illustrate that the system reliability has improved by the DSRP both by eliminating

negative effect of slower PEs and by shortening the download completion time.

Measurements shown in Fig. 6.16 (for $n_{tot} = 10$) are repeated for a larger geographic area as given in Fig 6.17 (for $n_{tot} = 20$). The system behavior for $n_{tot} = 20$ is similar to the case of $n_{tot} = 10$, where slower PEs take longer to complete download without DSRP, and employing DSRP significantly improves the performance and eliminates the negative effects of different speeds (i.e., the almost same mean download completion time). However, comparing Figs. 6.16 and 6.17 show that the mean download completion time for the case without DSRP in larger area is approximately twice of the case with a smaller area.

6.6.4 Effect of Geographic Area

In this section, the effect of the geographic area size on the mean download completion time over 20 repeated experiments is observed, where three different network sizes of n_{tot}= 10, 20, and 40 represent small, medium and large network sizes, respectively. Recall that n_{tot} is the radius of the hexagonal area, in terms of layers, as shown in Fig. 6.10.

The first observation is that, without DSRP, it always takes longer to download the compressed file in large size networks than in smaller networks. For example, in Fig. 6.18, for M=0.5, the task completion takes 968.8 seconds for the network with n_{tot}=10, while it takes 3260.4 seconds for n_{tot}=40. This observation is intuitive since there are more out

of range cells for a PU in larger networks, and therefore, it takes longer for a PE to come one-hop connectivity range of the PU. This fact is also confirmed by our numerical results presented in Section 6.4.3 (Fig. 6.13).

The DSRP mechanism significantly reduces the mean download completion time compared to the case without DSRP. In addition, we observe that, with the DSRP, the network size does not have a significant effect on the mean download completion time as opposed to the case without DSRP. Fig. 6.18 shows that the PU completes the downloading task within 615.3 seconds for all network sizes.

The experiments are repeated for the slower PEs (M=1.0) as given in Fig 6.19. The system behavior for M=1.0 is similar to the case of M=0.5, where it always takes shorter to download the compressed file for the case with DSRP in the same size network. Also, the mean download completion time is directly proportional to the network size for the case without DSRP. One important observation is that, with the DSRP, the mean completion times for the cases of M=0.5 and 1.0 remain almost the same.

6.6.5 Effect of Number of PEs

To measure the effect of the number of PEs on the mean download completion time, the number of PEs in the testbed is varied as 2, 4, 6 and 8. For all cases, n_{tot} and M are kept constant as 20 and 0.5, respectively.

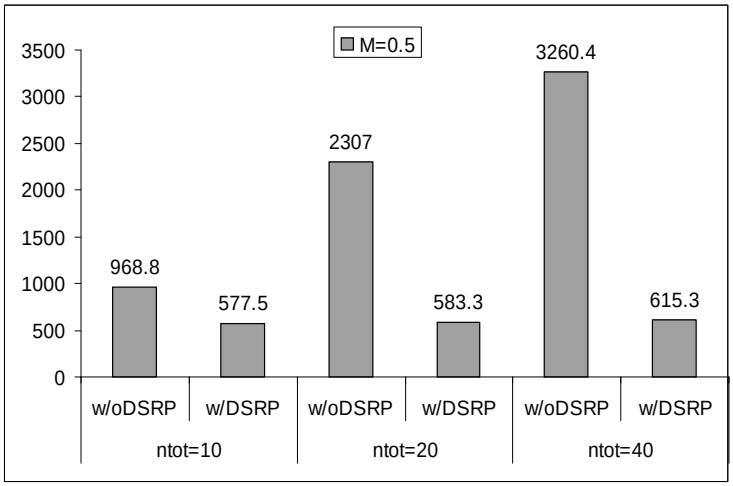

Figure 6.18: Network size effect on mean download completion time (seconds) from 20 experiments with and without DSRP for M=0.5

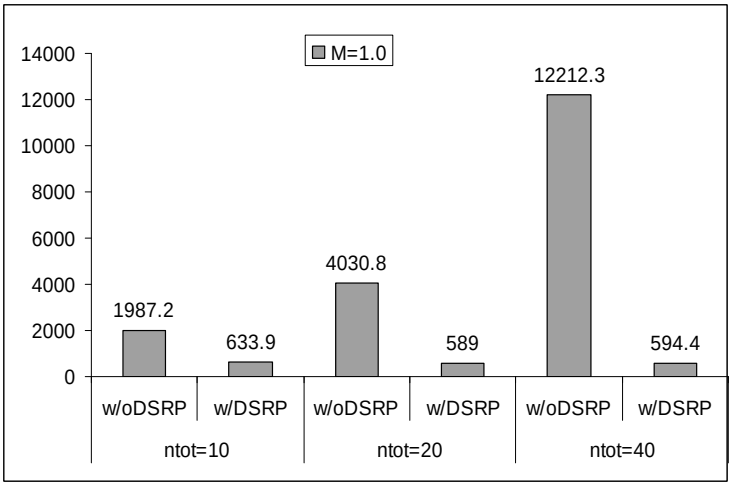

Figure 6.19: Network size effect on mean download completion time (seconds) from 20 experiments with and without DSRP for M=1.0

Let us first consider the cases without DSRP. From Fig. 6.20, the mean download completion time increases when the number of PEs increases. For example, the mean download completion time is measured as 796.1 seconds for the pool with 2 PEs, whereas 2307 seconds for the pool with 8 PEs. This result can be explained by *(i)* the increase in the number of picture files from 2 to 8 (i.e., a higher number of packets transmitted during both the pool setup and downloading the compressed file), and *(ii)* the reduced probability that all 8 PEs will be within the PU's communication range until their pictures are downloaded compared to the pool with 2 PEs (the lower probability implies the higher expected time for 8 PEs).

With the DSRP, the mean download completion times are very small compared to the without DSRP case. For example, the downloading task is completed within 279.5, 295.1, and 583.3 seconds for 2, 4, and 8 PEs, respectively whereas it was within 796.1, 1469.4, and 2307 seconds for the without DSRP case. An interesting measurement is that, without the pool setup times, the mean download completion time for the pool with 4 PEs (143.1 seconds) is less than the pool with 2 PEs (184.5 seconds). This result is less intuitive since the compressed file size is larger in the pool with 4 PEs. However, the probability that one PE will be within the PU's communication range is smaller in the pool with 2 PEs than the pool with 4 PEs. The DSRP mechanism requires that at least one PE is within the PU's communication range otherwise the PU will wait for until a PE reaches its range. As a result, the pool with 4 PEs will have a shorter completion time than the

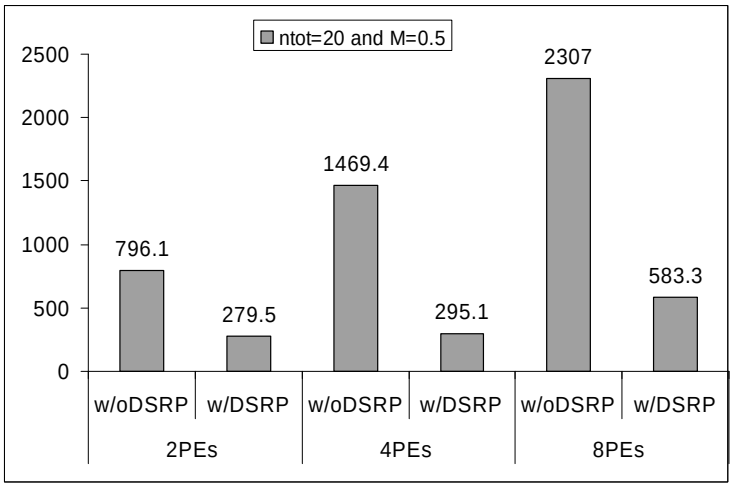

Figure 6.20: Effect of number of PEs on mean download completion time (seconds) from 20 repeated experiments with and without DSRP for n_{tot}=20 and M=0.5

pool with 2 PEs excluding the pool setup time.

Chapter 7

Conclusions and Future Research Directions

In this thesis, we introduce new analytical models based on novel Markov Chains whose states represent node degree changing with multiple link arrivals and departures, and number of link failures. These new models allows the computations of two important metrics characterizing the dynamics of a node's random movement, namely the expected time for the number of link changes (i.e., instances of link creation and failure) and the expected time for the node degree to drop below or exceed a threshold. We obtained the two-dimensional Markov chain, $P2$, which help us to calculate the desired expected first times for both $nlff$ and the node degree thresholds.

These first analytic models use steady state approximation of link creation and failure probabilities to study node link stability in wireless mobile networks with different node densities. We also introduce a second set of analytic models which are capable of analyzing node link stability for more realistic MANET applications by calculating the exact link failure and creation probabilities. Another potential limitation of the first analytic models is the use of random walk mobility model which allows a node to roam into its neighboring cells with only a fixed speed. However, in real-life scenarios, a node may move in any direction with any speed (i.e., random way point) or remain at the same point. Our new comprehensive models enhance the random walk mobility pattern to cover the fast/slow node movements with different speeds. This extension will allow our model to analyze more realistic scenarios in MANETs where speed variation affects the system performance. This work will be extended in the future to cover the cases where the nodes do not move in the next time step.

To verify the correctness of the analytic model, we conducted randomized simulations to obtain the mean node degree and the expected first times using the same discrete-time random walk model. For the mean node degree, we observed that simulation and analytic results match, where mean node degree decreases as the network density decreases. For the expected first times, the simulation results support the numerical calculations from our model especially around the mean node degree, where the expected first time increases with d_{thr} in both cases. We conclude that the d_{thr} range which our analytic model

accurately predicts the expected first times is around the mean node degree. Note that the degrees which are very small or large compared to the mean node degree are typically less of an interest for network designers. The future work will include two degree threshold values, one for lower and another for upper limits, to model realistic wireless networks, where each node can not have more than a certain number of neighbors.

Our proposed analytic models are scalable for realistic mobile networks, computationally efficient and relatively simple to apply, and, hence can be used to evaluate the performance of algorithms and protocols at different layers of MANETs. Our model can analyze the performance of various paradigms using VB algorithms such as backbone- or cluster-assisted routing protocols and service discovery architectures for applications such as reliable server pooling [102] in mobile ad-hoc networks. In forming a VB in MANETs, the most stable nodes with higher node degree are selected as the backbone nodes. Virtual-backbone formation algorithms apply node degree [59] or both node degree and *nlff* [61] metrics to resolve conflicts between the nodes competing to join the backbone. Using the mean degree as the criteria for joining or leaving the backbone, the numerical results show that expected times to stay within a VB for a given node are approximately the same for networks with different densities. Another interesting result is that, for a given network density, expected times for joining and leaving the backbone are approximately the same. The interference level which changes with dynamics of nodes' movements can be calculated by our model. We plan to extend our modelling framework to derive a

number of additional metrics that characterize network connectivity and convergence. In addition, some elements of SIR will be adapted as part of the input to our analytic models.

To perform optimizations in a dynamic network, PILSNER agents rely on information that is dispersed in the network. Controlled Dissemination Filter (CDF) enhances a data dissemination service for the agents through the efficient filtering of data along the dissemination tree. Current work focuses on evaluating tradeoffs involved in application of the CDF to information flows relevant to the PILSNER URCA, augmenting CDF with techniques developed in the DSRP architecture, and addressing security issues inherent in the data dissemination process: (1) preventing malicious filters to receive unauthorized data or to block data from intended recipients, (2) trust model and enhanced subscription architecture for security within CDF, and (3) security-based filters where certain nodes may not be authorized to receive data.

To improve the survivability and reliability in distributed robotics systems, we implement the DSRP mechanism in our FPGA-based distributed robotics testbed. DSRP is based on a VB, which is a highly distributed, scalable, and survivable network, formed and maintained through one-hop beacons among mobile robots. We demonstrate the effectiveness of DSRP in a search-and-rescue scenario, where robots cooperate to provide the rescuer with a set of pictures to build a panoramic view of the disaster site. We introduce new analytic models to evaluate the performance of these applications in distributed robotics

systems. The measurements collected from the testbed and the numerical results obtained from the analytic models confirm that the DSRP framework significantly improve the system reliability.

The future work will concentrate on further investigating the robustness, efficiency, and scalability of the DSRP mechanism in distributed robotics environments. Another important study area is the optimization of various distributed algorithms running at each robot to form the VB, to locate services among robots, and to keep updates among different domains. This research will define new metrics to quantify the performance of a distributed robotics system, and, hence, it will identify the key performance components. Also we plan to develop a high-performance, highly adaptive computational module integrating robotic functions such as control, sensor processing, communication, and on-board vision into a single FPGA chip.

Chapter 8

References

[1] M. Adler, Z. Ge, J. Kurose, D. Towsley, and S. Zabele. Channelization problem in large scale data dissemination. In *Proc. IEEE Int'l Conf. Netw. Protocols (ICNP)*, Riverside, CA, 2001.

[2] C. Aguero, V. Matellan, P. Quiros, and J. Canas. PERA: ad-hoc routing protocol for mobile robots. In *Proc. Int'l Conf. Advance Robot. (ICAR)*, Portugal, 2003.

[3] I.F. Akyildiz, Y.B. Lin, Y.R. Lai, and R.J. Chen. A new model for random walks in PCS networks. *IEEE J. Select. Areas Commun.* **18**(7), pp. 1254–1261, 2000.

[4] D. N. Alparslan and K. Sohraby. A generalized random mobility model for wireless ad hoc networks and its analysis: One-dimensional case. *IEEE/ACM Trans. Netw.* , 2006. (to appear).

[5] L. Alvisi, T.C. Bressoud, A. El-Khashab, K. Marzullo, and D. Zagorodnov. Wrapping server-side TCP to mask connection failures. In INFOCOM'01 [86].

[6] K. Alzoubi, P. Wan, and O. Frieder. Distributed heuristics for connected dominating set in wireless ad hoc networks. *KICS Journal of Communications and Networks* **4**(1), 2002.

[7] S. O. Anderson, R. Simmons, and D. Goldberg. Maintaining line of sight communications networks between planetary rovers. In *Proc. IEEE/RSJ Int'l Conf. Intel. Robot. Syst. (IROS)*, pp. 2266–2272, Las Vegas,Neveda, 2003.

[8] T. Anjali, C. Scoglio, and G. Uhl. A new scheme for traffic estimation and resource allocation for bandwidth brokers. *[Elsevier] Comput. Netw.* **41**, pp. 761–777, 2003.

[9] R. C. Arkin and J. Diaz. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *Proc. AMC Int'l Wksp Advance Motion Control*, 2002.

[10] P. Basu, N. Khan, and T.D.C. Little. A mobility based metric for clustering in mobile ad hoc networks. In *Proc. Int'l Wksp Wirel. Netw. Mob. Comput. (WNMC)*, Scottsdale, AZ, 2001.

[11] S. Bergbreiter and K.S. J. Pister. CotsBots: An off-the-shelf platform for

distributed robotics. In *Proc. IEEE/RSJ Int'l Conf. Intel. Robot. Syst. (IROS)*, pp. 1632–1637, Las Vegas, Neveda, 2003.

[12] P. Bremaud. *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues.* Springer-Verlag, New York, NY, 1999.

[13] W. Burgard, M. Moorsy, C. Stachniss, and F. Schneidery. Coordinated multi-robot exploration. *IEEE Trans. Robot. Automat.* **21**(3), pp. 376–386, 2005.

[14] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. In Basagni and Lee, eds, *Mobile Ad Hoc Networking—Research, Trends and Applications (S.I.)*, *[Wiley] Wirel. Commun. Mob. Comput.* **2**(5), pp. 483–502. 2002.

[15] F. Cao and J. P. Singh. Efficient event routing in content-based publish-subscribe service networks. In *Proc. IEEE INFOCOM*, **2**, pp. 929–940, 2004.

[16] Y. Cao, A. Fukunaga, and A. Kahng. Cooperative mobile robotics: antecedents and directions. *[Springer] Auton. Robot.* **4**(1), pp. 1–23, 1997.

[17] S. Carpin and L. E. Parker. Cooperative leader following in a distributed multi-robot system. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, pp. 2994–3001, Washington DC, 2002.

[18] L. Chaimowicz, M. F. M. Campos, and V. Kumar. Dynamic role assigment for cooperative robots. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, pp. 293–298, Washington DC, 2002.

[19] I. D. Chakeres and E. M. Belding-Royer. AODV routing protocol implemen-

tation design. In *Proc. Int'l Wksp Wirel. Ad Hoc Netw. (WWAN)*, Tokyo, Japan, 2004.

[20] W. Chen and D. Towsley. PRIEST: A private subscription scheme in publish-subscribe systems. Technical report no. 04-81, University of Massachusetts Amherst Computer Science, Boston, MA, 2004.

[21] Y.P. Cheny, A.L. Liestmany, and J. Liu. Clustering algorithms for ad hoc wireless networks. In Pan and Xiao, eds, *Ad Hoc and Sensor Networks*, *Nova Sci. Pub.* 2004.

[22] C.-Y. Chiu, G.-H. Chen, and E.H.-K. Wu. A stability aware cluster routing protocol for mobile ad hoc networks. In Tseng et al., eds, *Research in Ad Hoc Networking, Smart Sensing and Pervasive Computing (S.I.)*, *[Wiley] Wirel. Commun. Mob. Comput.* **3**(4), pp. 503–515. 2003.

[23] P.T. Conrad and P. Lei. Services provided by reliable server pooling. Internet draft, IETF, 2002. [draft-conrad-rserpool-service, work in progress].

[24] Cornell University. *Robocup*. (http://robocup.mae.cornell.edu/).

[25] M. S. Corson and A. Ephremides. A distributed routing algorithm for mobile wireless networks. *ACM/Baltzer Wirel. Netw. J.* , 1995.

[26] S. Corson and J. Macker. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations. RFC 2501, IETF, 1999.

[27] B. P. Crow, I. Widjaja, J.G. Kim, and P.T. Sakai. IEEE 802.11 wireless local area networks. *IEEE Communications* , 1997.

[28] T. Dayar and N. Akar. Computing moments of first passage times to a subset of states in markov chains. *SIAM Journal on Matrix Analysis and Applications* **27**(2), 2005.

[29] A. Doufexi, S. Armour, M. Butler, A. Nix, D. Bull, J. Mcgeehan, and P. Karlsson. A comparison of the HIPERLAN/2 and IEEE 802.11a wireless LAN standards. *IEEE Communications* **40**(5), pp. 172–180, 2002.

[30] A. Drenner, I. Burt, D. Goerke, B. E. Kratochvil, and N. Papanikolopoulos. COTS scout robot. In *Proc. SPIE Int'l Symp. Unmanned Ground Vehicle Technol.*, Orlando, Florida, 2003.

[31] A. Drenner, I. Burt, B. Kratochvil, B. Nelson, N. Papanikolopoulos, and K. B. Yesin. Communication and mobility enhancements to the scout robot. In *Proc. IEEE/RSJ Int'l Conf. Intel. Robot. Syst. (IROS)*, Lausanne, Switzerland, 2002.

[32] R. Dube. Signal stability based adaptive routing (SSA) for ad-hoc mobile networks. *IEEE Personal Commun.* , 1997.

[33] Eos Systems Inc. *PhotoModeler Pro.* (http://www.photomodeler.com/index.html).

[34] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.* **35**(2), pp. 114–131, 2003.

[35] M.A. Fecko, İ. Hökelek, S. Samtani, and A. Staikos. Controlled dissemination filter (CDF) for integrated link selection agents. In *IEEE/Create-Net COMSWARE*, 2007. (submitted for publication).

[36] M.A. Fecko, U.C. Kozat, S. Samtani, M.U. Uyar, and İ. Hökelek. Architecture and applications of dynamic survivable resource pooling in battlefield networks. In *Battlespace Digitization and Network-Centric Systems IV*, *Proc. SPIE* **5441**, pp. 204–214. (SPIE, Bellingham, WA), 2004.

[37] ———. Dynamic survivable resource pooling in mobile ad hoc networks. In *Proc. IEEE Int'l Symp. Comput. Commun. (ISCC)*, pp. 196–201, Alexandria, Egypt, 2004.

[38] ———. Reliable and dynamic access to services in battlefield ad hoc networks. In MILCOM'04 [68], pp. 1015–1020. **(invited paper)**.

[39] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proc. IEEE INFOCOM*, pp. 519–528, Tel Aviv, Israel, 2000.

[40] S. Galli, H. Luss, A. Mcauley, S. Samtani, and J. Sucec. A novel approach to OSPF-area design for large wireless ad-hoc networks. In *Proc. IEEE Int'l Conf. Commun. (ICC)*, Seoul, 2005.

[41] M Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad-hoc wireless networks. *IEEE/ACM Trans. Netw.* **10**(4), 2002.

[42] J. A. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, C. Mitter, , and B. Heile. IEEE 802.15.4: A developing standard for low-power low-cost wireless personal area networks. *IEEE Netw.* **15**, pp. 12–19, 2001.

[43] O. Madsen H. Schioler, L. Son and J. Nielson. Communicating cooperative robots with bluetooth. In *Proc. IEEE Wirel. Personal Multimedia Com-*

mun. (WPMC), Aalborg, Denmark, 2001.

[44] J. Haartsen, M. Naghshineh, J. Inouye, O. Joeressen, and W. Allen. Bluetooth: Vision, goals, and architecture. *ACM Mob. Comput. Commun. Rev.* **2**(4), pp. 38–45, 1998.

[45] J.C. Haartsen. The bluetooth radio system. *IEEE Wirel. Commun.* **7**(1), pp. 28–36, 2000.

[46] F. Hao, E.W. Zegura, and M.H. Ammar. Supporting server selection in differentiated service networks. In INFOCOM'01 [86].

[47] J. Heidemann, F. Silva, and D. Estrin. Matching data dissemination algorithms to application requirements. In *ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2003.

[48] J. L. Hill. *System architecture for wireless sensor networks*. PhD thesis, University of California, Berkeley. (in progress).

[49] D. F. Hougen, S. Benjaafar, J. C. Bonney, J. R. Budenske, M. Dvorak, M. Gini, H. French, D. G. Krantz, P. Y. Li, F. Malver, B. Nelson, N. Papanikolopoulos, P. E. Rybski, S. A. Stoeter, R. Voyles, and K. B. Yesin. A miniature robotic system for reconnaissance and surveillance. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, pp. 501–507, San Francisco, CA, 2000.

[50] D. F. Hougen, M. D. Erickson, P. E. Rybski, S. A. Stoeter, M. Gini, and N. Papanikolopoulos. Autonomous mobile robots and distributed exploratory missions. In *Proc. Int'l Symp. Distrib. Auton. Robot. Syst. (DARS)*, Knoxville, TN,

2000.

[51] E. Hyytiä and J. Virtamo. Random waypoint mobility model in cellular networks. *[Springer] Wirel. Netw. (WINET)* , 2006. (to appear).

[52] B. Jackson, I. Burt, B. E. Kratochvil, and N. Papanikolopoulos. A control system for teams of off-the-shelf scouts and megascouts. In *Proc. IEEE Mediterranean Conf. Control Automat.*, Rhodes, Greece, 2003.

[53] C. Jaikaeo and C.-C. Shen. Adaptive backbone-based multicast for ad hoc networks. In *Proc. IEEE Int'l Conf. Commun. (ICC)*, New York, NY, 2002.

[54] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad-hoc wireless networks. *[Kluwer Acad. Pub.] Mob. Comput.* **353**, 1996.

[55] E. Kadioglu and N. Papanikolopoulos. A method for transporting a team of miniature robots. In *Proc. IEEE/RSJ Int'l Conf. Intel. Robot. Syst. (IROS)*, Neveda, 2003.

[56] G. Karumanchi, S. Muralidharan, and R. Prakash. Information dissemination in partitionable mobile ad hoc networks. In *Proc. IEEE Symp. Reliab. Distrib. Syst. (SRDS)*, pp. 4–13, Lusanne, Switzerland, 1999.

[57] T. Korkmaz and M. Krunz. Hybrid flooding and tree-based broadcasting for reliable and efficient link-state dissemination. In *Proc. IEEE GLOBECOM*, Taipei, Taiwan, 2002.

[58] U.C. Kozat, M.A. Fecko, and S. Samtani. Optimal allocation of pooled servers for situation awareness applications. In *Proc. IASTED Int'l Conf. Commun.*

Netw. (CCN), Marina del Rey, CA, 2005.

[59] U.C. Kozat, G. Kondylis, B. Ryu, and M.K. Marinay. Virtual dynamic backbone for mobile ad hoc networks. In *Proc. IEEE Int'l Conf. Commun. (ICC)*, Helsinki, Finland, 2001.

[60] U.C. Kozat and L. Tassiulas. Network layer support for service discovery in mobile ad hoc networks. In *Proc. IEEE INFOCOM*, San Francisco, CA, 2003.

[61] ———. Service discovery in mobile ad hoc networks: An overall perspective on architectural choices and network layer support issues. *[Elsevier] Ad-Hoc Netw.* **2**(1), pp. 23–44, 2004.

[62] B. E. Kratochvil, I. T. Burt, A. Drenner, D. Goerke, B. Jackson, C. McMillen, C. Olson, N. Papanikolopoulos, A. Pfeifer, S. A. Stoeter, K. Stubbs, and D. Waletzko. Heterogeneous implementation of an adaptive robotic sensing team. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, Taipei, Taiwan, 2003.

[63] R. Kumar, H. Sawhney, J. Asmuth, J. Pope, and S. Hsu. Registration of video to geo-referenced imagery. In *Proc. IAPR Int'l Conf. Pattern Recognition (ICPR)*, **2**, pp. 1393–1400, 1998.

[64] J. Liu, Q. Zhang, B. Li, W. Zhu, and J. Zhang. A unified framework for resource discovery and QoS-aware provider selection in ad hoc networks. *ACM Mob. Comput. Commun. Rev.* **6**(2), pp. 13–21, 2002.

[65] J. Liu, Q. Zhang, W. Zhu, and B. Li. Service location for large-scale mobile

ad hoc networks. *[Kluwer] Wirel. Netw. (WINET)* **10**(1), pp. 33–40, 2003.

[66] J. Loughney, M. Stillman, Q. Xie, and R. Stewart. Comparison of protocols for reliable server pooling. Internet draft, IETF, 2002. [draft-ietf-rserpool-comp, work in progress].

[67] K. Manousakis, J.S. Baras, A.J. McAuley, and R. Morera. Rate of degradation of centralized optimization solutions and its application to high performance domain formation in ad hoc networks. In MILCOM'04 [68].

[68] *Proc. IEEE Military Commun. Conf. (MILCOM)*, Monterey, CA, 2004.

[69] N. Miyata, J. Ota, T. Arai, and H. Asama. Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment. *IEEE Trans. Robot. Automat.* **18**(5), pp. 769–780, 2002.

[70] S. Murthy and J.J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Netw. App. J.* , 1996. Special Issue on Routing in Mobile Communication Networks.

[71] K.J. O'Hara and T.R. Balch. Pervasive sensorless networks for cooperative multi-robot task. In *Proc. Int'l Symp. Distrib. Auton. Robot. Syst. (DARS)*, pp. 291–300, Toulouse, France, 2004.

[72] M. Gini D. Hougen P. Rybski, S. Stoeter and N. Papanikolopoulos. Performance of a distributed robotic system using shared communications channels. *IEEE Trans. Robot. Automat.* **18**(5), pp. 713–727, 2002.

[73] O. Papaemmanouil and U. Cetintemel. SemCast: Semantic multicast for

content-based data dissemination. In *Int'l Conf. Data Eng. (ICDE)*, Tokyo, Japan, 2005.

[74] L. E. Parker. Current state of the art in distributed autonomous mobile robotics. *Distrib. Auton. Robot. Syst.* **49**, p. 312, 2000.

[75] J. L. Pearce, P. E. Rybski, S. A. Stoeter, and N. Papanikolopoulos. Dispersion behaviors for a team of miniature robots. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, Taipei, Taiwan, 2003.

[76] C. E. Perkins, ed. Addison-Wesley, 2001.

[77] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *[Elsevier] Comput. Commun.* , 1994.

[78] M. Powers and T. Balch. Value-based communication preservation for mobile robots. In *Proc. Int'l Symp. Distrib. Auton. Robot. Syst. (DARS)*, Toulouse, France, 2004.

[79] I. Rekleitisy, V. Shuey, A. New, and H. Choset. Limited communication, multi-robot team based coverage. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, New Orleans, LA, 2004.

[80] E. M. Royer and C. Toh. A review of current routing protocols for ad-hoc movile wireless networks. *IEEE Personal Commun.* , 1999.

[81] P. Rybski, S. Stoeter, M. Erickson, M. Gini, D. Hougen, and N. Papanikolopoulos. A team of robotic agents for surveillance. In *Proc. Int'l Conf.*

Auton. Agent. (Agents 2000), pp. 9–16, Barcelona, Spain, 2000.

[82] P. E. Rybski, F. Zacharias, J. F. Lett, O. Masoud, M. Gini, and N. Papanikolopoulos. Building topological maps with sensor-limited miniature mobile robots. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, Taipei, Taiwan, 2003.

[83] P.E. Rybski, A. Larson, H. Veeraraghavan, M. LaPoint, and M. Gini. Communication strategies in multi-robot search and retrieval: Experiences with minDART. In *Proc. Int'l Symp. Distrib. Auton. Robot. Syst. (DARS)*, pp. 301–310, Toulouse, France, 2004.

[84] P. Sass and J. Freebersyser. FCS communications: Technology for the objective force. In *Proc. SPIE AeroSense*, Orlando, FL, 2002.

[85] H. Schulzrinne and E. Wedlung. Application-layer mobility using SIP. *ACM Mob. Comput. Commun. Rev.* **1**(5), 1999.

[86] B. Sengupta, R. Cruz, and G. Pacifici, eds. *Proc. IEEE INFOCOM*, Anchorage, Alaska, 2001.

[87] R. Shah, Z. Ramzan, R. Jain, R. Dendukuri, and F. Anjum. Efficient dissemination of personalized information using content-based multicast. *IEEE Trans. Mob. Comput.* **3**(4), pp. 394–408, 2004.

[88] G. Sibley, M. Rahimi, and G. Sukhatme. Robomote: A tiny mobile robot platform for large-scale ad hoc sensor networks. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, Washington DC, 2002.

[89] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun,

and H. Younes. Coordination for multi-robot exploration and mapping. In *Proc. Nat'l Conf. Artificial Intel.*, Austin, TX, 2000.

[90] R. Sivakumar, P. Sinha, and V. Bharghavan. CEDAR: A core-extraction distributed ad hoc routing algorithm. In Haas et al., eds, *Wireless Ad Hoc Networks (S.I.)*, *IEEE J. Select. Areas Commun.* **17**(8), pp. 1454–1465. 1999.

[91] S. A. Stoeter, F. Le Mauff, and N. P. Papanikolopoulos. Real-time door detection in cluttered environments. In *Proc. IEEE Int'l Symp. Intel. Control*, pp. 187–192, Rio, Greece, 2000.

[92] J. Sucec, K. Chang, J. Lee, H. Tanna, S. Samtani, L. Muzzelo, J. Palumbo, and M. Bereschinsky. A resource friendly approach for estimating available bandwidth in secure mobile wireless IP networks. In *Proc. IEEE Military Commun. Conf. (MILCOM)*, Atlantic City, NJ, 2005.

[93] J. Sucec, K. Chang, and S. Samtani. Bandwidth broker call admission control and management in support of HAIPE. Technical report, Telcordia Technologies, Inc., Piscataway, NJ, 2004.

[94] F. Sultan, K. Srinivasan, D. Iyer, and L. Iftode. Migratory TCP: Highly available Internet services using connection migration. In *Proc. IEEE Int'l Conf. Distrib. Comput. Syst. (ICDCS)*, Vienna, Austria, 2002.

[95] J. Sweeney, T. Brunette, Y. Yang, and R. Grupen. Coordinated teams of reactive mobile platforms. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, pp. 299–304, Washington DC, 2002.

[96] B.J. Thibodeau, A.H. Fagg, and B.N. Levine. Signal strength coordination for cooperative mapping. Technical report no. 04-64, University of Massachusetts Amherst Computer Science, Boston, MA, 2004.

[97] C.-K. Toh. Associativity-based routing for ad hoc mobile networks. *[Kluwer] Wirel. Personal Commun.* **4**(2), pp. 103–139, 1997.

[98] Y.-C. Tseng, Y.-F. Li, and Y.-C. Chang. On route lifetime in multihop mobile ad hoc networks. *IEEE Trans. Mob. Comput.* **2**(4), pp. 236–276, 2003.

[99] M. Tuexen, Q. Xie, R. Stewart, M. Shore, L. Ong, J. Loughney, and M. Stillman. Architecture for reliable server pooling. Internet draft, IETF, 2001. [draft-ietf-rserpool-arch, work in progress].

[100] ———. Requirements for reliable server pooling. RFC 3237, IETF, 2002.

[101] M.U. Uyar, J. Zheng, M.A. Fecko, and S. Samtani. Reliable server pooling in highly mobile wireless networks. In *Proc. IEEE Int'l Symp. Comput. Commun. (ISCC)*, pp. 627–632, Kemer-Antalya, Turkey, 2003.

[102] M.U. Uyar, J. Zheng, M.A. Fecko, S. Samtani, and P.T. Conrad. Evaluation of architectures for reliable server pooling in wired and wireless environments. In Li et al., eds, *Recent Advances in Service Overlay Networks (S.I.)*, *IEEE J. Select. Areas Commun.* **22**(1), pp. 164–175. 2004.

[103] R.M. Voyles. Terminatorbot: A robot with dual-use arms for manipulation and locomotion. In *Proc. IEEE Int'l Conf. Robot. Automat. (ICRA)*, pp. 61–66, San Francisco, CA, 2000.

[104] A. Wagner and R. Arkin. Multi-robot communicationsensitive reconnaissance. In *Proc. IEEE/RSJ Int'l Conf. Intel. Robot. Syst. (IROS)*, pp. 4674–4681, New Orleans, LA, 2004.

[105] Wind River Systems, Inc. *VxWorks 5.x.* (http://www.windriver.com/products/).

[106] J. Wu and F. Dai. A distributed formation of a virtual backbone in MANETs using adjustable transmission ranges. In *Proc. IEEE Int'l Conf. Distrib. Comput. Syst. (ICDCS)*, pp. 372–379, Tokyo, Japan, 2004.

[107] Xilinx, Inc. *ML310 development board.* (http://www.xilinx.com/products/boards/).

[108] E.W. Zegura, M.H. Ammar, Z. Fei, and S. Bhattacharjee. Application-layer anycasting: A server selection architecture and use in a replicated Web service. *IEEE/ACM Trans. Netw.* **8**(4), pp. 455–466, 2000.