ABSTRACT

Title of Dissertation: DESIGN OPTIMIZATION AND SECURITY FOR COMMUNICATION NETWORKS

Mehdi Kalantari Khandani,
Doctor of Philosophy, 2005

Dissertation directed by: Professor Mark Shayman
Department of Electrical & Computer Engineering

In this work we introduce a new mathematical tool for optimization of routes, topology design, and energy efficiency in wireless sensor networks. We introduce a vector field formulation that models communication in the network, and routing is performed in the direction of this vector field at every location of the network. The magnitude of the vector field at every location represents the density of amount of data that is being transited through that location. We define the total communication cost in the network as the integral of a quadratic form of the vector field over the network area.

With the above formulation, we introduce a mathematical machinery based on partial differential equations very similar to the Maxwell's equations in electrostatic theory. We show that in order to minimize the cost, the routes should be found based on the solution of these partial differential equations.

In our formulation, the sensors are sources of information, and they are similar to the positive charges in electrostatics, the destinations are sinks of information and they are similar to negative charges, and the network is similar to a non-homogeneous dielectric media with variable dielectric constant (or permittivity coefficient).

In one of the applications of our mathematical model based on the vector fields, we offer a scheme for energy efficient routing. Our routing scheme is based on changing the permittivity coefficient to a higher value in the places of the network where nodes have high residual energy, and setting it to a low value in the places of the network where the nodes do not have much energy left. Our simulations show that our method gives a significant increase in the network life compared to the shortest path and weighted shortest path schemes.

Our initial focus is on the case where there is only one destination in the network, and later we extend our approach to the case where there are multiple destinations in the network. In the case of having multiple destinations, we need to partition the network into several areas known as regions of attraction of the destinations. Each destination is responsible for collecting all messages being generated in its region of attraction. The complexity of the optimization problem in this case is how to define regions of attraction for the destinations and how much communication load to assign to each destination to optimize the performance of the network. We use our vector field model to solve the optimization problem for this case. We define a vector field, which is conservative, and hence it can be written as the gradient of a scalar field (also

known as a potential field). Then we show that in the optimal assignment of the communication load of the network to the destinations, the value of that potential field should be equal at the locations of all the destinations.

Another application of our vector field model is to find the optimal locations of the destinations in the network. We show that the vector field gives the gradient of the cost function with respect to the locations of the destinations. Based on this fact, we suggest an algorithm to be applied during the design phase of a network to relocate the destinations for reducing the communication cost function. The performance of our proposed schemes is confirmed by several examples and simulation experiments.

In another part of this work we focus on the notions of responsiveness and conformance of TCP traffic in communication networks. We introduce the notion of responsiveness for TCP aggregates and define it as the degree to which a TCP aggregate reduces its sending rate to the network as a response to packet drops. We define metrics that describe the responsiveness of TCP aggregates, and suggest two methods for determining the values of these quantities. The first method is based on a test in which we drop a few packets from the aggregate intentionally and measure the resulting rate decrease of that aggregate. This kind of test is not robust to multiple simultaneous tests performed at different routers. We make the test robust to multiple simultaneous tests by using ideas from the CDMA approach to multiple access channels in communication theory. Based on this approach, we introduce tests of responsiveness for aggregates, and call it CDMA based Aggregate Perturbation Method (CAPM). We use CAPM to perform congestion control. A

distinguishing feature of our congestion control scheme is that it maintains a degree of fairness among different aggregates.

In the next step we modify CAPM to offer methods for estimating the proportion of an aggregate of TCP traffic that does not conform to protocol specifications, and hence may belong to a DDoS attack. Our methods work by intentionally perturbing the aggregate by dropping a very small number of packets from it and observing the response of the aggregate. We offer two methods for conformance testing. In the first method, we apply the perturbation tests to SYN packets being sent at the start of the TCP 3-way handshake, and we use the fact that the rate of ACK packets being exchanged in the handshake should follow the rate of perturbations. In the second method, we apply the perturbation tests to the TCP data packets and use the fact that the rate of retransmitted data packets should follow the rate of perturbations. In both methods, we use signature based perturbations, which means packet drops are performed with a rate given by a function of time. We use analogy of our problem with multiple access communication to find signatures. Specifically, we assign orthogonal CDMA based signatures to different routers in a distributed implementation of our methods. As a result of orthogonality, the performance does not degrade because of cross interference made by simultaneously testing routers. We have shown efficacy of our methods through mathematical analysis and extensive simulation experiments.

Design Optimization and Security For Communication Networks

by

Mehdi Kalantari Khandani

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Dedication

To My Parents

Acknowledgments

I would like to express my heartfelt gratitude to my advisor, Professor Mark Shayman, for his support, patience, and encouragement throughout my PhD studies. It have been very fortunate to have an advisor that always finds the time for listening to the countless problems that I faced in the course of the research. His technical advice was significant to the completion of this dissertation and has taught me many valuable lessons and insights on the workings of academic research.

My thanks also go to the members of my committee, Professor Samrat Bhattacharjee, Professor Armand Maowski, Professor Prakash Narayan, and Professor Sennur Ulukus, for providing many valuable comments that improved the technical material of this dissertation.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction on Sensor Networks

Wireless sensor networks have been studied extensively in recent years. Such networks are made of several hundred to several thousand sensors distributed in a geographical area. There are many different applications for such networks including military, environment monitoring, agriculture and home applications. Sensors are very simple identical electronic devices equipped with a processor and small memory, a transmitter, and a receiver. Generally, sensors use radio frequency channels for the purpose of communication.

In many applications the sensors perform measurements of specific metrics such as temperature, pressure, movement or other physical values in a periodic or non-periodic way. Most of the time it is desired to send the data of all sensors to a specific destination for processing, archiving and other purposes. This station is a data sink, and it has enough processing power, storage space, and capability of communicating with the sensors. For the purpose of communication to this destination, the sensors relay the packets of each other in a multi-hop way.

In this work, we introduce a novel approach to solve the routing problem in a wireless network by formulating it as an optimization problem. We make use of the concept of vector fields to define the communication load at every place of the network and show that this vector field is conservative under certain assumptions. By using this conservative vector field we define a powerful tool for routing by writing the conservative vector field as the gradient of a scalar potential function. The routing of packets to each destination is done based on the value of this potential function on each node and its value on the neighboring nodes. Our primary focus is on the *many-to-one scenario* in which many sources want to send their data toward a single destination. Later, we generalize our approach to the case in which we have several destinations.

We introduce a mathematical machinery based on partial differential equations very similar to the Maxwell's equations in electrostatic theory. The routes are found based on the solution of these partial differential equations. In our formulation, sources of information are similar to the positive charges in electrostatics, the destinations are similar to negative charges, and the network is similar to a non-homogeneous dielectric media with variable dielectric constant (or permittivity coefficient).

As the first application of our methodology, we offer an energy efficient routing scheme for the sensor networks. Our energy efficiency is based on matching the routes to energy constraints in order to increase the network life. When the energy of the sensors in some area of the network is low due to heavy communication activity in the past, we increase the cost of routing through this area to protect the sensors from early energy depletion. Our routing scheme is based on changing the permittivity factor to a higher value in the places in the network where we have a high residual energy of the nodes, and set it to a low value in the places of the network where the nodes do not have much energy left. Our simulations show that our method gives a significant increase in the network life compared to the shortest path and weighted shortest path schemes.

The second application of our methodology is to provide an optimal network clustering algorithm into several regions. Each region has a destination inside it, which the wireless nodes inside that region communicate with. Given the geographic information of the communication demand in an area of the network, we give an optimal method for assigning the load of the network to the different destinations. We mathematically prove that in order to minimize a quadratic cost function of communication load, the value of potential function should be equal in the locations of all destinations.

The third application of our framework is finding optimal locations for different destinations. We prove that relocating a destination in the opposite direction of the gradient of potential function improves a quadratic cost function related to the communication load in the network.

In addition to the mathematical proof of our results, we have verified effectiveness of our methods through several simulation experiments.

1.1 Related Work

The routing problem in sensor networks has been studied by many researchers. Sequential Assignment Routing (SAR) is proposed in [39], and it takes into account the energy constraints by making a tree rooted in the destination. The tree starts to grow toward the sensors on the paths with enough residual energy. The routing from each sensor to the destination is based on the structure of the tree. Minimum Cost Forwarding Algorithm for Large Sensor Networks is proposed in [38]. In this approach, each sensor maintains the least cost from it to the destination. For transmission of a packet, it is broadcast, and after receiving a packet, a sensor checks if it is on the least cost path of the source sensor to the destination. If so, it retransmits that packet. Similar routing schemes can be found in [36] and [37].

Link state and distance vector are the conventional methods for the purpose of routing in the data networks [7]. Several problems like mobility and high link failure rate causes these approaches not to be proper for routing in ad hoc networks if we do not make any modification in them. The main problem of distance vector is its slow convergence [5], and in ad hoc networks with high link failure rate it is not proper to be used in its original form. Link state on the other hand shows a good performance in terms of keeping track of network changes, but it imposes a huge communication overhead by flooding the information of link failure and creation, which happens very frequently in ad hoc networks. Modifications to these methods have been done to make them proper for the wireless ad hoc networks [8][9][10][27]. Fisheye State Routing [8][9] and Ad Hoc On Demand Distance Vector [10] are among the most popular proposed protocols. Fisheye State Routing is a modified version of the link state protocol, and the nodes that are far from a wireless link receive fewer updates about that link. The frequency of update decays exponentially as the the number of hops from the link increases. This causes the very far nodes from a link to only receive update about that link if the link is stable enough.

AODV is a modified version of distance vector and it remedies the problem of increased routing overhead by making it on demand so that a node sends a routing request only if it needs to send data packets. This algorithm has a route discovery phase in which query packets are flooded in the network by the source in order to find a path. Dynamic Source Routing [20] is another alternative for on demand routing. In DSR the source specifies the complete path to the destination in the header of data packets. Optimized Link State Routing Protocol (OLSR) is modified link state protocol and it uses multi-point relay (MPR) sets to reduce the number of routing packets needed to be sent [33][32].

Many protocols have been developed that take into account the location information for the purpose of routing in both wired and wireless networks [28][30][16][24][22][25][26][29][11]. The most popular geometric routing scheme in the wireless networks is greedy routing [21][23][35]. In this scheme every intermediate node tries to forward a packet to a neighbor node that make it closer to the destination. When the number of nodes in the network is large enough, the paths generated by greedy routing are close to the line of sight of the source and the destination. In [26] the authors have given a robustness analysis for the location based routing and its sensitivity to the errors of the location of the destination. GRID routing protocol is proposed in [29]; in this protocol the network is partitioned into geographical grids and each grid is a square. At each grid a grid leader is elected and the routing is done among the grid leaders from the source to the destination.

Intuitively, the greedy routing scheme gives very good results if for every intermediate node there exist a neighbor node that is closer to the destination of a packet. But this is not necessarily the case in general. It may happen that a packet faces a deadend (i.e., a local optimum). The general approach to overcome this problem is face routing [16]. Face routing tries to send a packet not necessarily along the shortest geographical path toward the destination, but it tries to get around the areas of the network that there is no nodes in them that are closer to the destination. Such areas are called faces. The solution is based on extracting a connected planar sub-graph of the original network graph in which every vertex represents a network node and every pair of neighboring nodes have an edge in between in the corresponding graph. The approach in [16] guarantees delivery of the packets to the destination under certain assumptions. In [31], the Grid's Location Service (GLS) is proposed. In GLS each node keeps a set of its neighboring nodes up-to-date about its location information. By using the location information, it tries to do efficient geographical routing.

In Location Aided Routing [11] the routes are established on demand from the source node. The location information is used to limit the nodes that perform the flooding mechanism for route discovery. Distance Routing Effect Algorithm for Mobility [12] also uses the location information for routing. In this approach each node keeps a location table that stores and updates the location of all nodes in the network. Each node periodically sends a message to the other nodes and advertises its location information. The frequency of updating depends on the distance of the nodes and the rate of mobility. The location information is used to relay packets through the nodes that are in the direction of the destination.

GOAFR+ is proposed in [30] and it is the combination of greedy and face routing. In this approach the packet from the source is sent to the destination in a greedy routing scheme. Whenever it reaches a deadend in which there is no neighboring node that makes the packet closer to the destination, it switches to the face routing mode, and it continues in the face routing mode until it escapes from the local minimum.

Some approaches have been offered in which partial information about the node location is assumed to be known [25],[19],[34]. In [19], the concept of location proxy is introduced. A location proxy is a node that has complete information about its location. The routing approach proposed in [19] assumes some of the nodes in the network can serve as location proxies, and these nodes collect and forward the packets of their neighboring nodes that do not have their location information.

Authors of [34] have made a similar set of assumption on the information about the location of the node. In [34], there are a set of nodes with a special placement in the network (e.g., perimeter nodes) that know the exact information about their location. The nodes that do not have their actual location information try to estimate their location based on the information that they get from the nodes that have the actual or estimate of their location.

The structure of sensor networks may vary considerably depending on the application [40]. The sensor nodes might be deployed either in prescribed locations or in an ad hoc way. Although nodes might be stationary or moveable, most of the time their mobility is very limited. Communication in sensor networks is usually multipoint-to-point, as the sensory data flows from the sensors to the sink nodes. However, multicasting and peer-to-peer communication are also necessary in some specific applications, such as in security networks and parking lot networks [42].

In the traditional flat network architecture, all sensor nodes are homogeneous in terms of their communication and processing capability. However, it has been shown that the flat wireless ad hoc network architecture without an infrastructure support is not scalable as the number of nodes per unit area becomes large [43] [44]. Consequently, hierarchical wireless ad hoc network models with additional backbone nodes have been proposed to increase the network connectivity and capacity [45] [46] [47]. The scalability issue is even more crucial for a sensor network, since in general, the number of nodes in a sensor network can be several orders of magnitude higher than in an ad hoc network.

The throughput capacity of wireless networks has been analyzed by several researchers [45][48][49]. In [48], it is shown that the number of base stations must scale by at least $\Theta(\sqrt{n})$, n being the number of wireless nodes, to achieve better scaling of capacity than the flat network architecture. In [49], it is shown that for networks where the ratio of wireless nodes to base stations is bounded above by some constant, the throughput capacity is found as $\Theta(\frac{1}{logn})$ if all nodes choose a common transmission power. In [45], this result is improved by employing power control in the network. We need to emphasize that all these capacity results are obtained under the assumption that backbone nodes (i.e., base stations) are assumed to be connected by a wired network with unlimited capacity which act as relays for wireless nodes.

In a different line of work, the possibility of having a wireless backbone support is considered. As opposed to the wired case, the analysis is mostly focused on routing and topology control mechanisms to enhance the connectivity, fault tolerance, and lifetime of the entire network. In general, a hierarchical wireless sensor network consists of a limited number of special nodes (i.e., backbone nodes) in addition to ordinary wireless sensor nodes, to support the data aggregation and data routing. However, in several problems, sensor nodes can be dynamically grouped into clusters by choosing clusterheads to aggregate data and communicate with the sink [50] [51] [52] In this case, the clusterheads form the so-called virtual backbone as these nodes are chosen from among the regular sensor nodes. Alternatively, they can also be selected from a given set of backbone capable nodes.

The idea of using a routing methodology in sensor networks inspired by the way the electrostatic field propagates in a dielectric medium was first introduced in our works [53], and [54]. Our work was followed by related work by Toumpis and Tassiulas [55], where they have shown that the approach minimizes the number of sensors required to handle the total communication burden of the network. A different approach that uses multipath routing based on electrostatic force is described in [56].

Surveys on the routing schemes of wireless ad hoc networks can be found in [6] and [17]. Similar surveys of the sensor networks have been given in [41] and [40]

Chapter 2

Formulation of Communication Load as A Vector Field

In this chapter, we introduce the main framework of routing problem in sensor networks as an optimization problem. We will use the concept of vector fields for routing in wireless network, and we will show that routing constraints can be written in the form of partial differential equations with specific boundary conditions. Also, we introduce a quadratic cost function to optimize routes in the network, and we will prove that for minimizing that cost function, we need to solve a set of partial differential equations that are analogous to the set of partial differential equations known as Maxwell's equations in the theory of Electrostatics.

2.1 Basic Assumptions and Definitions

Consider a network of N wireless sensors that can communicate with each other through radio links. The nodes are densely located in a region A in the plane, and they are intended to collect information about the events in the area of the network. Each sensor is responsible for events happening in its neighborhood. All messages are desired to be collected in a destination node (access point), and for now, we assume there is only one node of this type in the network. Later, we generalize our approach to the case in which we have multiple destinations. When an event is generated at some place in the network, the closest sensor to location of the event generates a message. All messages should be sent to the destination, which is assumed to have enough storage, energy and processing power. Furthermore, we make the following assumptions:

A1: The messages in the geographical area of the network happen with a known spatial density rate denoted by $r(x, y) \geq 0$ for the place (x, y). $r(x, y)$ states how many messages are generated per unit of area per unit of time. We call this quantity the *load density*, which means that for the area $a \subseteq A$ the rate of messages generated inside a is:

$$w(a) = \int_a r(x, y)\, dxdy \tag{2.1}$$

in which integration is over area a. It is important to note that $r(x, y)$ does not include messages passing through a on their path to the destination. The units of $r(x, y)$ are messages per square meter per second.

A2: For the purpose of routing, a single-valued direction is kept for every point

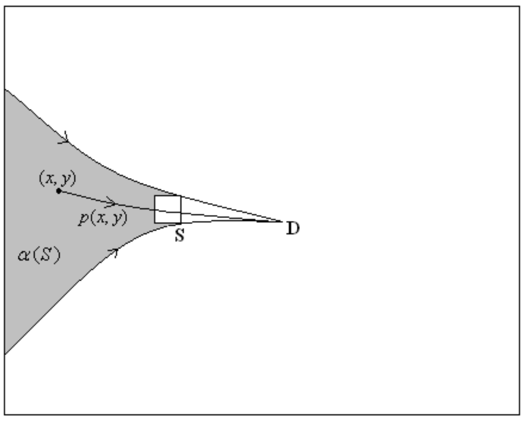

Figure 2.1: The illustration of *Upstream Area* of a given set S. The Shaded area in this figure shows the upstream area of the rectangle shown as S

(x, y) of the network. At place (x, y), this quantity represents the direction in which a message travels along the forwarding sensors from its source sensor to the destination; the message may be generated at place (x, y), or received from an upstream place of the network. For place (x, y) of the network, we denote this direction by the *unit* vector $\hat{\theta}(x, y)$. The value of $\hat{\theta}(x, y)$ is not defined for the location of the destination.

The above assumption implies that for every location of the network there exists a single path between that location and the destination. Mathematically, we define a path as a directed curved line starting at (x, y) and ending at the destination. Let $p(x, y)$ denotes the the set of points in A that belong to this path. Note that based on assumption **A2** the paths have the *suffix property*, which means if the path of a point (x, y) passes through a point (x', y'), then the path of (x', y') coincides with the reminder (suffix) of the path from (x, y).

It should be noted that the chosen paths are not constrained by the location of intermediate sensors. Instead, the paths are *abstract* paths in the plane that represent desired paths for the transmission of messages. For communication to occur, we need to define the routes in terms of the paths. Another important note is connection between the paths and routes. We define a *route* as a sequence of sensor nodes starting at a sensor and ending at the destination In order to find a route from a sensor to the destination, we approximate the path starting at the location of the sensor by a piecewise linear path with sensors in its vertices. This approximation is justified if we assume sensors are distributed dense enough in the area of the network. However, if sensors are not dense enough in certain areas, such areas will be considered as places with limited resources and we can adapt the framework by penalizing the traversal of such areas. (The weighting function $K(x, y)$ introduced later would take large values for these areas.)

Given a set of abstract paths for each location (x, y), for a connected set L in A, we define the notion of *Upstream Area*, $\alpha(L)$ as:

$$\alpha(L) = \{(x, y) \in A \;\text{ s.t. }\; \exists \;\; (x', y') \in L \text{ and } (x', y') \in p(x, y)\} \tag{2.2}$$

The above notion is simply the network area whose generated traffic passes through L in its way to the destination. This concept has been clarified in figure 2.1. The shaded area in this figure shows the upstream area of L. As can be seen in this figure, the path of every point (x, y) in the shaded area passes through L.

Next, we define a vector field on A which we refer to as the *load density* vector field and denote by $\vec{D}$. This vector field represents the flux density of the paths to the destination. Given a point $(x_1, y_1) \in A$, we choose a small line segment L containing (x_1, y_1), and perpendicular to $\hat{\theta}(x_1, y_1)$. The magnitude of the load density vector field is the density of messages passing through L, which can be written as the ratio of all load generated in the upstream area of L divided by the length of L:

$$\vec{D}(x_1, y_1) = \lim_{|L| \to 0} \frac{\hat{\theta}(x_1, y_1)}{|L|} \int_{\alpha(L)} r(x, y)\, dxdy \tag{2.3}$$

It is important to note that $|\vec{D}(x_1, y_1)|$ is the sum of the communication load of all the paths that pass through L. So $|\vec{D}(x_1, y_1)|$ represents the actual amount of communication activity at point (x_1, y_1). The direction of $\vec{D}(x_1, y_1)$ is $\hat{\theta}(x_1, y_1)$, which is the single valued direction on which the traffic at point (x_1, y_1) is forwarded according to assumption **A2**.

Finally, we define a scalar function $\rho(x, y)$ on the network. This function represents the spatial density of rate at which the messages are generated in the network. This quantity is a function of location, and obviously $\rho(x, y) = r(x, y)$ for $(x, y) \neq (x_0, y_0)$, in which (x_0, y_0) is the location of the destination. However, since all messages end at the destination, the density of the rate has a Dirac delta form at the location of the destination. Hence:

$$\rho(x, y) = r(x, y) - w_0\delta(x - x_0)\delta(y - y_0) \tag{2.4}$$

in which w_0, the weight of delta function, is the integral of $r(x, y)$ in the network area:

$$w_0 = \int_A r(x, y)dxdy \tag{2.5}$$

The above definition of $\rho(x, y)$ implies that

$$\int_A \rho(x, y)dxdy = 0 \tag{2.6}$$

The definition of $\vec{D}(x, y)$ given by equation (3.5) satisfies:

$$\oint_C \vec{D} \cdot \vec{dn} = \int_{S(C)} \rho(x, y)dxdy \tag{2.7}$$

in which the integral is over a closed contour C, $\vec{dn}$ is a differential vector normal to that contour at each point of its boundary and pointing to outside of the counter, dot represents the inner product of vectors in two-dimensional space, and $S(C)$ is the area surrounded by the closed contour C. Equation (2.7) is analogous to Gauss' law in the electrostatic theory, and it has a very simple explanation: the rate at which the messages exit a contour is the net amount of the sources inside that contour.

With the above definition of $\rho(x, y)$ and $\vec{D}(x, y)$ equation (2.7) can be expressed in partial differential equation form:

$$\vec{\nabla} \cdot \vec{D}(x, y) = \rho(x, y) \tag{2.8}$$

where $\vec{\nabla}$ is defined as:

$$\vec{\nabla} = \frac{\partial}{\partial x}\hat{\imath} + \frac{\partial}{\partial y}\hat{\jmath} \tag{2.9}$$

in which x and y represent the variables in the Cartesian coordinate frame, and $\hat{\imath}$ and $\hat{\jmath}$ represent the unit vectors along x and y axes respectively.

Depending on how we select the set of paths, the value of $\vec{D}(x, y)$ is different, but independent of path selection method, $\vec{D}(x, y)$ satisfies the following equations:

$$\begin{cases} \vec{\nabla} \cdot \vec{D}(x, y) = \rho(x, y) \\ D_n(x, y) = 0 \quad for\ (x, y) \in Boundary\ of\ A \end{cases} \tag{2.10}$$

in which A denotes the geographical set that contains the network and $D_n(x, y)$

denotes the normal component of $\vec{D}(x, y)$ on the boundary of A. The first equation in (2.10) is the natural limitation imposed by the fact that all the traffic generated at the network should be delivered to the destination. The second equation comes from the fact that no load is desired to exit the geographical area of the network or enter into it through the boundary. It is important to notice that equations (2.10) do not give $\vec{D}(x, y)$ uniquely.

Conversely, if we have a $\vec{D}(x, y)$ that satisfies equations (2.10), we can find the path that can be used to send the traffic of each point (x, y) to the destination. In order to define the routes based on the values of $\vec{D}(x, y)$, we need to define the concept of *load flow lines*. These lines are similar to the electric flux lines in electrostatic theory [1] [2][4]. The load flow lines are a family of curved lines that are always tangent to the direction of $\vec{D}(x, y)$ and their orientation is the same as the orientation of $\vec{D}(x, y)$. The load flow lines cannot intersect except at the destination; if they intersect, at the point of intersection the direction of the field would not be single-valued. The other property of the load flow lines is that these lines always end at the destination; this fact is because the value of divergence in equations (2.10) is nonnegative at every place of the network, except it is negative at the destination.

Based on the definition of the load flow lines, the path corresponding to each point (x, y) can be easily found as the flux lines of the load density vector field, and finally $\hat{\theta}(x, y)$, can be written as:

$$\hat{\theta}(x, y) = \vec{D}(x, y)/|\vec{D}(x, y)|. \tag{2.11}$$

The singularities introduced by the above equation do not make any problem since if $\vec{D}(x, y) = 0$, then place (x, y) has not been used for routing and no communication is made through it. It is straightforward to see that if we plug the paths generated by load flux lines in equation (3.5), we get the original value of $\vec{D}(x, y)$.

2.2 Optimizing Routes

In the previous section we established the basic concept of load vector density field, and described its connection to routing. Thus, given $\vec{D}(x, y)$, we obtain the paths and based on the paths we find the routes to the destination. However, equations (2.10) do not specify $\vec{D}(x, y)$ uniquely. The remaining issue is to decide what additional condition(s) to place on $\vec{D}(x, y)$ so the resulting vector field generates a desirable set of routes. The intuition we follow is that by making $\vec{D}$ as uniform as possible, we obtain routes that cause the traffic to be highly dispersed throughout the network. In turn, this decreases both node congestion and collisions and lead to high throughput.

Spreading the communication load in the network can be formulated as minimizing the following quadratic cost function:

$$J(\vec{D}) = \int_A |\vec{D}(x, y)|^2 dxdy \tag{2.12}$$

The quadratic form of the cost function in equation (2.12) causes the load to be distributed as uniformly as possible. It prevents having high loads somewhere in the network while the resources are underutilized somewhere else. One interesting fact about this cost function is that it is similar to the definition of energy in electrostatic theory. The above optimization problem can be summarized as:

$$\begin{aligned}
&\text{Minimize } J(\vec{D}) = \textstyle\int_A |\vec{D}(x,y)|^2 dxdy \\
&\text{Subject to:} \\
&\vec{\nabla} \cdot \vec{D}(x,y) = \rho(x,y) \\
&D_n(x,y) = 0,\ \forall (x,y) \in \text{Boundary of } A
\end{aligned}$$

The following theorem provides the key to finding the solution of the optimization problem defined by (2.13).

Theorem 1: If $\vec{D}^*(x,y)$ denotes the optimal solution of equation (2.13), then it satisfies:

$$\vec{\nabla} \times \vec{D}^*(x,y) = 0 \tag{2.13}$$

In the above equation $\vec{\nabla}\times$ is the two dimensional curl operator, and it is defined in the following way for a vector field $\vec{F} = [F_x\ F_y]$:

$$\vec{\nabla} \times \vec{F}(x,y) = \left(-\frac{\partial F_x(x,y)}{\partial y} + \frac{\partial F_y(x,y)}{\partial x}\right)\hat{k} \tag{2.14}$$

in which $\hat{k}$ is a unit vector perpendicular to $\hat{\imath}$ and $\hat{\jmath}$. More precisely, $\hat{k} = \hat{\imath} \times \hat{\jmath}$.

Proof: In order to prove the theorem, it suffices to prove that for every closed contour C we have:

$$\oint_C \vec{D}(x,y) \cdot \vec{dl} = 0 \tag{2.15}$$

in which $\vec{dl}$ is a differential vector tangent to C. First we prove the above fact for a contour C that is a rectangle similar to that in Figure 2.2. In this figure we define two other equally spaced rectangles inside and outside C, and call those C_{in} and C_{out} respectively. The distance between the edges of C_{in} and C_{out} is assumed to be equal for the four pair of corresponding edges, and we denote it by β. Assume T is the area surrounded between C_{in} and C_{out} . We divide T into four parts: T_1, T_2, T_3, and T_4 as illustrated in Figure 2.2.

Now we define the following vector field:

$$\vec{\delta}(x,y) = \begin{cases} \epsilon\hat{i} & if \ \ (x,y) \in T_1 \\ \epsilon\hat{j} & if \ \ (x,y) \in T_2 \\ -\epsilon\hat{i} & if \ \ (x,y) \in T_3 \\ -\epsilon\hat{j} & if \ \ (x,y) \in T_4 \\ 0 & otherwise. \end{cases} \tag{2.16}$$

in which ϵ is a small positive constant. Equation (2.16) defines a vector field that makes a lossless counterclockwise rotation in T. It can easily be verified that:

$$\vec{\nabla} \cdot \vec{\delta}(x,y) = 0 \tag{2.17}$$

Now we observe the fact that if we define $\vec{D_1}(x,y) = \vec{D^*}(x,y) + \vec{\delta}(x,y)$, then $\vec{\nabla} \cdot \vec{D_1}(x,y) = \rho(x,y)$, and hence, $\vec{D_1}(x,y)$ is in the feasible set of the optimization problem defined by equation (2.13). The variation of the cost function defined by equation (2.13) after adding $\vec{\delta}(x,y)$ to $\vec{D^*}(x,y)$ can be written as:

$$\begin{aligned}&\Delta J = J(\vec{D_1}(x,y)) - J(\vec{D^*}(x,y)) = \int_A |\vec{D^*}(x,y) + \vec{\delta}(x,y)|^2 dxdy \\ &- \int_A |\vec{D^*}(x,y)|^2 dxdy\end{aligned} \tag{2.18}$$

Since $\vec{\delta}(x,y) = 0$ for $(x,y) \notin T$, we have

$$\begin{aligned}\Delta J &= \int_T (|\vec{D^*}(x,y) + \vec{\delta}(x,y)|^2 - |\vec{D^*}(x,y)|^2) dxdy \\ &= 2\int_T \vec{D^*}(x,y) \cdot \vec{\delta}(x,y) dxdy + \int_T |\vec{\delta}(x,y)|^2 dxdy\end{aligned}$$

If we assume ϵ is small enough, we can ignore the term that has ϵ^2. Then:

$$\Delta J = 2\int_T \vec{D^*}(x,y) \cdot \vec{\delta}(x,y) dxdy \tag{2.19}$$

On the other hand, for small enough β and ϵ we have:

$$\int_T \vec{D^*}(x,y) \cdot \vec{\delta}(x,y) dxdy = \epsilon\beta \oint_C \vec{D^*}(x,y) \cdot \vec{dl} \tag{2.20}$$

From the theory of calculus of variations the value of ΔJ should be zero since $\vec{D^*}(x,y)$ minimizes the cost function defined by equation (2.13). Hence:

$$\oint_C \vec{D^*}(x,y) \cdot \vec{dl} = 0 \tag{2.21}$$

So far we have proved the validity of equation (2.13) for a rectangular contour. To complete the proof it suffices to observe the fact that the area surrounded by an arbitrary contour C_1 can be divided into many rectangular elements, and the integral over the boundary of C_1 defined by equation (2.13) is the sum

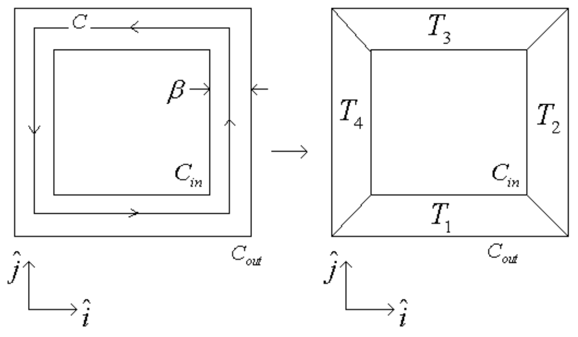

Figure 2.2: Illustration of the notations for the proof of Theorem 1.

of integrals over the boundaries of the small rectangles.**QED**.

Based on the result of Theorem 1, we can write a set of partial differential equations for the optimal $D^*(\vec{x}, y)$:

$$\begin{aligned} \vec{\nabla} \cdot \vec{D}^*(x,y) &= \rho(x,y) \\ \vec{\nabla} \times \vec{D}^*(x,y) &= 0 \end{aligned} \tag{2.22}$$

Mathematically, a vector field for which $\vec{\nabla} \times \vec{D}(x,y) = 0$ is called a conservative vector field. It is proved that such a vector field can be expressed as the gradient of a scalar field. In other words:

$$\vec{D}(x,y) = \vec{\nabla} V(x,y) \tag{2.23}$$

in which $V(x,y)$ is a scalar function known as the potential function. Then the set of equations defined by (2.22) reduces to:

$$\nabla^2 V(x,y) = \rho(x,y) \tag{2.24}$$

in which the operator ∇^2 is defined as:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{2.25}$$

The boundary conditions for $\vec{D}(x, y)$ imply that the gradient of $V(x, y)$ is zero on the boundary along the direction that is normal to the boundary. In other words:

$$\vec{\nabla} V(x, y) \cdot \hat{n}(x, y) = 0, \, \forall (x, y) \in \text{Boundary of } A \tag{2.26}$$

in which $\hat{n}(x, y)$ is a unit vector normal to the boundary.

The partial differential equation defined by (2.25) is known as the Poisson equation. The potential function found as the solution of this equation has many interesting interpretations. Firstly, the potential function gives a rough idea of how much effort by the network is needed to send data from a source to the destination. This effort is proportional to the potential difference of the source and the destination. Secondly, the potential function gives insight into the routing. Based on equation (2.23), the routing is done in the direction of the gradient of the potential function. Some concerns like the possibility of forming routing loops are naturally avoided since the potential function changes strictly monotonic in the nodes that form a path from the source to the destination.

2.3 Energy Efficiency

In this section, we try to find the paths that give energy efficiency. In order to do so, we need to define the notion of residual energy density in the network. For this purpose, we divide the network through evenly spaced vertical and horizontal grids, such that the network area is partitioned into small rectangles. The number of grids is chosen such that the sensors inside neighboring rectangles are within the single hop transmission range of each other (roughly, the edge of each rectangle is about half of the transmission range of a sensor). Then for all points (x, y) belonging to the i^{th} rectangle S_i, we define the following residual energy density:

$$\Omega(x,y,t) = \frac{1}{|S_i|} \sum_{\text{sensor } j \ \in S_i} e_j(t) \tag{2.27}$$

in which $e_j(t)$ denotes the residual energy of sensor j at time t, and $|S_i|$ is the area of S_i . Note that $\Omega(x, y, t)$ is a piecewise constant function on (x, y), and it takes a constant value for all points (x, y) in the same rectangle.

In order to find the routes that give energy efficiency, we use the fact that the amount of communication activity at place (x, y) of the network is proportional to $|\vec{D}(x, y)|$. In other words, at place (x, y), the average amount of $|\vec{D}(x, y)|$ of date is transmitted per unit of time. This means that the depletion rate of energy at (x, y) is proportional to $|\vec{D}(x, y)|$. No transmissions are possible through the places with zero energy. In order to get energy efficiency, the intuition we follow is trying to match the value of $|\vec{D}(x, y)|$ to the density of residual energy at place (x, y) of the network. In other words, we try to make more routing through the places with high residual energy, while penalizing use of low residual energy places for routing. This goal can be achieved by minimizing the following cost function:

$$J(\vec{D}) = \int_A K(x,y)|\vec{D}(x,y)|^2 dxdy \tag{2.28}$$

in which $K(x,y)$ is a *positive* scalar weight function on A, and it takes a high value in the places of the network that have low residual energy to penalize routing through them, and it takes a small value in the places with enough energy. $K(x,y)$ is a non-increasing function of $\Omega(x,y,t)$, and we will explain their relationship later in this section. It should be noted that the problem of minimizing $J(\vec{D}(x,y))$ is subject to the constraints given by equation (2.13), and indeed the optimization problem of (2.13) is the special case of optimization problem of (2.28) where the weight function $K(x,y)$ is uniform in the network (i.e., $K(x,y) = c$). The above optimization problem can be summarized in the following way:

$$\begin{aligned}
&\text{Minimize } J(\vec{D}) = \textstyle\int_A K(x,y)\left|\vec{D}(x,y)\right|^2 \mathrm{d}s \\
&\text{Subject to:} \\
&\vec{\nabla}\cdot\vec{D}(x,y) = \rho \\
&D_n(x,y) = 0 \;\; (x,y) \in \text{Boundary of } A
\end{aligned}$$

The following theorem provides the key to finding the solution of the optimization problem defined by (2.29).

Theorem 2: If $\vec{D}^*(x, y)$ denotes the optimal solution of equation (2.29), then it satisfies:

$$\vec{\nabla} \times \vec{E}^*(x, y) = 0 \tag{2.29}$$

in which

$$\vec{E}^*(x, y) = K(x, y)\vec{D}^*(x, y) \tag{2.30}$$

Proof: In order to prove the theorem, it suffices to prove that for every closed contour C we have:

$$\oint_C \vec{E}^* \cdot \vec{dl} = 0 \tag{2.31}$$

in which $\vec{dl}$ is a differential vector element tangent to the contour. The rest of proof of this theorem is similar to that of Theorem 1.

Based on the result of Theorem 2, we can write a set of partial differential equations for the optimal $\vec{D}^*(x, y)$ and $\vec{E}^*(x, y)$:

$$\begin{aligned} \vec{\nabla} \cdot \vec{D}^*(x, y) &= \rho(x, y) \\ \vec{\nabla} \times \vec{E}^*(x, y) &= 0 \end{aligned} \tag{2.32}$$

The set of equations given by (2.32) is analogous to Maxwell's equations in the electrostatic theory. In this analogy, $\vec{E}^*(x, y)$ is analogous to the electric field density, $\vec{D}^*(x, y)$, is analogous to the electric displacement, and $K(x, y)$ is analogous to the inverse of permittivity factor in a non-homogeneous media. In the theory of partial differential equations it is proved that the above equations along with the boundary condition given by (2.10) give $\vec{D}^*(x, y)$ and $\vec{E}^*(x, y)$ uniquely.

Since $\vec{\nabla} \times \vec{E^*}(x,y) = 0$ then $\vec{E^*}(x,y)$ is a conservative vector field, and it can be expressed as the gradient of a scalar field:

$$\vec{E^*} = \vec{\nabla} U \tag{2.33}$$

Then the set of equations defined by (2.32) reduces to:

$$\vec{\nabla}^2 U(x,y) - \frac{\vec{\nabla} K(x,y).\vec{\nabla} U(x,y)}{K(x,y)} = K(x,y)\rho(x,y) \tag{2.34}$$

Similar to before, the boundary conditions for $\vec{D}(x,y)$ imply that the gradient of $U(x,y)$ is zero on the boundary along the direction that is normal to the boundary:

$$\vec{\nabla} U(x,y) \cdot \hat{n}(x,y) = 0, \, \forall (x,y) \in \text{Boundary of } A \tag{2.35}$$

in which $\hat{n}(x,y)$ is a unit vector normal to the boundary. For the case in which $K(x,y)$ is constant, Equation (2.34) reduces to the well known Poisson equation:

$$\nabla^2 U(x,y) = K(x,y)\rho(x,y) \tag{2.36}$$

There is one final issue about the way we should choose $K(x,y)$. This scalar field should be assigned in a way that it has a high value at the places with low residual energy. There might be different ways of doing so, but one easy assignment of $K(x,y)$ can be done in the following way:

$$K(x,y,t) = \frac{1}{\Omega(x,y,t)}. \tag{2.37}$$

As equation (2.37) shows, $K(x, y)$ depends on t. In practice, we do not need to change the value of $K(x, y)$ very frequently, which would cause frequent change of the paths and routes. The update of $K(x, y)$ can be done with a low frequency, only after a considerable change in the residual energy of the network happens.

Chapter 3

Generalization To Multiple Destinations

3.1 Multiple Destinations with Fixed Locations of Destinations

So far we have introduced the vector field based routing method for the case where there is only one destination access point in the network and all the traffic generated by the network nodes is sent to a single destination. While this is the case in many applications both in wireless ad hoc and sensor networks, a more general case is where we have more than one destination in the network.

The summary of the case with one destination can be written as:

$$\begin{aligned}
&\vec{\nabla}\cdot D^{*}(\vec{x},y)=\rho(x,y)\\
&\vec{\nabla}\times\vec{E^{*}}(x,y)=0\\
&D_n(x,y)=0,\ \forall(x,y)\in\text{Boundary of }A
\end{aligned}$$

Recall that

$$\rho(x,y)=r(x,y)-w_0\delta(x-x_0)\delta(y-y_0)$$

where

$$w_0=\int_A r(x,y)dxdy$$

The fact that we have only one destination is applied in the above mathematical formulation by defining a Dirac delta function for the value of $\rho(x,y)$ at (x_0,y_0), where the destination is located. As can be seen in the above formulation, the weight of this delta function is equal to the total amount of traffic generated in the network with the opposite sign. The above formulation forces all the routing paths of the network to end at the destination.

One complexity of the case where we have several destinations is how we distribute the load of the network among them. In the case of multiple destinations, we write the optimization problem as:

$$\begin{aligned}
&\text{Minimize } J(\vec{D})=\textstyle\int_A K(x,y)|\vec{D}(x,y)|^2dxdy\\
&\text{Subject to:}\\
&\vec{\nabla}\cdot\vec{D}(x,y)=\rho(x,y)\\
&\rho(x,y)=r(x,y)-\textstyle\sum_{i=1}^{M}w_i\delta(x-x_i)\delta(y-y_i)\\
&D_n(x,y)=0\ ,\forall(x,y)\in\text{Boundary of }A
\end{aligned}$$

in which M is the number of destinations, (x_i, y_i) is the location of i^{th} destination and w_i is a nonnegative weight of the i^{th} destination. Since in this case the load of the network is received by the M destinations, we have:

$$\sum_{i=1}^{M} w_i = \int_A r(x,y)dxdy = w_0$$

This implies:

$$\int_A \rho(x,y)dxdy = 0$$

It is important to note that in the multiple destination case the optimization is both on $\vec{D}(x,y)$ and the values of w_i. In this case, the paths starting from each point of the network end at one of the destinations. Based on the paths, we partition the area of the network into M disjoint sets corresponding the M destinations. Let T_i denote the set for the i^{th} destination. Then a point (x,y) belongs to T_i if the path corresponding to (x,y) ends at the destination i.

We call T_i the *region of attraction* of destination i. It is straightforward to verify that the region of attraction for each destination is a connected set. This is because $(x_i, y_i) \in T_i$, and (x_i, y_i) belongs to every path that ends at the destination i. Then if two point such as (x,y) and (x', y') both belong to T_i, and if p and p' are their corresponding paths to the i^{th} destination respectively, then based on the suffix property of paths the union of p and p' is a connected subset of T_i containing both (x,y) and (x', y').

Based on the above definition of T_i, we can write the weight of i^{th}

destination as:

$$w_i = \int_{T_i} r(x,y)dxdy \tag{3.1}$$

The complexity of problem in the multiple destination case is based on the fact that optimization is both on $\vec{D}(x,y)$, and the weight values $w_1, w_2, \cdots, w_M$. If the weight values are fixed, the following lemma can be stated:

Lemma 1: In the case of multiple destinations, if $w_1, w_2, \cdots, w_M$, are fixed, the necessary and sufficient condition for optimality of the cost function in (3.1) is:

$$\vec{\nabla} \times \vec{E}^*(x,y) = 0 \tag{3.2}$$

in which

$$\vec{E}^*(x,y) = K(x,y)\vec{D}^*(x,y) \tag{3.3}$$

The proof of this Lemma is similar similar to the proof of Theorem 1 and Theorem 2. If the weights are given, the value of $\rho(x,y)$ is known, and the optimal solution of the problem is found by solving the following PDE:

$$\begin{aligned} &\vec{\nabla}^2 U(x,y) - \tfrac{\vec{\nabla} K(x,y).\vec{\nabla} U(x,y)}{K(x,y)} = K(x,y)\rho(x,y) \\ &\vec{\nabla} U(x,y) \cdot \hat{n}(x,y) = 0,\ \forall (x,y) \in \text{Boundary of } A \end{aligned} \tag{3.4}$$

and ultimately $\vec{E}(x,y)$ is from $\vec{E}(x,y) = \nabla U(x,y)$, and $\vec{D}(x,y)$ if found from $\vec{E}(x,y)$.

If the weight values of the destinations are added to the optimization variables of (3.1), the condition $\vec{\nabla} \times \vec{E}^*(x,y) = 0$ is not a sufficient condition for optimality of the cost function. To illustrate this case, we consider study a simple example.

Example 1: Consider a network in the area of a rectangle with length 1 and width 0.01 in which the total load of 1 unit is uniformly distributed; this results in $r(x, y) = 100$ for every (x, y) in the network. Furthermore, assume $K(x, y) = 1$ for all points in the network. We consider two destinations in the network, destination 1 located at point $(0.15, 0.005)$, and destination 2 located at $(0.5, 0.005)$. This network and the cartesian coordinate frame are shown in the top plot of Figure 3.1.

To continue, we consider two different distributions of load between the two destinations:

Case 1: In this case, we assume $w_1 = w_2 = 0.5$. With this assignment of the load, the network is partitioned into two sets which represent the regions of attraction of the two destinations. T_1, the region of attraction of destination 1, is every point in the network with $x < 0.5$, and T_2 is every point in the network with $x > 0.5$. Because the network is so narrow in the y direction, all paths from every point in the network to the destinations are approximately parallel to the x axis, and therefore $\vec{D}(x, y)$ is approximately horizontal (parallel to the x axis) at every point of the network. The only exception is where we are very close to one of the destinations where the flux lines of $\vec{D}(x, y)$ need to change their direction to end at the destinations. By ignoring this small deviation of $\vec{D}(x, y)$ from horizontal lines, we ignore the component of $\vec{D}(x, y)$ in the y direction.

With the above approximation of paths, we can easily find the value of $\vec{D}(x, y)$ based on the definitions of upstream area and $\vec{D}(x, y)$. For example, at point $x = (0.12, y_1)$, we define a line segment L_1 as all points with $x = 0.12$.

Obviously, L_1 is a vertical line segment with length 0.01. Since all the paths are approximately parallel to the x axis, $\alpha(L_1)$ (the upstream area of L_1) is all the points in the network with $x < 0.12$. Therefore from equation (3.5) we have:

$$\begin{aligned}
&\vec{D}(0.12, y_1) = \lim_{|L|\to 0} \tfrac{\hat{\theta}(0.12,y_1)}{|L|} \int_{\alpha(L)} r(x,y)\, dxdy \\
&\approx \tfrac{\hat{\theta}(0.12,y_1)}{|L_1|} \int_{\alpha(L_1)} r(x,y)\, dxdy \\
&= \tfrac{\hat{\imath}}{0.01} \int_{x=0}^{0.12} \int_{y=0}^{0.01} 100 dxdy \\
&= 12\hat{\imath}
\end{aligned} \tag{3.5}$$

Note that in writing the above equation we have used $|L_1| = 0.01$ and $\hat{\theta}(0.1, y_1) = \hat{\imath}$, in which $\hat{\imath}$ is the unit vector along the x axis. Similarly we can find value of $\vec{D}(x,y)$ for all points:

$$\vec{D}(x,y) = \begin{cases} 100x\hat{\imath} & if \;\; x < 0.15 \\ (100x - 50)\hat{\imath} & if \;\; 0.15 < x < 0.5 \\ (100x - 100)\hat{\imath} & if \;\; 0.5 < x < 1 \end{cases} \tag{3.6}$$

As can be seen in the above equation, $\vec{D}(x,y)$ does not depend on y, and it is a function of x. The value of $\vec{D}(x,y)$ versus x and corresponding $U(x,y)$ are shown in Figure 3.1. The value of cost function $\int_A |\vec{D}(x,y)|^2 dxdy$ in this case is 588.5.

Case 2: In this case, we assume $w_1 = 0.31$ and $w_2 = 0.69$. This distribution gives the following for $\vec{D}(x,y)$:

$$\vec{D}(x,y) = \begin{cases} 100x\hat{\imath} & if \;\; x < 0.15 \\ (100x - 31)\hat{\imath} & if \;\; 0.15 < x < 0.5 \\ (100x - 100)\hat{\imath} & if \;\; 0.5 < x < 1 \end{cases} \tag{3.7}$$

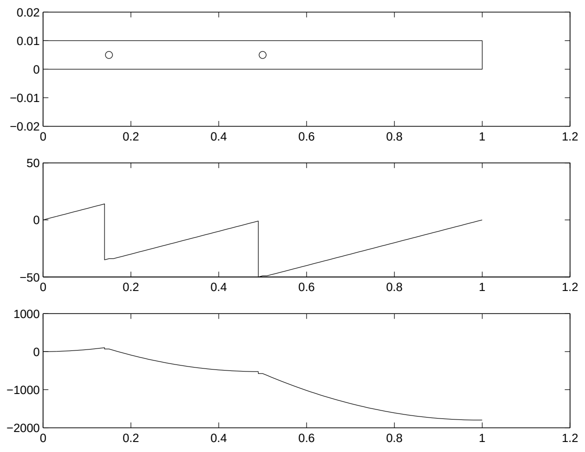

Figure 3.1: Top: The network with circles showing the location of destinations. Middle: the value of $\vec{D}(x, y)$ versus x for $w_1 = w_2 = 0.5$, Bottom: the corresponding potential function $U(x, y)$ as a function of x.

The value of $\vec{D}(x, y)$ versus x and corresponding $U(x, y)$ are shown in Figure 3.2. The value of cost function $\int_A |\vec{D}(x, y)|^2 dxdy$ in this case is 475.4.

Comparison of the two cases in the above example shows that the distribution of the load plays an important role in the optimization problem. While in both cases we have $\vec{\nabla} \times \vec{E}^*(x, y) = 0$, in the second case we have a smaller value for the cost function.

The following theorem gives the basic idea to solve the optimization problem of (3.1), where we have multiple destinations:

Theorem 3 If the value of potential function at the locations of M destinations is U_1, U_2, ..., U_M, then the necessary and sufficient conditions for

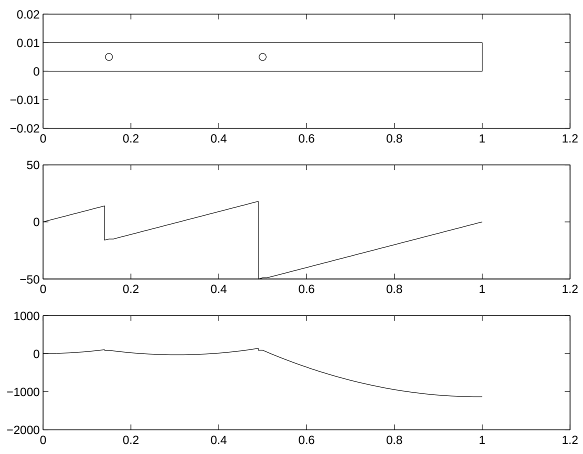

Figure 3.2: Top: the network with circles showing the location of destinations. Middle: the value of $\vec{D}(x, y)$ versus x for $w_1 = 0.31$ and $w_2 = 0.69$, Bottom: the corresponding potential function $U(x, y)$ as a function of x.

the optimality of cost function in (3.1) are:

$$\vec{\nabla} \times \vec{E}(x, y) = 0$$
$$U_i = U_j, \text{ for } \forall \; 1 \leq i, j \leq M$$

Proof: The first condition in the theorem is $\vec{\nabla} \times \vec{E}(x, y) = 0$. The proof for this condition is similar to case of having a single destination. We assume this condition holds and we show that the second condition is necessary and sufficient for optimality.

In the forward proof, we show that for optimality of (3.1) the value of the potential function should be the same at all destinations. For this purpose, we use contradiction. Assume for some i and j, we have $U_i < U_j$. Then we prove that the load distribution can be changed in the way of decreasing the cost function. We need the following two identities throughout the proof:

Identity 1: If c is a scalar field and $\vec{F}$ is a vector field, then:

$$\vec{\nabla} \cdot (c\vec{F}) = c\vec{\nabla} \cdot \vec{F} + \vec{F} \cdot \vec{\nabla} c \tag{3.8}$$

Identity 2: If A is a region in the plane with boundary ∂A, and F is a vector field defined on A, then

$$\int_A \vec{\nabla} \cdot \vec{F} dxdy = \oint_{\partial A} \vec{F} \cdot \vec{dn} \tag{3.9}$$

in which $\vec{dn}$ is the differential vector normal to the boundary of A pointing outward. The second identity is also known as the Divergence Theorem in the vector calculus literature.

To continue the proof, we make a small positive change in the weights of the i^{th} and the j^{th} destinations in the following way:

$$\begin{aligned} w'_i &= w_i + \epsilon \\ w'_j &= w_j - \epsilon \end{aligned} \tag{3.10}$$

In other words, we increase the weight of the i^{th} destination by ϵ and decrease the weight of the j^{th} destination by the same amount. Assume $\vec{D}(x, y)$, $\vec{E}(x, y)$, $U(x, y)$ and $\rho(x, y)$ represent the values of vector fields, the potential function and and the density of sources before applying the above change and $\vec{D}'(x, y)$, $\vec{E}'(x, y)$, $U'(x, y)$, and $\rho'(x, y)$ represent the same quantities after applying the change. After this change, the density of sources is:

$$\rho'(x,y) = \rho(x,y) - \epsilon\delta(x-x_i)\delta(y-y_i) + \epsilon\delta(x-x_j)\delta(y-y_j) \tag{3.11}$$

in which (x_i, y_i) is the location of the i^{th} destination and (x_j, y_j) is the location of the j^{th} destination. Now we make the following definitions:

$$\begin{aligned} \vec{D}'(x,y) &= \vec{D}(x,y) + \vec{\delta}(x,y) \\ \vec{E}'(x,y) &= \vec{E}(x,y) + \vec{e}(x,y) \\ U'(x,y) &= U(x,y) + u(x,y) \end{aligned} \tag{3.12}$$

It is easy to verify that:

$$\begin{aligned} &\vec{\nabla} \cdot \vec{\delta}(x,y) = -\epsilon\delta(x-x_i)\delta(y-y_i) + \epsilon\delta(x-x_j)\delta(y-y_j) \\ &\vec{\nabla} \times \vec{e}(x,y) = 0 \\ &\vec{e}(x,y) = \vec{\nabla}u(x,y) \end{aligned} \tag{3.13}$$

The change in the value of cost function after applying the change is:

$$\begin{aligned} &\delta J = J(\vec{D}'(x,y)) - J(\vec{D}(x,y)) = \\ &\textstyle\int_A K(x,y)|\vec{D}'(x,y)|^2 dxdy - \int_A K(x,y)|\vec{D}(x,y)|^2 dxdy \end{aligned} \tag{3.14}$$

By substituting the value of $\vec{D}'(x,y)$ we have

$$\delta J = 2\int_A K(x,y)\vec{\delta}(x,y) \cdot \vec{D}(x,y) dxdy + \int_A K(x,y)|\vec{\delta}(x,y)|^2 dxdy \tag{3.15}$$

Since we assume ϵ is a very small value, we can ignore the second term in the above equations, and we have:

$$\begin{aligned}&\delta J = 2\int_A K(x,y)\vec{\delta}(x,y)\cdot\vec{D}(x,y)dxdy = \\ &2\int_A \delta(x,y)\cdot\vec{E}(x,y)dxdy = \\ &2\int_A e(x,y)\cdot\vec{D}(x,y)dxdy\end{aligned} \tag{3.16}$$

Note that in writing the second and third equalities in the above we have used the fact that $K(x,y)\vec{\delta}(x,y) = \vec{e}(x,y)$ and $K(x,y)\vec{D}(x,y) = \vec{E}(x,y)$. Now we use Identity 1 for $c = U(x,y)$ and $F = \delta(x,y)$:

$$\vec{\nabla}\cdot(U(x,y)\vec{\delta}(x,y)) = U(x,y)\vec{\nabla}\cdot\vec{\delta}(x,y) + \vec{\delta}(x,y)\cdot\vec{\nabla}U(x,y) \tag{3.17}$$

By using the fact that $\vec{E}(x,y) = \vec{\nabla}U(x,y)$, the above equation can be written as:

$$\vec{\delta}(x,y)\cdot\vec{E}(x,y) = \vec{\nabla}\cdot(U(x,y)\vec{\delta}(x,y)) - U(x,y)\vec{\nabla}\cdot\vec{\delta}(x,y) \tag{3.18}$$

By substituting this value for $\vec{\delta}(x,y)\cdot\vec{E}(x,y)$ in equation (3.16) we have:

$$\delta J = 2\int_A \vec{\nabla}\cdot(U(x,y)\vec{\delta}(x,y))dxdy - 2\int_A U(x,y)\vec{\nabla}\cdot\vec{\delta}(x,y)dxdy \tag{3.19}$$

Now we use the Divergence Theorem given in Identity 2 for $\vec{F} = U(x,y)\vec{\delta}(x,y)$. We have:

$$\int_A \vec{\nabla}\cdot(U(x,y)\vec{\delta}(x,y))dxdy = \oint_{\partial A} U(x,y)\vec{\delta}(x,y)\cdot\vec{dn} \tag{3.20}$$

in which $\vec{dn}$ is a differential vector normal to ∂A and pointing outward. Recall that the boundary conditions in the optimization problem of (3.1) implies both $\vec{D}(x,y)$ and $\vec{D}'(x,y)$ have zero components in the direction normal to the boundary of A. So $\vec{\delta}(x,y) = \vec{D}'(x,y) - \vec{D}(x,y)$ also has zero normal component at every point of the boundary of A. This causes the inner product in the integrand of equation (3.20) to be 0. Therefore

$$\int_A \vec{\nabla} \cdot (U(x,y)\vec{\delta}(x,y))dxdy = 0 \tag{3.21}$$

On the other hand from equation (3.13) we have $\vec{\nabla} \cdot \vec{\delta}(x,y) = -\epsilon\delta(x-x_i)\delta(y-y_i) + \epsilon\delta(x-x_j)\delta(y-y_j)$. Therefore:

$$\begin{aligned}&\textstyle\int_A U(x,y)\vec{\nabla} \cdot \vec{\delta}(x,y)dxdy = \\ &-\epsilon \textstyle\int_A U(x,y)\delta(x-x_i)\delta(y-y_i)dxdy + \epsilon \int_A U(x,y)\delta(x-x_j)\delta(y-y_j)dxdy = \\ &-\epsilon U(x_i,y_i) + \epsilon U(x_j,y_j) = -\epsilon(U_i - U_j)\end{aligned} \tag{3.22}$$

By substituting (3.22) and (3.21) in (3.19) we get:

$$\delta J = 2\epsilon(U_i - U_j) < 0 \tag{3.23}$$

and this ends the forward part of the proof. The above equation states that if the potential at the destination i is smaller than that in the destination j, then we can reduce the cost function by decreasing the weight of the destination j by some small ϵ value and adding that amount to the weight of the destination i.

Now we continue the proof of Theorem 3 in the backward part. From the forward part of the proof we know if $\vec{D}(x,y)$ is the optimal solution of (3.1), then its corresponding potential function takes the same value in the location of all destinations. Also for this load vector field we have $\vec{\nabla} \times \vec{E}(x, y) = 0$.

Assume in addition to the optimal $\vec{D}(x, y)$, there exists a $\vec{D}'(x, y)$ that satisfies $\vec{\nabla} \times \vec{E}'(x, y) = 0$ as well as the conditions of the optimization problem given in (3.1). Also assume the corresponding potential function of $\vec{D}'(x, y)$ takes the same value in the location of all destinations. Then we prove that $\vec{D}'(x, y) = \vec{D}(x, y)$.

Let $U(x, y)$, $w_1, w_2, ..., w_M$, and $\rho(x, y)$ represent the quantities of the optimal solution, and $U'(x, y)$, $w'_1, w'_2, ..., w'_M$ and $\rho'(x, y)$ represent the similar quantities corresponding to $\vec{D}'(x, y)$. We define the difference of quantities for the two solutions in the following way:

$$\begin{aligned}
\vec{e}(x, y) &= \vec{E}(x, y) - \vec{E}'(x, y) \\
\vec{\delta}(x, y) &= \vec{D}(x, y) - \vec{D}'(x, y) \\
u(x, y) &= U(x, y) - U'(x, y) \\
\sigma(x, y) &= \rho(x, y) - \rho'(x, y)
\end{aligned} \tag{3.24}$$

It is easy to verify that

$$\begin{aligned}
\vec{\nabla} \cdot \vec{\delta}(x, y) &= \sigma(x, y) \\
\vec{\nabla} \times \vec{e}(x, y) &= 0 \\
\vec{e}(x, y) &= \vec{\nabla} u(x, y)
\end{aligned} \tag{3.25}$$

Recall that

$$\begin{aligned}
\rho(x, y) &= r(x, y) - \textstyle\sum_{i=1}^{M} w_i \delta(x - x_i)\delta(y - y_i) \\
\rho'(x, y) &= r(x, y) - \textstyle\sum_{i=1}^{M} w'_i \delta(x - x_i)\delta(y - y_i)
\end{aligned}$$

Hence:

$$\sigma(x,y) = -\sum_{i=1}^{M}(w_i - w_i')\delta(x-x_i)\delta(x-y_i) \tag{3.26}$$

which implies that $\sigma(x,y)$ is zero everywhere in the network that is not a destination.

We use contradiction to prove $\vec{D}(x,y) = \vec{D}'(x,y)$. If $\vec{D}(x,y) \neq \vec{D}'(x,y)$, then there exist some destinations for which $w_i \neq w_i'$. This is because of the fact that if for all destinations we have $w_i = w_i'$, we have $\rho(x,y) = \rho'(x,y)$, and therefore $\vec{D}(x,y) = \vec{D}'(x,y)$. If we assume there exist some destinations for which $w_i \neq w_i'$, then we select the destination j for which the corresponding value of $w_j - w_j'$ is minimum. Since $\sum_{i=1}^{M}(w_i - w_i') = 0$, then $w_j - w_j' < 0$. Hence the flux lines of $\vec{\delta}(x,y)$ exit this destination; this is because the value of divergence of $\vec{\delta}(x,y)$ is positive at the location of destination j:

$$\vec{\nabla}\cdot\vec{\delta}(x_j,y_j) = \sigma(x_j,y_j) = -(w_j - w_j')\delta(x-x_j)\delta(y-y_j) > 0 \tag{3.27}$$

The flux lines exiting the destination j can end only at the destinations for which the value of $\sigma(x,y)$ is negative. This is because the value of divergence of $\vec{\delta}(x,y)$ should be negative at a location that flux lines end. Since $\sigma(x,y)$ can take nonzero values only at the locations of the destinations, every flux line exiting destinations j ends at a destination k for which $w_k - w_k' > 0$, and hence $\sigma(x_k,y_k) < 0$. Let L be one of such flux lines. Based on the definition of flux lines, L is tangent to both of $\vec{\delta}(x,y)$ and $\vec{e}(x,y)$ at every point of it. Next we consider the following value of the potential difference of destinations

j and k:

$$u(x_j, y_j) - u(x_k, y_k) = \int_L \vec{e}(x, y) \cdot \vec{dl} \tag{3.28}$$

in which the integration is in the direction of the flux line L (e.g., from the location of destination j to the location of destination k). In equation (3.28) $\vec{dl}$ is a differential vector along L, and hence it has the same direction as $\vec{e}(x, y)$ at every point of L. Therefore, we have:

$$\vec{e}(x, y) \cdot \vec{dl} = |\vec{e}(x, y)||\vec{dl}| \tag{3.29}$$

The definition of flux lines implies that $\vec{e}(x, y)$ is nonzero at every point of L. Hence:

$$\vec{e}(x, y) \cdot \vec{dl} = |\vec{e}(x, y)||\vec{dl}| > 0 \tag{3.30}$$

By comparing (3.30) and (3.28) we have:

$$u(x_j, y_j) - u(x_k, y_k) > 0 \tag{3.31}$$

Recall the fact that $U(x, y)$ takes the same value in the locations of all destinations and the same fact is true for $U'(x, y)$. Hence $u(x, y) = U(x, y) - U'(x, y)$ takes the same value in the locations of all destinations. The statement of equation (3.31) contradicts this fact. Therefore we have $w_i = w'_i$ for all destinations and hence $\vec{D}(x, y) = \vec{D}'(x, y)$. **QED.**

The following is an intermediate result of Theorem 3:

Corollary 1: If we increase the weight of destination i by a small amount ϵ, and subtract that amount from the weight of destination j, then the first order increment in the cost function in (3.1) is:

$$\delta J = 2\epsilon(U_i - U_j) \tag{3.32}$$

We use Theorem 3 and Corollary 1 to introduce an iterative algorithm that gives the optimal assignment of load among the destinations.

Assume we start with an arbitrary assignment of the weights to the destinations. Weight assignment is subject to the constraints that the sum of the weights should be the total amount of the load in the network:

$$\sum_{i=1}^{M} w_i = \int_A r(x,y)dxdy = w_0 \tag{3.33}$$

Given the initial assignment of the weights, we solve the following equations to find the corresponding values for the resulting $U(x,y)$. Then if the value of $U(x,y)$ is the same at the location of all destinations, we have found the optimal solution, otherwise we continue the iteration by reassigning the weight values.

We use Corollary 1 to reassign the weights. We know that in order to improve the cost function we have to decrease the weight of destinations with a high value of the potential function and increase the weight of the destinations with small value of potential. To do so, we use the average value of the potential function at all destinations as a reference:

$$\bar{U} = \frac{1}{M}\sum_{i=1}^{M} U_i \tag{3.34}$$

If a destination has a smaller potential value than the above average, its weight is increased, otherwise, its weight is decreased. We use the amount of deviation from the average to specify the exact amount of change for the weight of each destination.

$$\begin{aligned} \Delta w_i &= \gamma(\bar{U} - U_i) \\ w_i' &= w_i + \Delta w_i \end{aligned} \tag{3.35}$$

in which γ is a small positive step size, and w_i' represents the weight of the i^{th} destination after applying the change. Note that the above change of the weights preserves the property that $\sum_{i=1}^{M} w_i' = w_0$. We may stop the iterations when the maximum of absolute value of $\bar{U} - U_i$ among all destinations is below a certain threshold:

$$\max_i |\bar{U} - U_i| < \xi \tag{3.36}$$

in which ξ is a small positive value. The following lemma shows that iterations of (3.35) improve the cost function:

Lemma 2: If the value of γ small enough, then updating the weight of destination by iteration $\Delta w_i = \gamma(\bar{U} - U_i)$ decreases the cost function in the optimization problem of (3.1).

Proof: To prove we make use of Corollary 1, and show that the iteration of weights given by (3.35) can be written as a sequence of steps, and at each step we only change the weight of a pair of destinations by decreasing the weight of one of them and increasing the weight of the other one.

Without loss of generality, we assume for $i = 1, 2, ..., j$ we have $U_i > \bar{U}$

and for $i = j+1, j+2, ..., M$ we have $U_i < \bar{U}$. Then we define $j \times (M-j)$ steps of changing weights, and at each step, we decrease the weight of one of the first j destinations and increase one of the $M-j$ other destinations. Assume $1 \leq k \leq j$ and $j+1 \leq l \leq M$. Then in the step kl we make the following change:

$$\epsilon_{kl} = \frac{\gamma}{\sum_{i=1}^{j}(U_i - \bar{U})}(U_k - \bar{U})(\bar{U} - U_l) \tag{3.37}$$

in which ϵ_{kl} is the amount we decrease the weight of destination k and increase the weight of destination l. It is easy to verify $\epsilon_{kl} > 0$. This is because all the terms in (3.37) are positive. Based on Corollary 1, by applying this change we get the following increment in the cost function:

$$\delta J_{kl} = -2\frac{\gamma}{\sum_{i=1}^{j}(U_i - \bar{U})}(U_k - \bar{U})(\bar{U} - U_l)(U_k - U_l) < 0 \tag{3.38}$$

To complete the proof, it suffices to verify that the total change in the weight of each destination is equal to the amount specified in (3.35). The total change in the weight of destination k can be written as:

$$\begin{aligned}
&\Delta w_k = -\textstyle\sum_{l=j+1}^{M} \epsilon_{kl} = \\
&-\tfrac{\gamma}{\sum_{i=1}^{j}(U_i-\bar{U})} \textstyle\sum_{l=j+1}^{M}(U_k - \bar{U})(\bar{U} - U_l) = \\
&-\tfrac{\gamma(U_k-\bar{U})}{\sum_{i=1}^{j}(U_i-\bar{U})} \textstyle\sum_{l=j+1}^{M}(\bar{U} - U_l) = \\
&\gamma(\bar{U} - U_k)
\end{aligned} \tag{3.39}$$

Note that for writing the last equality in (3.39) we used the fact that $\sum_{i=1}^{j}(U_i - \bar{U}) = \sum_{l=j+1}^{M}(\bar{U} - U_l)$.

Similarly the total change in the weight of the l^{th} destination is:

$$\begin{aligned}
&\Delta w_l = \textstyle\sum_{k=1}^{j} \epsilon_{kl} = \\
&\tfrac{\gamma}{\sum_{i=1}^{j}(U_i - \bar{U})} \textstyle\sum_{k=1}^{j}(U_k - \bar{U})(\bar{U} - U_l) = \\
&-\tfrac{\gamma(U_l - \bar{U})}{\sum_{i=1}^{j}(U_i - \bar{U})} \textstyle\sum_{k=1}^{j}(\bar{U}_k - \bar{U}) = \\
&\gamma(\bar{U} - U_l)
\end{aligned} \tag{3.40}$$

QED

The significance of Lemma 2 is the fact that it reduces the number of iterations needed to find the optimal solution. Corollary 1 gives the basic idea of finding the optimal solution; however, it recommends changing the weight of only one pair of destinations at every iteration. Lemma 2 gives a way to update the weight of all M destinations at every iteration to reduce the cost function.

3.2 Relocating The Destinations

Another interesting problem in the case of multiple destinations is relocating the destinations in a way that the cost function is further minimized. So far we have assumed that the destinations are fixed at their locations. In order to relocate the destinations, we need to find a direction for each destination that moving the destination along that direction has the steepest decrease in the value of cost function. This is the equivalent to moving the destinations in the direction of the gradient of the cost function with respect to the location of each the destinations. In this section, we give the mathematical basis of finding the gradient of the cost function with respect to the locations of the destinations.

The following Theorem gives the basis for a method to move destinations:

Theorem 4: Assume the location of destination i is (x_i, y_i). If a small incremental change is made to the location of this destination, and it is moved to $(x_i + \delta x, y_i + \delta y)$, the amount of change in the optimal value of the cost function given in (3.1) is:

$$\delta J = 2w_i \vec{E}(x_i, y_i) \cdot \vec{\delta z} \tag{3.41}$$

in which $\vec{\delta} z = (\delta x\hat{i} + \delta x\hat{j})$, where $\hat{i}$, and $\hat{j}$ are the unit vectors along x and y axes respectively.

Proof: The proof for this theorem is very similar to that of Theorem 3. However, in the case of Theorem 3 we made a small change in the weight of a destination, but here the weights are fixed and we change the location of the destination.

Assume $\vec{D}(x, y)$, $\vec{E}(x, y)$, $U(x, y)$ and $\rho(x, y)$ represent the values of vector fields, the potential function and the density of sources for the case in which destination i is at location (x_i, y_i), and assume $\vec{D}'(x, y)$, $\vec{E}'(x, y)$, $U'(x, y)$, and $\rho'(x, y)$ represent the same quantities after we move destination i to the new location $(x_i + \delta x, y_i + \delta y)$. After this change, the density of sources is:

$$\rho'(x,y) = \rho(x,y) + w_i\delta(x-x_i)\delta(y-y_i) - w_i\delta(x-x_j-\delta x)\delta(y-y_j-\delta y) \quad (3.42)$$

Now we make the following definitions:

$$\begin{aligned}
\vec{D}'(x,y) &= \vec{D}(x,y) + \vec{\delta}(x,y) \\
\vec{E}'(x,y) &= \vec{E}(x,y) + \vec{e}(x,y) \\
\vec{U}'(x,y) &= U(x,y) + u(x,y)
\end{aligned} \quad (3.43)$$

It is easy to verify that:

$$\begin{aligned}
&\vec{\nabla}\cdot\vec{\delta}(x,y) = w_i\delta(x-x_i)\delta(y-y_i) - w_i\delta(x-x_j-\delta x)\delta(y-y_j-\delta y) \\
&\vec{\nabla}\times\vec{e}(x,y) = 0 \\
&\vec{e}(x,y) = \vec{\nabla}u(x,y)
\end{aligned} \quad (3.44)$$

The change in the value of the cost function after moving the i^{th} destination is:

$$\delta J = J(\vec{D}'(x,y)) - J(\vec{D}(x,y)) = \int_A K(x,y)|\vec{D}'(x,y)|^2 dxdy - \int_A K(x,y)|\vec{D}(x,y)|^2 dxdy \quad (3.45)$$

By substituting the value of $\vec{D}'(x,y)$ we have

$$\delta J = 2\int_A K(x,y)\vec{\delta}(x,y)\cdot\vec{D}(x,y)dxdy + \int_A K(x,y)|\vec{\delta}(x,y)|^2 dxdy \quad (3.46)$$

Since we assume $(\delta x, \delta y)$ is a very small in both components, we can ignore the second term in the above equation. Hence:

$$\begin{aligned}&\delta J = 2\textstyle\int_A K(x,y)\vec{\delta}(x,y)\cdot\vec{D}(x,y)dxdy = \\ &2\textstyle\int_A \vec{\delta}(x,y)\cdot\vec{E}(x,y)dxdy = \\ &2\textstyle\int_A \vec{e}(x,y)\cdot\vec{D}(x,y)dxdy\end{aligned} \tag{3.47}$$

Note that in writing the second and third equalities in the above we have used the fact that $K(x,y)\vec{\delta}(x,y) = \vec{e}(x,y)$ and $K(x,y)\vec{D}(x,y) = \vec{E}(x,y)$. Now we use Identity 1 for $c = U(x,y)$ and $F = \delta(x,y)$:

$$\vec{\nabla}\cdot(U(x,y)\vec{\delta}(x,y)) = U(x,y)\vec{\nabla}\cdot\vec{\delta}(x,y) + \vec{\delta}(x,y)\cdot\vec{\nabla}U(x,y) \tag{3.48}$$

By using the fact that $\vec{E}(x,y) = \vec{\nabla}U(x,y)$, the above equation can be written as:

$$\vec{\delta}(x,y)\cdot\vec{E}(x,y) = \vec{\nabla}\cdot(U(x,y)\vec{\delta}(x,y)) - U(x,y)\vec{\nabla}\cdot\vec{\delta}(x,y) \tag{3.49}$$

By substituting this value for $\vec{\delta}(x,y)\cdot\vec{E}(x,y)$ in equation (3.47), we have:

$$\delta J = 2\int_A \vec{\nabla}\cdot(U(x,y)\vec{\delta}(x,y))dxdy - 2\int_A U(x,y)\vec{\nabla}\cdot\vec{\delta}(x,y)dxdy \tag{3.50}$$

Now we use the divergence theorem given in Identity 2 for $\vec{F} = U(x,y)\vec{\delta}(x,y)$ We have:

$$\int_A \vec{\nabla}\cdot(U(x,y)\vec{\delta}(x,y))dxdy = \oint_{\partial A} U(x,y)\vec{\delta}(x,y)\cdot\vec{dn} \tag{3.51}$$

in which $\vec{dn}$ is a differential vector normal to ∂A and pointing outward. Recall that the boundary conditions in the optimization problem of (3.1) state both

$\vec{D}(x,y)$ and $\vec{D}'(x,y)$ have zero components in the direction normal to the boundary of A. So $\vec{\delta}(x,y) = \vec{D}'(x,y) - \vec{D}(x,y)$ also has zero normal component at every point of the boundary of A. This causes the inner product in the integrand of equation (3.51) to be 0. Therefore

$$\int_A \vec{\nabla} \cdot (U(x,y)\vec{\delta}(x,y))dxdy = 0 \tag{3.52}$$

On the other hand from equation (3.42) we have $\vec{\nabla} \cdot \vec{\delta}(x,y) = w_i\delta(x-x_i)\delta(y-y_i) - w_i\delta(x-x_j-\delta x)\delta(y-y_j-\delta y)$. Therefore:

$$\begin{aligned}&\textstyle\int_A U(x,y)\vec{\nabla} \cdot \vec{\delta}(x,y)dxdy = \\ &w_i \textstyle\int_A U(x,y)\delta(x-x_i)\delta(y-y_i)dxdy - w_i \int_A U(x,y)\delta(x-x_j-\delta x)\delta(y-y_j-\delta y)dxdy \\ &= w_i(U(x_i,y_i) - U(x_i+\delta x, y_i+\delta y))\end{aligned} \tag{3.53}$$

By substituting (3.53) and (3.52) in (3.50) we get:

$$\delta J = 2w_i(U(x_i+\delta x, y_i+\delta y) - U(x_i,y_i)) \tag{3.54}$$

Now we use the fact that $\vec{E}(x,y) = \vec{\nabla}U(x,y)$, hence if δx and δy are small values, then we can write:

$$U(x_i+\delta x, y_i+\delta y) - U(x_i,y_i) = \vec{E}(x_i,y_i) \cdot (\delta x\hat{i} + \delta y\hat{j}) = \vec{E}(x_i,y_i) \cdot \vec{\delta}z \tag{3.55}$$

By substituting (3.55) in (3.54) we get:

$$\delta J = 2w_i\vec{E}(x_i,y_i) \cdot \vec{\delta}z \tag{3.56}$$

QED.

Corollary 2: The necessary condition for optimality of the cost function with respect to the location of destinations is that:

$$\vec{E}(x_i, y_i) = 0, \quad \forall i, 1 \leq i \leq M \tag{3.57}$$

We offer a relocation method based on Theorem 4 and move each destination in the opposite direction of $\vec{E}(x, y)$. From Theorem 4 we know that this improves the cost.

Generally, because of the density of sources $\rho(x, y)$ has a Dirac delta form in the location of destinations; the equation $\vec{D}(x, y) = \rho(x, y)$ implies that the flux lines converge to the destinations from all directions, and for this reason, $\vec{E}(x, y)$ shows a very high sensitivity to (x, y) when we are close to a destination. This may degrade performance of the algorithm that relocates the destinations based on the value of $\vec{E}(x, y)$, especially when we numerically solve the PDE equations that give $\vec{E}(x, y)$. To avoid this situation, we use an average of $\vec{E}(x, y)$ in the neighborhood of each destination to find the best direction to relocate it. This average can be defined as:

$$\vec{E}_i = \frac{1}{|S_i|} \int_{S_i} \vec{E}(x, y) dxdy \tag{3.58}$$

in which S_i is a small area in the plane containing destination i, and $|S_i|$ represents the area of S_i.

To relocate destinations, we use iterations to move them, and at each iteration we move each destination in the opposite direction of the average value of $\vec{E}(x, y)$ in its neighborhood. Such iterations can be defined as:

$$(x_i, y_i)^{j+1} = (x_i, y_i)^j - \theta \vec{E}_i^j \tag{3.59}$$

in which $(x_i, y_i)^j$ is the location of destination i at the iteration j, and θ is a small step size. We may stop the iterations when the maximum of absolute value of $\vec{\bar{E}}_i$ among all destinations is below a certain threshold:

$$\max_i |\vec{E}_i^j| < \epsilon \tag{3.60}$$

in which ϵ is a small positive number.

Equation (3.35) of the previous section gives iterations that change weight of destinations to decrease the cost when the locations of the destinations are fixed. In this section we introduced iteration of (3.59) that relocate destinations to decrease the cost when the weights of the destinations are fixed. Depending on the application, one or both set of the iterations may be used to achieve the best solution. For example, if in a network the location of the destinations cannot be changed, we do not use iterations of (3.59). An important note is that in the applications that we have freedom to both relocate and change the weights of the destinations, we can use iterations of (3.59), and (3.35) in any order, and the problem converges to a joint solution for both iterations. For example, we can use the iteration of (3.35) until they converge to a solution, and for that solution we use (3.59). Since (3.59) change the locations of the destinations we may need to use (3.35) again after completing (3.59), and we continue this procedure until both iterations converge to a joint equilibrium. An important fact is that regardless of the order, at each iteration of (3.35) and (3.59), the cost function decreases, and since the cost function is nonnegative, the sequence of the values of the cost is a non-increasing lower bounded sequence, and hence, with proper selection of step sizes γ and θ, the iterations always converge to joint equilibrium point.

Example 2: We again use the network we introduced in Example 1. This network is defined in the area of a rectangle with length 1 and width 0.01 and $r(x, y) = 100$ and $K(x, y) = 1$ for every (x, y). There are two destinations in the network, Destination 1 is initially located at point $(0.15, 0.005)$, and destination 2 is initially located at $(0.5, 0.005)$. We assume $w_1 = w_2 = 0.5$. This network and the used cartesian coordinate frame are shown in the top plot of Figure 3.3. The asterisk in this figure shows the initial location of Destination 1, and the circle shows the initial location of Destination 2.

As we saw before in Example 1, the value of $\vec{D}(x, y)$ for the initial locations of destinations is:

$$\vec{D}(x,y) = \begin{cases} 100x\hat{\imath} & if\ \ x < 0.15 \\ (100x - 50)\hat{\imath} & if\ \ 0.15 < x < 0.5 \\ (100x - 100)\hat{\imath} & if\ \ 0.5 < x < 1 \end{cases} \tag{3.61}$$

which results in the value of cost function to be 589.5. Since $\vec{D}(x, y)$ does not depend on y, we use the value of $0.5(D_x(x_i^-) + D_x(x_i^+))$ for calculating the value of the average $\vec{E}(x, y)$ at the two destinations. Note that $K(x, y) = 1$,

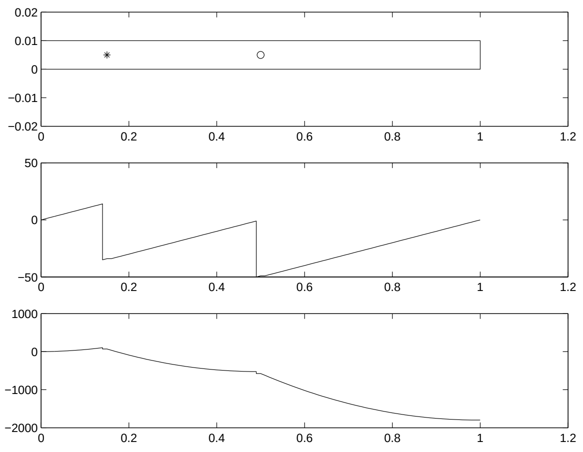

Figure 3.3: Top: the network with the asterisk showing the initial location of Destination 1 and the circle showing the locations of Destination 2. Middle: the value of $\vec{D}(x,y)$ versus x for initial placement of the destinations, Bottom: the corresponding potential function $U(x,y)$ as a function of x

which implies $\vec{E}(x,y) = \vec{D}(x,y)$. This implies:

$$\begin{aligned} \vec{E}_1 &= 0.5(D_x(0.15^-) + D_x(0.15^+) = 0.5 \times (15 - 35) = -10 \\ \vec{E}_2 &= 0.5(D_x(0.5^-) + D_x(0.5^+) = 0.5 \times (0 - 50) = -25 \end{aligned} \tag{3.62}$$

Since we move each destinations in the opposite direction of $\bar{E}_i$, the above means that both destinations should be relocated to a new location on their right. In order to update the locations of destinations we choose $\theta = 0.0004$ and use the following iterations:

$$x_i^{j+1} = x_i^j - \theta \vec{E}_i^j \tag{3.63}$$

in which x_i^j represents the value of the location of destination i in the iteration j; similarly $\bar{E}_i^j$ represents $\bar{E}_i$ in iteration j. It is straightforward to verify that

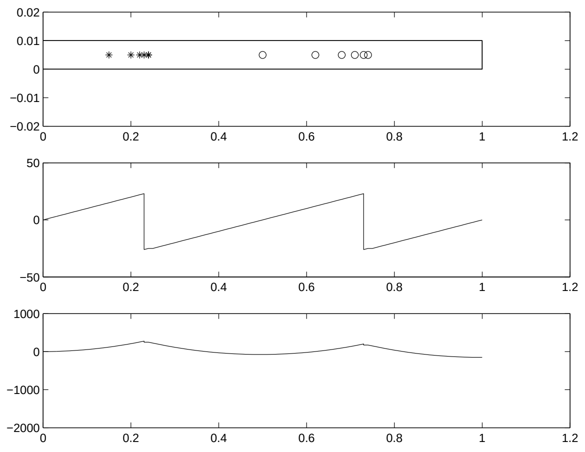

Figure 3.4: Top: the network with asterisks showing the locations of Destination 1 in different iterations and circles showing the locations of Destination 2 in different iterations. Middle: the value of $\vec{D}(x, y)$ versus x for the final solution, Bottom: the corresponding potential function $U(x, y)$ as a function of x for the final solution.

after 5 iterations the algorithm places Destination 1 at point $(0.25, 0.005)$, and Destination 2 at $(0.75, 0.005)$. Simple calculations show that this placement of destinations is optimal, and for both locations of the destinations we have $\bar{E}_i = 0$. The values of $\vec{D}(x, y)$, $U(x, y)$, and the locations of the destinations during the iterations are shown in Figure 3.4.

Another interesting observation about Example 2 is the way the cost function decreases during the iterations. The values for the 5 iterations are: 588.5, 314.5, 242.4, 221.5, 212.5, and finally, 210.5 for the optimal solutions. As the numbers show, the cost function monotonically decreases during the iterations. Furthermore, comparison of the initial value of the cost function with its value after the last iteration shows that using the iterations can have a significant improvement in the cost.

Chapter 4

Simulation Experiments for Sensor Networks

In this chapter we show the results of the simulations for the proposed methods in the previous chapters. The results are presented in different categories of single and multiple destinations.

4.1 Single Destination Experiments

In this simulation scenario sensors of the network are distributed in a $1000m \times 1000m$ square. The network area has been partitioned into $21 \times 21 = 441$ equal squares by equally spaced horizontal and vertical grids. The number of sensors $N = 441$. In each small square a sensor is placed randomly. The destination is placed in the center of the network area. The generation of the events inside each small square is done according to a Poisson process with a rate $0 < \lambda_i < 1$. λ_i can be considered as the integral of $r(x, y)$ on the area on which the i^{th} sensor collects messages. The sensor inside each small square is

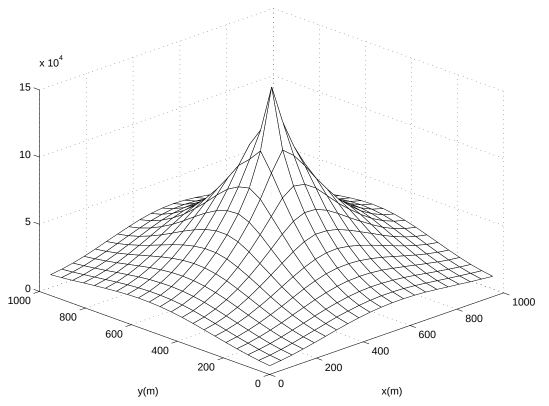

Figure 4.1: The value of the potential function U

responsible for all events that happen inside that square.

To start the network, we distribute the initial energy of 20000 units among the sensors. Since nodes closer to the destination need more energy because they have to do more switching, the initial energy assignment is done such that the initial energy of the sensors is inversely proportional to their distance from the destination. It should be emphasized that if the sensors are identical in terms of their energy, such assignment of initial residual energy can be done by changing the density of the sensors in the network design phase; in other words, sensors are distributed with a higher density for the places in the network that we need a higher energy for communication and forwarding activity. In our experiments each transmission or switching needs one unit of energy, and the transmission range of each sensor is $85m$. We define three scenarios for the experiments. In the first scenario, we illustrate the electrostatic routing and compare the routes generated by it with the shortest path routes. The second scenario compares the electrostatic routes with the

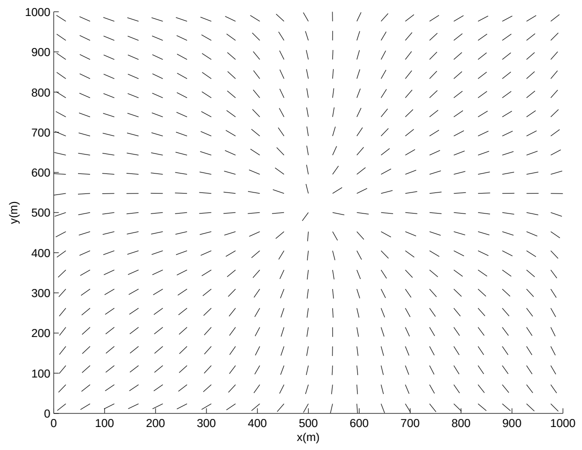

Figure 4.2: The direction of $\vec{D^*}(x, y)$: the line segments show the direction of $\vec{D^*}(x, y)$.

weighted shortest paths. In this case, the cost of the links that have a low energy sensor are increased. The third scenario illustrates how the electrostatic routes react in response to the change of residual energy of sensors, and how they avoid passing through low energy sensors and tend to the high energy sensors.

4.1.1 Scenario 1: General Evaluation

In this scenario we show the general performance of the electrostatic routing and illustrate its advantage over shortest path routes. In order to find the potential function $U(x, y)$, we have solved the partial differential equation given by (2.34) numerically, with the boundary conditions given by equation (2.35) on the 21×21 grid. Furthermore, we have made use of equation (2.37) for defining the value of $K(x, y)$ in terms of the residual energy. The resulting

$U(x,y)$ is shown in the Figure 4.1. The value of $\vec{E}(x,y)$ is found by taking the gradient of $U(x,y)$, and finally $\vec{D}(x,y)$ is calculated from $\vec{E}(x,y)$ by dividing $\vec{E}(x,y)$ by $K(x,y)$. Figure 4.2 shows the direction of $\vec{D}(x,y)$ at different places of the network. The line segments in this figure show the direction of the optimal load density vector field $\vec{D}(x,y)$. The paths from the sensors to the destination are found by following these line segments, and the routes are calculated by approximating the paths by the sequence of relaying sensors. The resulting routes from all sensors to the destination have been plotted in Figure 4.3. In this figure each star shows the place of a sensor in the network. The relatively high number of sensors allows us to find routes by approximating paths with the relaying sensors.

To have a basis of comparison we have also calculated the routes that use the shortest path to the destination. Figure 4.4 shows the routes calculated by this method. By comparing this figure with Figure 4.3, it can be seen that in the case of using optimal $\vec{D}(x,y)$ the routes are more evenly spaced in the network.

To evaluate the difference of using $\vec{D}(x,y)$ for routing to the case in which we use the shortest path routes, we turn the network on at $t = 0$, and let it run until the nodes run out of energy and no more communication is possible to the destination. In the case of using the routes based on $\vec{D}(x,y)$, the routes were updated periodically according to the updated values of the residual energy of the sensors. Also, during the shortest path experiment, if a sensor contributing to a path of another sensor to the destination ran out of energy, a new shortest path was calculated if there was at least one path

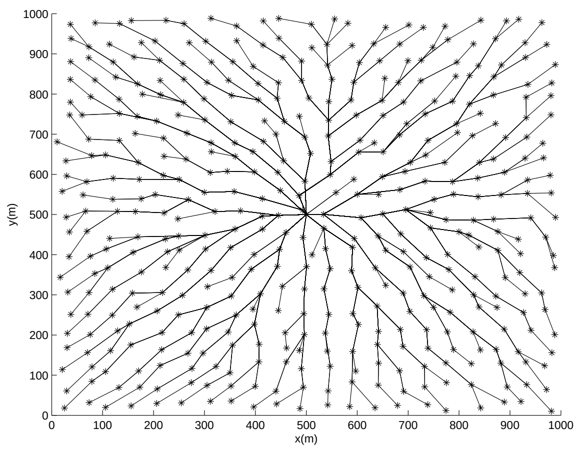

Figure 4.3: The routes from all sensors to the destination. These routes are found by using $\vec{D}(x, y)$.

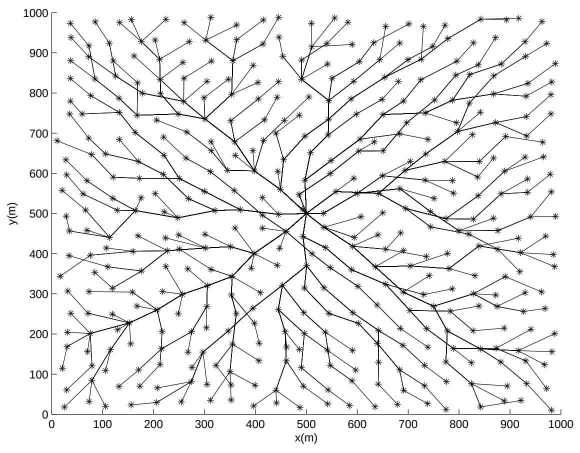

Figure 4.4: Shortest path routes to the destination.

Table 4.1: The performance comparison of routes based on $\vec{D}(x, y)$ with shortest path and weighted shortest path routes.

Exp.	$\vec{D}(x, y)$ routes	SP	Imp.(%)	WSP	Imp.(%)
1	2112	1734	22%	1925	10%
2	1901	1465	30%	1632	16%
3	1928	1681	15%	1685	14%
4	1744	1278	36%	1412	24%
5	1839	1233	49%	1346	37%
6	1761	1592	10%	1610	9%
7	1749	1414	24%	1520	15%
8	1774	1193	49%	1311	33%

through the live sensors.

The simulation showed that for the case of using $\vec{D}(x, y)$, the total number of 2112 messages were delivered to the destination, and for the case of shortest path routing the total number of delivered messages was 1734.

Several other experiments were done with the same conditions as in the above experiment but with different randomly generated locations of the nodes and the traffic sources. The results are shown in Table 6.1. The second column of this table shows the total number of delivered messages for the case in which we use the routes generated by $\vec{D}(x, y)$, and its third column shows the same quantity for the shortest path case. It can be seen that in all cases the number of delivered messages was increased considerably, and the average increase was 29%.

4.1.2 Scenario 2: Comparison with Weighted Shortest Path

In this set of simulation experiments the performance of electrostatic routes was compared with the performance of the weighted shortest path routing method. In the weighted shortest path method, the weight (cost) of a wireless link is defined as a decreasing function of the residual energy of its source sensor. For link l, with the source sensor i, we define the cost c_l as:

$$c_l = \frac{1}{e_i}. \tag{4.1}$$

Recall that e_i is the residual energy of the sensor i. Since the residual energy of the sensors changes over time, e_i is a function of time, and the weighted shortest path to the destination changes as energy of some sensors deplete. This fact was taken into account in the simulation experiments, and the sensors maintained periodical updates about the value of the link costs.

It is important to note that the definition of the link weights in equation (4.4) is consistent with the way the value of $K(x, y)$ is defined in equation (2.37); in the former case the weight of the links are inversely proportional to the residual energy of their source sensors, and in the later case the permittivity factor is inversely proportional to the residual energy. Therefore, the similar definitions give a fair basis for the performance comparison of the electrostatic routes with the weighed shortest path routes.

The 5th and 6th columns of Table 6.1 show the performance of the

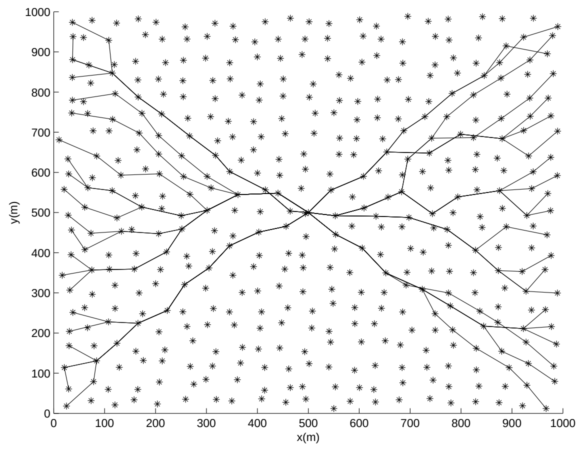

Figure 4.5: The placement of sensors and shortest path routes for the base experiment. Only sensors on the left and right generate messages.

weighted shortest path. The 5th column of this table shows the total number of delivered messages by using weighted shortest path routes. The 6th column of this table shows the percentage of performance improvement by using electrostatic routing instead of weighted shortest path. The same 8 different experiments of the previous scenario were repeated, and as can be seen in the table, the average performance improvement of electrostatic paths over weighted shortest paths was 19.8%.

4.1.3 Scenario 3: Illustrating Dependence of Electrostatic Routes on Residual Energy

The purpose of this set of experiments is to illustrate how changing the value of residual energy can change routes. For this experiment a network similar to that explained in the previous experiment was used, but in this network,

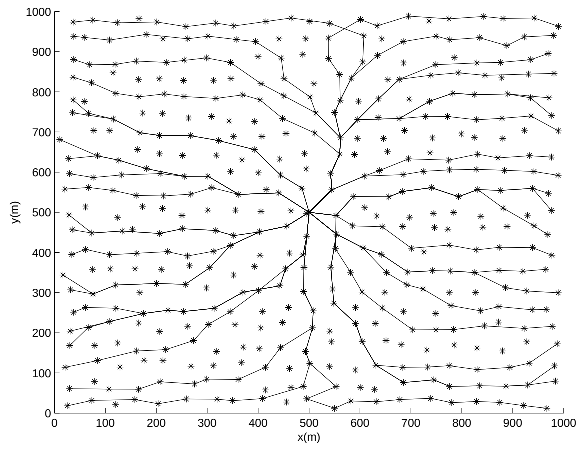

Figure 4.6: The placement of sensors and electrostatic routes for the base experiment.

the messages were generated only by the sensors on the very right or very left of the network, and the sensors in between were only used for relaying the messages of the sensors of the two sides to the destination. Mathematically, this means that $\rho(x, y)$ is non-zero for the areas of the network that are close to the right and left boundaries. Again 441 nodes were placed in the network randomly and in the way described before, and residual energy was initialized in the same way as in the previous experiments. Figure 4.5 shows the placement of the sensors and the shortest path routes to the destination, and Figure 4.6 shows the same sensors with routes generated according to $\vec{D}(x, y)$. Comparison of the two figures shows that the electrostatic routes are more evenly distributed in the network resulting in a better use of energy. In a simulation run for this case, the shortest path routes gave delivery of 405 messages to the destination, while this value increases to 692 for the case where we used routes based on $\vec{D}(x, y)$. We refer to this experiment as the base experiment.

In the next experiment the value of residual energy of a few sensors was changed to observe how it affects routes. For this purpose the residual energy of the sensors shown by the circles in Figure 4.7 were reduced to 1/3 of their value in the base experiment. As can be seen in this figure, the routes avoid passing through the circle nodes. This can be seen by comparing the routes in the left and the right of the network. While routes in the right of the network are evenly distributed in the network, the routes in the left are tending to the top left and bottom left in order to avoid passing through the low residual energy area. In this case the traffic of two sensors in the left passes through the low energy area, while in the base case, the routes of 7 sensors in the left pass through this area (by inspecting Figures 4.7 and 4.6). In the simulation, the shortest path routes resulted in delivery of 269 messages, while the electrostatic routes delivered 632 messages to the center, which is more than a 100 percent improvement.

The last experiment is the opposite of the previous experiment. In this experiment the initial residual energy of the sensors shown by circles was initialized to twice their value in the base experiment. Figure 4.8 shows the resulting routes. It can be seen that the many routes are attracted by the high energy area. This can be observed by comparing the routes in the left half to those in the right half of the figure. In this case, the cumulative traffic of 14 sensors on the left pass through the high energy area. In the simulation, the shortest path routes deliver 495 messages for this case, and by using electrostatic routes this number increases to 816. It is interesting to note that the behavior of routes in the neighborhood of high or low residual energy places of the network is a reminder of the behavior of electric field in

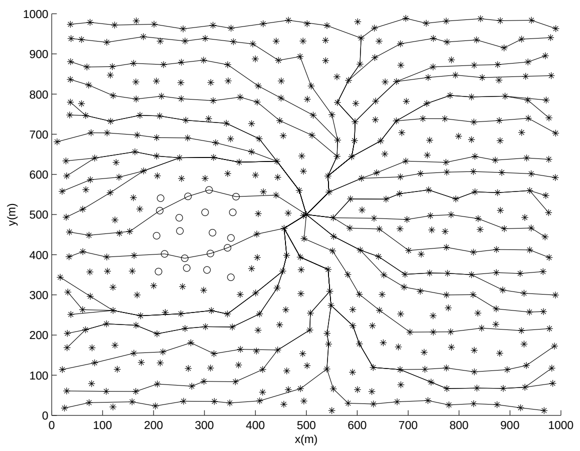

Figure 4.7: The effect of decreasing energy on routes. The routes avoid passing through the low energy sensors shown by circles.

the neighborhood of high or low permittivity environments. The flux lines of electric field tends to pass through the higher permittivity places, which is very similar to the way the routes behave here.

4.2 Multiple Destination Experiments

In this section we present the simulation experiments with multiple destinations. The network is a 1×1 area and we assumed the total load to be 100, which was uniformly distributed in the network (i.e., $r(x, y) = 100$). With this basic setting, we did several experiments that we present in different scenarios.

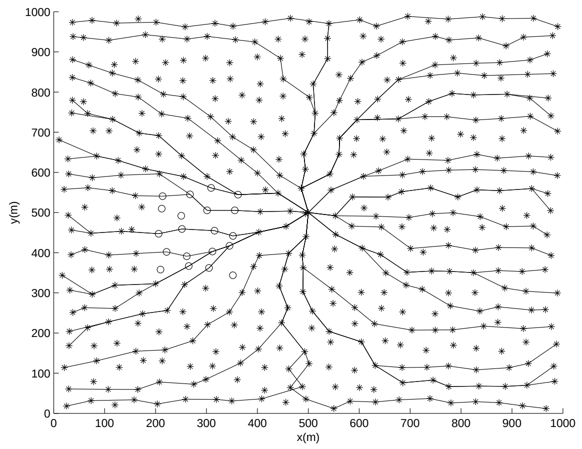

Figure 4.8: The effect of increasing energy on routes. The routes tend to pass through the high energy sensors shown by circles.

4.2.1 Scenario 1: Four Destinations without Relocation of Destinations

In this experiment, we placed 4 destinations in the network. The destinations are located at:

$$\begin{aligned} (x_1, y_1) &= (0.45, 0.45) \\ (x_2, y_2) &= (0.75, 0.75) \\ (x_3, y_3) &= (0.65, 0.25) \\ (x_4, y_4) &= (0.25, 0.75) \end{aligned} \tag{4.2}$$

Next we divided the total load of 100 units evenly among the destinations and assumed $K(x, y) = 1$. This means that $w_1 = w_2 = w_3 = w_4 = 25$. Then we solved the PDE equation for the potential function $U(x, y)$, and from it we found $\vec{D}(x, y)$. The resulting potential values are shown in Figure 4.9. The equipotential lines of the the potential function are shown in Figure 4.10. The value of $\vec{D}(x, y)$ is shown in Figure 4.11 and the regions of attraction for the

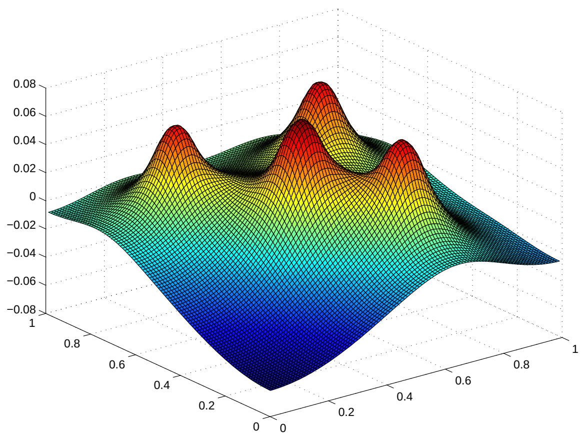

Figure 4.9: The value of the potential function $U(x, y)$ for the case with four destinations

four destinations are shown in Figure 4.12. The value of potential function at the destinations is: $U_1 = 0.0723$, $U_2 = 0.0601$, $U_3 = 0.0607$, and $U_4 = 0.0571$. The total value of the cost function in this case is 4.88. Since the value of potential function is not the same at the location of destinations, we know that we can update the weight of destinations to reduce the cost function.

In the next experiment of this scenario, we use the iterations of equation (3.35) to update the weights of destinations in order to reduce the cost function. The calculations show that the algorithm converges to the weight values within 1 percent of the optimal weights in 3 iterations. The values of optimal weights are:

$$\begin{aligned} w_1 &= 22.07 \\ w_2 &= 25.50 \\ w_3 &= 25.94 \\ w_4 &= 26.48 \end{aligned} \tag{4.3}$$

With the above values of weights, the cost function reduces to 4.07. We have

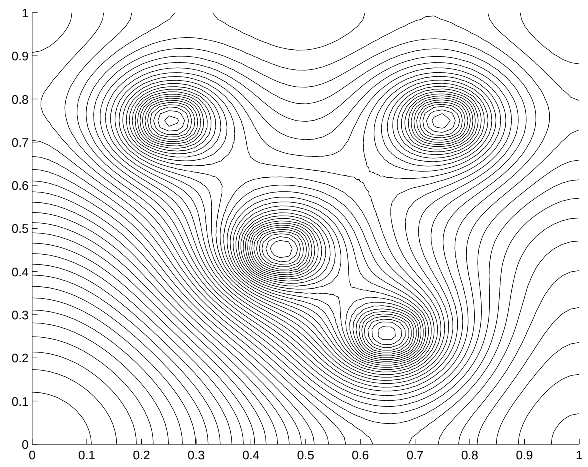

Figure 4.10: The equipotential lines for the case with four destinations

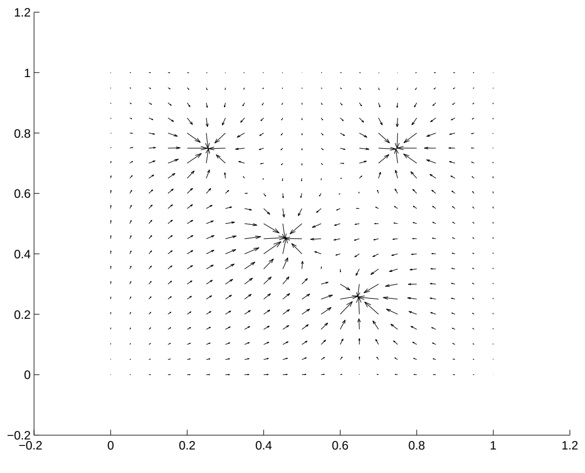

Figure 4.11: The value of $\vec{D}(x, y)$ at different places of the network for the case with four destinations

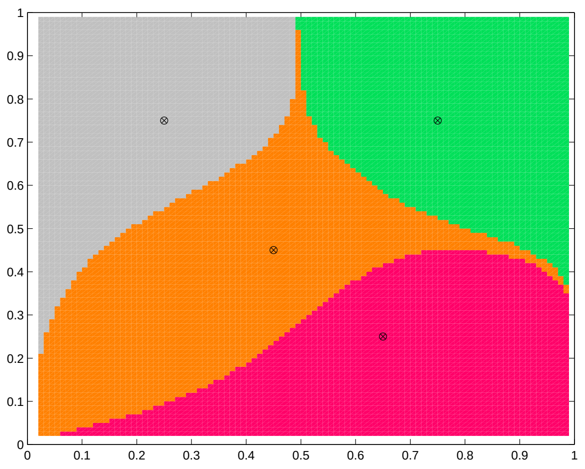

Figure 4.12: The regions of attraction for different destinations in the case of four destinations. Destinations are shown by circles.

used the gradient step size $\gamma = 200$, and for the stop criterion of iterations, we used $\xi = 0.001$. Also, the value of potential function at the destinations was calculated to be $U_1 = U_2 = U_3 = U_4 = 0.062$.

4.2.2 Scenario 2: Four Destinations with Relocation of Destinations

In this scenario we use the value of $\bar{\vec{E}}_i$ to relocate the destinations. We start with the example of four destinations in the previous case, with equal assignment of the weights. Our simulation shows that after 6 iterations, the destinations are relocated to the final positions of:

$$\begin{aligned}(x_1, y_1) &= (0.251, 0.257)\\(x_2, y_2) &= (0.750, 0.748)\\(x_3, y_3) &= (0.748, 0.249)\\(x_4, y_4) &= (0.251, 0.759)\end{aligned} \tag{4.4}$$

As can be seen, the algorithm has successfully placed the four destinations evenly in the centers of the four quarters of the network. This scenario shows the effectiveness of our method for relocating the destinations. Figure 4.13 shows the location of destinations during the iterations. Squares in this figure show the initial placement of the destinations, and circles show the final placement after the iterations. As can be seen in this figure, the algorithm has a fast convergence to the optimal locations. The other interesting result of this experiment is how the cost function changes as we relocate the destinations. This information is shown in Figure 4.14. This figure shows the value of cost function during the iterations. As can be seen in this figure, the cost function monotonically decreases during iterations. The value of cost in the final solution is 3.167, which is significantly improved compared to the initial value of 4.88.

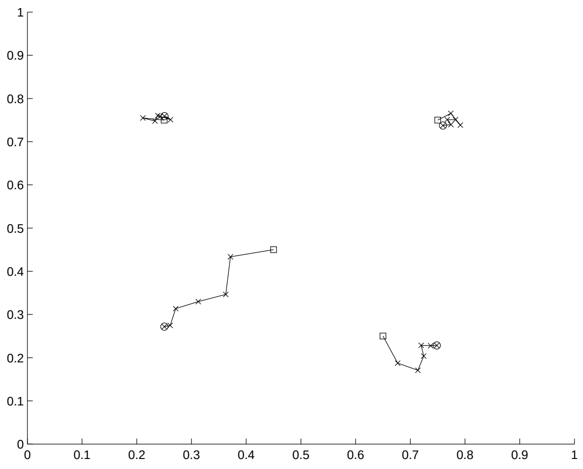

Figure 4.13: The location of destinations during the relocating iterations. Squares show the initial placement of the destinations, and circles show the final placement after the iterations. The algorithm shows a fast convergence to the optimal positions.

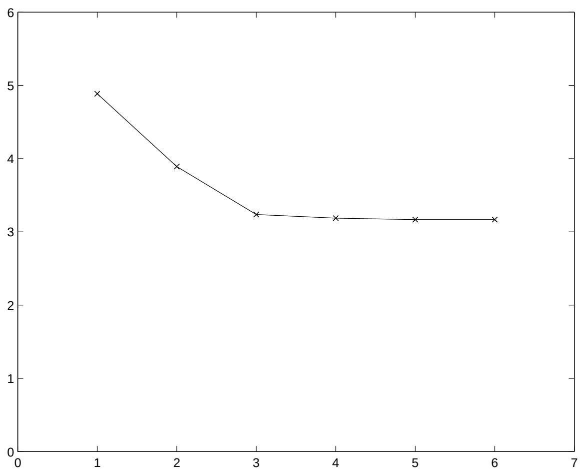

Figure 4.14: The cost function during the iterations of relocating destinations. The value of cost monotonically decreases during the iterations.

Chapter 5

TCP Conformance Tests

5.1 Introduction of TCP Responsiveness and Conformance

A key characteristic of TCP traffic is its response to packet drops. This forms the basis for congestion control. The degree to which a TCP aggregate reduces its rate in response to packet drops depends on packet size, round trip time and the distribution of window sizes among the constituent flows. A TCP aggregate may also include noncooperative or malicious flows that do not participate in the TCP congestion control algorithm. Such flows are called *nonconforming.*

In this chapter we introduce a technique for quantifying the responsiveness of a TCP aggregate to packet drops and for estimating the fraction of traffic that is nonconforming to TCP protocol specifications. Such techniques are useful for congestion control based on random early drop [57]. With quantitative information about the responsiveness of TCP traffic, when a router gets close to congestion it will know how many drops are needed to keep the rate of the traffic within the capacity of its outgoing links.

Another application of conformance tests is detection of distributed denial of service (DDoS) attacks. As perpetrators become more sophisticated, it can be anticipated that DDoS attacks will become increasingly stealthy with attack traffic designed to closely resemble ordinary Internet traffic. Studies show that more than 90 percent of the Internet traffic is generated by TCP traffic sources [58]. Furthermore, http traffic accounts for more than 42 percent of current traffic, while the average amount of packets exchanged per http flow is on the order of 10 packets. Consequently, a stealthy attacker might well choose to clog access links by generating large numbers of short-lived TCP flows. Such flows would be difficult to distinguish from ordinary traffic; they would also be difficult to trace back. Even if an attack source generated packets over a significant duration of time, by changing the spoofed source address or the source port number it could make the traffic appear to be composed of many small flows.

An important feature of this work is that we apply the conformance tests at aggregate level, not at flow granularity. Performing the conformance test on aggregates has several advantages. First, our approach scales with the number of flows, and many flows can be bundled together to form an aggregate and the conformance test is done for the resulting aggregate. The second advantage of aggregate based testing is the fact that the majority of current traffic of the Internet is composed of short-lived flows known as the Internet mice [58]. It is extremely hard to perform tests on such traffic at flow level because many flows have a small number of packets and are active only for a few round trip times, and often they end before a router can keep track of them. However, if we consider many such flows together, we get an aggregate that is composed of many flows that appear, survive for a few round trip times, and disappear; the aggregate composed of these flows has some statistical properties that can help us to define a conformance measure for it.

Our first approach to measure the conformance of an aggregate is to perturb the arrival rate of that aggregate by intentionally dropping a very small number of packets, and observing the way the rate of the aggregate responds. A normal TCP aggregate shows a predictable response as a result of instantaneous packet drops, and we estimate this response and use it as a *conformance measure* or the *conformance coefficient* of the aggregate. By doing this periodically, the conformance of the aggregate can be determined. We call this the Aggregate Perturbation Method (APM).

One complication of APM in a distributed implementation is the possible interference due to simultaneous tests being performed by different routers. Flows in an aggregate may experience perturbations at multiple routers. In a distributed implementation, in order to perform its test, a router should not need to be aware of the perturbations applied by other routers. One approach to solve this problem is inspired by the direct sequence spread spectrum CDMA approach in multiple access communication channels. Each router is assigned a dropping signature that specifies its packet dropping rate as a function of time. Different routers are assigned signatures that are orthogonal in a certain sense. It can be shown that under certain assumptions, this approach enables each individual router to find the conformance coefficient of the aggregates passing through it without requiring any information to be shared with the other routers. We have named this method CDMA based Aggregate Perturbation Method (CAPM).

A technical limitation of the original CAPM is the fact that the nominal conformance coefficients depend on the traffic characteristics such as the average lifetime of the flows, the round trip times, and the statistical distribution of the congestion window size. In other words, when we drop a single packet from an aggregate, we do not know how much rate reduction to expect unless we have some statistical measurements of the aggregate during a long enough interval in the past.

In order to make the nominal conformance coefficients of the aggregates independent of the traffic characteristics we offer two alternative CAPM algorithms for detecting and mitigating a DDoS attack. The new algorithms work based on testing conformance of legitimate TCP connections in retransmitting the lost packets or not advancing to the next state of the TCP finite state machine when specific packets are dropped during the 3-way handshake at the start of a connection. Both algorithms are based on the observation that at the time of a DDoS attack the conformance coefficients of aggregates with a high proportion of DDoS packets show a significant deviation from their nominal values.

In one of the extensions of the CAPM algorithm, we apply the method to TCP SYN packets, which are generated at the start of a TCP connection. Our test is based on perturbing the traffic by drooping the initial SYN packets of TCP connections with a very small probability. If no attack is present, we expect such perturbations show a predictable response in the rate of ACK packets that are generated as a part of the 3-way TCP handshake. Note that a TCP connection is started by a SYN packet from a client to a server that initiates the request, and it is responded by a SYN/ACK packet from the server to client, and finally by an ACK from the client to the server to complete establishment of the connection. Our proposed perturbation test on SYN packets is designed to detect a TCP SYN flood attack, when many DDoS sources repeatedly request a connection by sending SYNs to a server and not completing the connection. Our proposed approach has the capability of detecting blind transmission of ACK packets without needing to assume that paths between the DDoS source and the victim are symmetric. By a blind transmission of the ACK, we mean an ACK generated by a DDoS source before receiving a SYN/ACK packet from the TCP server. Such ACKs may be generated by the DDoS sources so that the aggregates containing DDoS attack SYNs contain an equal number of ACK packets, which camouflage the attack by making the traffic similar to the normal traffic if no perturbation is applied.

In another form of CAPM algorithm, we apply the test to the aggregates of TCP data packets being observed at a router, and we introduce a simple method to detect packet retransmission rate at the router. Similar to the previous test, if no attack is present, we expect the detected packet retransmission rate of an aggregate to follow the shape of the function that was used to perturb that aggregate by dropping packets from it. Our simulation experiments show that both of the above introduced tests give excellent performance in detecting nonconforming traffic and determining the proportion of DDoS packets in an aggregate.

In contrast to the original CAPM, the extensions of CAPM algorithm that we offer make the nominal conformance coefficients of aggregates independent of their statistical characteristics. Such modifications make the approach more suitable for DDoS detection and defense. Significance of this extension is the fact that the modified tests do not need the history and state of the test results in the past in order to detect a DDoS attack. Furthermore, the new tests can be applied to the finer granularity sub-aggregates of an aggregate with a positive primary test to precisely locate the parts of the aggregate with higher concentration of DDoS traffic. The basic CAPM tests do not work well when an aggregate is divided into sub-aggregates. This is because after dividing an aggregate into sub-aggregates the nominal values of the conformance coefficients of the sub-aggregates (which depend on the past history of the statistical information of sub-aggregates) are not known.

5.2 Related Work

There is a large literature on techniques for combatting DDoS attacks. Several approaches have been proposed including pushback, traceback, and ingress filtering. Pushback [59][60][61], includes detection of attack, identification of an attack signature, and advertisement of a filter to the upstream routers to rate limit the attack traffic. Traceback techniques [62][63][64][65][66][67][68] [69][70][71][72][73][74] are designed to determine the path, and ultimately the source, of the attack. There are two main classes of traceback methods: the first uses explicit ICMP messages [63][64][68], while the second marks the packets by writing partial information regarding the paths in unused fields of IP headers [65][72][73][74].

Authors of [64] propose that routers store hashed information on recently received packets to recover the paths to the packet sources in the event that traceback is needed. In ingress filtering [75], edge routers check the validity of the source IP addresses of the packets. A packet with a source IP address that does not belong to any of the valid sources in the network is filtered by the edge router of that network when attempting to exit. D-WARD [76] monitors the traffic at the egress router of a stub domain in order to determine whether the ratio of outgoing to incoming traffic for a set of remote addresses is abnormally high. A high ratio is taken as a signal that an attack is being mounted from within the stub domain.

Many researchers have conducted studies to do identification and modeling of TCP traffic in the granularity of flow under steady state conditions [77][78][79][80]. In [77] the authors propose a method of testing a flow by comparing the steady state throughput of a TCP flow with the theoretical predicted value for conforming (responsive) flows. The objective of that study is to identify and penalize nonconforming flows for congestion control purposes. The TCP Friendly congestion control schemes are generalizations of flow based TCP congestion control mechanisms to the general flows [78]. Stochastic Fair Blue [81] offers a per flow test for responsiveness by mapping different flows to parallel bins. The approach is based on the fact that the bins containing a nonconforming flow are likely to be overloaded. However, if there are many nonconforming flows in a traffic aggregate, it is likely that many bins are overloaded. In LRU-RED [82] a router tries to drop less from responsive high bandwidth flows at the time of congestion.

Our approach is distinguished from this body of existing work in several respects: (1) It applies the concept of conformance to aggregates rather than to individual flows. (2) It determines conformance by actively dropping packets, rather than relying on congestion-induced drops. (3) It can be applied in a distributed manner without requiring communication among routers in order to detect a DDoS attack.

5.3 Aggregate Perturbation Method

In this section, we introduce the Aggregate Perturbation Method (APM) for quantifying the responsiveness of TCP aggregates to packet drops by reducing the sending rate to the network. APM works based on instantaneously dropping a number of packets from an aggregate at some point and observing the resulting transient decrease in the rate of the aggregate.

We assume the TCP aggregates are composed of TCP flows that conform to TCP-Reno congestion control mechanism. TCP-Reno has two different phases known as *slow start* and *congestion avoidance*. Slow start begins after making a connection, and upon successful transmission of every packet and receiving acknowledgement from the receiver the window size is increased by one. Congestion avoidance starts after the window size exceeds a threshold value, and in this phase the window size is increased one per round trip time, and upon experiencing a drop it is decreased to half its current value.

Assume at some router we have an aggregate of TCP flows with arrival rate of $\lambda(t)$. In order to test the aggregate for responsiveness, at time $t = 0$, we drop D packets from it. It is expected that the aggregate responds to the packet drops by decreasing its rate for a while after time $t = 0$. We define the following responsiveness measure for the aggregate as a response to packet drops:

$$\eta(D) = \int_0^{t_r} (\lambda(0^-) - \lambda(t))\, \mathrm{d}t \tag{5.1}$$

in which $\lambda(0^-)$ is the instantaneous rate at the moment before dropping the first packet, and t_r is a nonnegative finite time, and it can be chosen to be the minimum time for the recovery of all flows that received drops (in the order of a few times the longest round trip in the aggregate). To achieve better results, $\lambda(0^-)$ may be replaced by a short-term average of the rate of the aggregate in a time interval earlier than $t = 0$. $\eta(D)$ is simply a measure of how many more packets could have been sent by the aggregate if we had not dropped D packets. This measure is illustrated in Figure 5.1.

Our approach for quantifying responsiveness of a TCP aggregate is based on the responsiveness $\eta(D)$ as a response to packet drops; under the same value of D for different aggregates, those with a higher $\eta(D)$ are more responsive. In other words, $\eta(D)/D$ can give a quantitative value of the responsiveness of an aggregate.

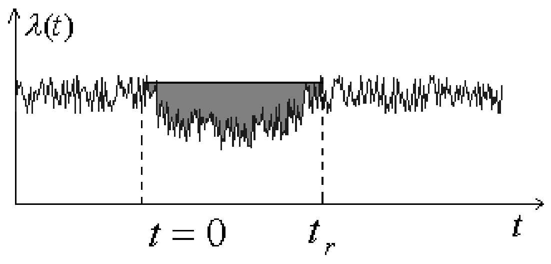

Figure 5.1: The shaded area shows $\eta(D)$, the responsiveness measure of aggregate defined by equation (5.1). D packet are dropped from the aggregate at $t = 0$.

5.4 CDMA Based Aggregate Perturbation Method (CAPM)

One of the problems of distributed implementation for APM is the potential of simultaneous perturbations; the measurements of a perturbing router on an aggregate can be falsified by the simultaneous perturbations being done on the same aggregate in a downstream or upstream router. This phenomenon is illustrated in the Figure 5.2. As it can be seen in this figure, the response of the APM test of a router at $t = t_1$ is overlapped by the response of the aggregate to another router's test at time $t = t_2$, which causes interference. This interference happens when t_1 and t_2 are close enough to each other (more precisely $|t_2 - t_1| < t_r$). In this case the measure given by equation (5.1) does not give accurate information about responsiveness of the aggregate, and interference causes the results of both tests to be falsified.

In this section we introduce CAPM to overcome the above problem. In CAPM every perturbing router uses a unique perturbing pattern. We will show that under proper assignment of the perturbing patterns and proper definition of aggregate responsiveness measure for each perturbing router, the test and measurement of each router will be robust to the interference caused by the other simultaneous perturbing routers.

CAPM is different from APM in two ways. The first difference is that we spread the packet drops over time. In other words, instead of dropping D packets from the aggregate instantaneously at time $t = 0$, we spread the packet drops over a time interval $[0, T]$. In this scheme perturbation is done according to the packet drop rate function $r_i(t) : [0, T] \rightarrow \mathcal{R}$ for the i^{th} router. The responsiveness test is done during the interval $[0, T]$, and at time $t \leq T$, the i^{th} router drops $r_i(t)$ packets per second from the aggregate. Since many TCP flows send packet bursts, the CAPM spreads the packet drops over interval [0,T], so that the probability of having more than one packet drop of the same connection is reduced. We refer to $r_i(t)$ function as the *drop signature* of the i^{th} router.

The second difference between CAPM and APM is the way we define the responsiveness measure for the i^{th} router as the response to dropping with rate $r_i(t)$. In this case instead of the simple integral given by equation (5.1), we use a weighted integral to measure responsiveness of the aggregate under perturbation:

$$\eta_h(r_i) = \int_0^T h(t)\Delta\lambda(t)dt \qquad (5.2)$$

in which $\Delta\lambda(t) = \lambda(0^-) - \lambda(t)$, and $h(t)$ is a weighting function that states at what time instants the results are more important to us, and at what time instants we are less interested in the rate decrease of aggregate.

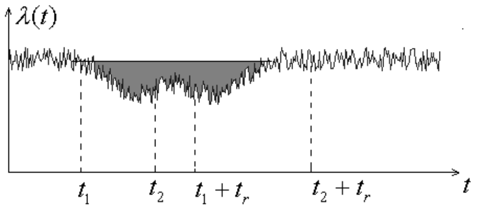

Figure 5.2: The interference effect of simultaneous APM tests done by different routers. Before the aggregate recovers from a router's perturbation at $t = t_1$, another router performs a test at $t_2 < t_1 + t_r$. The results of both tests are falsified.

In the next step we try to use an approach similar to Direct Sequence Spread Spectrum CDMA in multiple access communication to solve interfering problems of multiple simultaneous perturbing routers. In this approach, each router perturbs the traffic according to its unique drop signature based on a CDMA code assigned to it. The idea is that if we define the drop signature of different routers so that they are orthogonal to each other in a certain sense, then by proper definition of the weight function $h(t)$ the measure of responsiveness in a router defined in equation (5.2) will be independent of the perturbations done by the other routers.

Similar to the CDMA systems, we define the drop signature of the i^{th} perturbing router in the following way:

$$r_i(t) = A_i \sum_{j=1}^{N} c_j p_{T_c}(t - (j-1)T_c) = A_i s_i(t) \tag{5.3}$$

in which A_i is a known perturbation amplitude of the i^{th} router, N is a positive integer called the spreading factor, $T_c = T/N$, $(c_1, c_2, \ldots, c_N)$ is a binary sequence assigned to the particular router known as the code of the router. In (5.3), $s_i(t)$ denotes the *normalized drop signature,* and $p_{T_c}(t)$ is a real-valued function known as the chip waveform and it satisfies the following property:

$$\int_{-\infty}^{\infty} p_{T_c}(t) p_{T_c}(t - nT_c)\,\mathrm{d}t = 0, \;\; n = 1, 2, \ldots. \tag{5.4}$$

The measurement of the i^{th} router about the responsiveness of the aggregate is made based on the *Matched Filter* output. The matched filter output is the value of $\eta_h(r_i)$ evaluated at $h(t) = s_i(t)$:

$$y_i = \int_0^T s_i(t) \Delta\lambda(t)\,\mathrm{d}t. \tag{5.5}$$

Since in our problem $r_i(t)$ is a drop rate, it should be nonnegative, and hence $p_{T_c}(t)$ should be nonnegative. For this purpose we suggest the popular simple rectangular chip waveform:

$$p_{T_c}(t) = \begin{cases} 1 & if \;\; 0 < t < T_c \\ 0 & otherwise. \end{cases} \tag{5.6}$$

Usually, in the CDMA systems assignment of the codes is very important. Users with a potential of high interference (e.g., neighbor routers in our problem) are assigned codes that cause their drop signatures to be orthogonal (or close to orthogonal)

$$\int_0^T s_i(t) s_j(t) dt = 0, \;\; for \;\; i \neq j. \tag{5.7}$$

Unfortunately, the statement of (5.7) cannot be satisfied with the current definition of drop signatures defined in (5.3). That is because both $s_i(t)$ and $s_j(t)$ are nonnegative rate functions, and hence the integral defined in (5.7) can never be zero. We can solve this problem by making a minor change of the orthogonality requirement and the structure of the matched filter. First, we replace the orthogonality condition by a similar condition in which the normalized drop signatures are orthogonal after removing their DC components:

$$\int_0^T s_i^a(t) s_j^a(t)\,\mathrm{d}t = 0, \;\; for \;\; i \neq j \tag{5.8}$$

in which $x^a(t)$ is $x(t)$ after eliminating its DC component over $[0, T]$:

$$x^a(t) = x(t) - \frac{1}{T}\int_0^T x(t)\,\mathrm{d}t \tag{5.9}$$

Furthermore, we change the matched filter output for the i^{th} router in the following way:

$$y_i = \eta_{s_i^a}(r) = \int_0^T s_i^a(t)\Delta\lambda(t)\,\mathrm{d}t \tag{5.10}$$

y_i is the value of η_h in (5.2) evaluated for $h(t) = s_i^a(t)$. One important fact about notation $\eta_h(r)$ in (5.10) is that in this equation r is the total perturbing function, since the rate decrease $\lambda(0^-) - \lambda(t)$ is affected by this total drop rate (i.e., $r(t) = \sum_k r_k(t)$, where k is an index that covers the set of all router perturbations that the aggregate experiences). It can be shown that if the total drop rate $r(t)$ is small enough compared to the rate of aggregate, then the system with input $r(t)$ and output $E[\Delta\lambda(t)]$ can be approximated by a linear system. In other words, the system can be linearized around its operating point.

Now we state the following theorem; for the purpose of this theorem we assume $r_k(t)$ are piecewise constant functions as it was defined in (5.3).

Theorem 5: Assume that the overall drop rate $r(t) = \sum_k r_k(t)$ is small enough such that the system with input r and output $E[\Delta\lambda]$ can be approximated by a linear system. Furthermore assume the holding time of the piecewise constant functions $r_k(t)$ on each constant interval is large enough compared to the response time of the aggregate. Then under the orthogonality assumption of (5.8) we have:

$$E[y_i] = E[\eta_{s_i^a}(r)] = E[\eta_{s_i^a}(r_i)] \tag{5.11}$$

Note that the middle term of equation (5.11) is the measure of responsiveness with the weight function $h(t) = s_i^a(t)$ when all routers perturb the aggregate; however, the right term is the measure of responsiveness with the same weight function when only the i^{th} router perturbs the aggregate. The significance of Theorem 5 is that it states under orthogonality condition of equation (5.8) that the expected responsiveness measure at router i, $E[\eta_{s_i^a}(r)]$, is independent of perturbations being done at the other routers.

Proof: Let $\Delta\lambda^x(t)$ to be the rate change of aggregate when it is perturbed with drop rate $x(t)$. By definition we have:

$$E[\eta_{s_i^a}(r)] = \int_0^T s_i^a(t) E[\Delta\lambda^r(t)]\, \mathrm{d}t. \tag{5.12}$$

From the assumption that $E[\Delta\lambda^r]$ is linear in r we can conclude:

$$E[\Delta\lambda^r(t)] = E[\Delta\lambda^{r_i}(t)] + \sum_{j\neq i} E[\Delta\lambda^{r_j}(t)]. \tag{5.13}$$

Substituting (5.13) in (5.12) yields:

$$E[\eta_{s_i^a}(r)] = E[\eta_{s_i^a}(r_i)] + \sum_{j\neq i} E[\eta_{s_i^a}(r_j)]. \tag{5.14}$$

To complete the proof, it suffices to prove $E[\eta_{s_i^a}(r_j)] = 0$ for $j \neq i$. We have $r_j(t) = A_j s_j(t)$. Now we use the assumption that $r_j(t)$ changes slower than the aggregate response time. Hence $r_j(t)$ can be approximated by using a piecewise constant function. For an interval on which $r_j(t)$ is constant, the traffic aggregate responds and settles down to a value. In the next interval $r_j(t)$ jumps to a new value, and so $\Delta\lambda^{r_j}(t)$ responds accordingly, and after experiencing a small transient time settles down to a new steady state value. According to the linearity assumption $\Delta\lambda^{r_j}(t)$ on each interval is proportional to the constant value of $r_j(t)$ on that interval. This means that $E[\Delta\lambda^{r_j}(t)]$ tracks the piecewise constant shape of $r_j(t)$. So by ignoring the short transients of $E[\Delta\lambda^{r_j}(t)]$ at the beginning of each interval we will have:

$$E[\Delta\lambda^{r_j}(t)] \approx C_j r_j(t) = C_j A_j(s_j^d + s_j^a(t)) \tag{5.15}$$

in which s_j^d is the DC component of $s_j(t)$ over interval $[0, T]$. Recall

$$E[\eta_{s_i^a}(r_j)] = \int_0^T s_i^a(t) E[\Delta\lambda^{r_j}(t)]\,\mathrm{d}t. \tag{5.16}$$

Substituting (5.15) in (5.16) and using orthogonality assumption of (5.8) yields: $E[\eta_{s_i^a}(r_j)] = 0$. **QED**

In Theorem 5 we have assumed that the holding time of $r_k(t)$ on the intervals on which it is constant is large enough compared to the aggregate response time. Generally, the response time of an aggregate is characterized by the round trip time of the flows contributing to it. Therefore for the piecewise constant function $r_k(t)$, the length of each constant interval should be significantly larger than the typical round trip time of the flows in the aggregate. This condition can be satisfied by making T_c long enough (e.g., 10 to 20 times the typical round trip time).

One useful observation about (5.10) is:

$$\int_0^T s_i^a(t)\lambda(0^-)\,\mathrm{d}t = 0 \tag{5.17}$$

And so we have the following simple equation for the output of the matched filter for the i^{th} router:

$$y_i = -\int_0^T s_i^a(t)\lambda(t)\,\mathrm{d}t \tag{5.18}$$

From (5.2) and (5.11), we have the following expression for the average output of the matched filter of the i^{th} perturbing router:

$$E[y_i] = E[\eta_{s_i^a(r_i)}]$$

This equation gives the basis for quantifying the responsiveness of TCP aggregates. Denote:

$$K_i = E[y_i]/A_i \tag{5.19}$$

Notice that K_i is a coefficient that describes how much the aggregate responds to packet drops. We call this quantity the *response coefficient* of the aggregate. Note that y_i is fully observable and can easily be measured by using (5.18). The amplitude of perturbing function A_i is known to the router that does the perturbation. Finding K_i is the only problem of the estimator. This coefficient can be estimated during the times that there is no congestion in the network. Or it can be estimated by a long term average of y_i/A_i based on multiple tests. Based on the result of Theorem 5, the estimation value of K_i is not affected by the perturbations done by the other routers, under the orthogonality assumption.

There are some key issues about how to choose the value of T_c. As stated before, T_c should be long enough such that the rate decrease of the aggregate as a result of packet drops in one chip duration can show up, and the aggregate rate settles down. On the other hand, too large T_c does not improve the performance in estimating the response coefficients, and it only causes longer test and more packet drops, which causes the test to be more expensive.

5.5 Fair Congestion Control by Using CAPM

In this section we suggest a method to use CAPM to do congestion control in a fair way. Random Early Drop [?] is one of the popular approaches to proactively prevent congestion in a router. By utilizing CAPM a router collects information about how responsive different aggregates are -i.e., K_i coefficients defined in the previous section. Knowing these coefficients helps a router to determine how much it should drop from each aggregate to reduce its bandwidth to a certain value.

Assume traffic at a router which is composed of many aggregates is intended to be forwarded through an outgoing link that has bandwidth shortage. So it is desired to keep the traffic bandwidth within the outgoing link capacity. If the router applies equal drop probability governed by a congestion controller such as a RED controller for all aggregates, the aggregates with higher response coefficients will back off more aggressively compared to the aggregates with smaller response coefficients. A certain degree of fairness among aggregates can be achieved by taking into account their response coefficients. Assume the traffic is a combination of M aggregates, and let $\lambda_i(t)$, and K_i denote the estimated instantaneous arrival rate and the response coefficient of the i^{th} aggregate respectively. Assume that we want to rate limit the total traffic, and let the output of congestion controller at time t be $p(t)$. With the information of response coefficients of aggregates the router can estimate how this total drop probability should be assigned among the aggregates to get a specific amount of rate decrease for each one.

To illustrate the above approach assume it is desired to have the same amount of rate decrease for all aggregates. Then we can assign the packet drops among different aggregates in a way that the product of the response coefficient and the drop rate is equal for all of them. In other words:

$$K_i\theta_i(t) = K_j\theta_j(t) \quad 1 \leq i, j \leq M \tag{5.20}$$

in which $\theta_i(t)$ and $\theta_j(t)$ denote the average drop rate of the i^{th} and j^{th} aggregate respectively. In the above equation subscripts are index of aggregates in the same routers. Heuristically equation (5.20) means that the rate decrease of the aggregates should be equal. It is important to note that equation (5.19) suggests using (5.20) as a heuristic to equalize the rate decreases of the aggregates; however, (5.20) is not a mathematical consequence of (5.19).

If $p_i(t)$ is the drop probability of the i^{th} aggregate, we have $\theta_i(t) = \lambda_i(t)p_i(t)$. Therefore, equation (5.20) can be written in the following way:

$$K_i\lambda_i(t)p_i(t) = K_j\lambda_j(t)p_j(t) \quad 1 \leq i, j \leq M \tag{5.21}$$

which gives $M - 1$ linear equations. To find the numerical values of the drop probabilities we need one other equation. We use the fact that the total drop probability of the traffic should be $p(t)$. In other words:

$$\sum_{i=1}^{M} \frac{p_i(t)\lambda_i(t)}{\lambda(t)} = p(t) \tag{5.22}$$

in which $\lambda(t) = \lambda_1(t) + \lambda_2(t) = \cdots + \lambda_M(t)$ is the total rate of the traffic.

In the above approach we have tried to get the same rate decrease for different aggregates, however, one can apply the response coefficients in different ways to achieve an arbitrary value of rate decrease for each aggregate. For example, it may be desired to have the same percentage of rate decrease for different aggregates; in this case it is very easy to write equations similar to equation (5.21) to find drop probabilities.

Chapter 6

Estimating Nonconforming Proportion of TCP SYN Packets and Data Packets in An Aggregate

In the previous section we explained the CAPM and its application in congestion control. In this section we modify CAPM to make it suitable for estimating the portion of nonconforming TCP SYN packets in an aggregate. This modification is for the purpose of using CAPM in detecting DDoS attacks. Our test is based on the correspondence of SYN and ACK pairs. A normal TCP connection goes through a 3-way handshake at the start of a session. A client sends a SYN packet to the server, and the server sends back a SYN/ACK packet to the client. The client sends an ACK packet to the server to complete the connection.

Our key idea to estimate the proportion of nonconforming SYNs in an aggregate is based on the observation that in a normal TCP aggregate, the average rate of SYN packets should be equal to the average rate of ACK packets that complete the handshake. Note that in addition to the ACK packets being sent as part of the 3-way handshake, there are other ACK packets being sent as acknowledgement of data packets, but we assume that we can make use of TCP sequence numbers in the TCP headers to distinguish the ACK packets being sent in response to SYN/ACK packets from the other ACK packets. Unless we state otherwise, by ACK packets we mean those ACK packets being sent as a part of the 3-way TCP handshake.

For the purpose of estimating the nonconforming portion of TCP SYNs in an aggregate, we use a signature waveform to perturb the aggregate by dropping a small portion of SYN packets, and observe the response of the aggregate to the dropped SYNs. For a SYN that is not dropped, we expect to observe an ACK in a short period afterward (about the round trip time of the connection), and for a dropped SYN we expect not to observe the ACK at least for the retransmission time of the SYN, which is usually in the order of 3 seconds, and much longer than the typical round trip time of connections which is in the order of 100 milliseconds.

Perturbations to the SYN packets are done by dropping them with a small rate $r(t) = As(t)$, in which A is the amplitude of test and $s(t)$ is the signature of perturbations. $s(t)$ is generated by a binary sequence of length N denoted by $\{c_1, c_2, ..., c_N\}$. Each binary number in this sequence represents a time interval of length T_c, known as the chip interval. If c_i is 1, we drop packets with rate A in the corresponding interval of length T_c, otherwise we do not drop any SYN packet. This scheme is equivalent to the following definition of the signature waveform:

$$s(t) = \sum_{j=1}^{N} c_j p_{T_c}(t - (j-1)T_c) \tag{6.1}$$

in which $p_{T_c}(t)$ is known as the chip function in the literature of CDMA Spread Spectrum Communication; it is 1 in interval $[0, T_c]$, and 0 otherwise.

Dropping SYN packets with rate $r(t)$ causes the arrival rate of SYN packets and ACK packets to become unbalanced during the intervals that we drop the packets. It other words, dropping SYNs will cause the number of arriving ACKs to reduce during perturbation intervals, and the average difference of the rate of arrival of SYNs and the rate of arrival of ACKs at the router will be $r(t)$. We use this observation as a key idea to distinguish the conforming SYNs from the nonconforming SYNs.

Assume $\lambda_S(t)$ represents the arrival rate of the SYNs at a router, and $\lambda_A(t)$ represents the arrival rate of ACKs. Then we define a matched filter output in the following way:

$$\alpha = \int_0^{NT_c} (\lambda_S(t) - \lambda_A(t)) s^a(t) dt \tag{6.2}$$

in which $s^a(t)$ is the AC component of $s(t)$. We define α/A as the *conformance coefficient* of the aggregate. Under normal conditions of the traffic, we expect that $\lambda_S(t) - \lambda_A(t) \simeq r(t)$, and this leads us to the following value for the matched filter output:

$$\alpha = A \int_0^{NT_c} s^a(t)^2 dt \tag{6.3}$$

We choose the binary sequence $\{c_1, c_2, ..., c_N\}$ such that it has an even number of ones and zeros, and this means that its DC component is 1/2, so $s^a(t) = s(t) - 1/2$. Therefore, $s^a(t)$ only takes the two values $-1/2$ and $1/2$. So $s^a(t)^2 = 1/4$, and

$$\int_0^{NT_c} s^a(t)^2 dt = \frac{1}{4} NT_c \tag{6.4}$$

By substituting (6.4) in (6.3) we get the following value for the output of matched filter under normal conditions of the network:

$$\alpha = \frac{1}{4} ANT_c \tag{6.5}$$

Assume m represents the total number of packet drops made by the test. So

$$\alpha = \frac{m}{2}. \tag{6.6}$$

We use the matched filter output and its deviation from the above nominal value to estimate the portion of nonconforming SYNs in the aggregate. We use the idea that SYNs belonging to a TCP SYN flood attack or nonconforming SYNs do not respond in a proper way to drops and cause deviation of matched filter output from its nominal value.

6.1 An Estimator Based on the Matched Filter Output

In this section, we study the properties of matched filter output defined in the previous section, and we define an estimator that uses the matched filter output to estimate the nonconforming component of SYN packets in an aggregate. We define a nonconforming SYN as a SYN that is not followed by an ACK, or it is followed by a blind ACK, which means an ACK is sent without waiting for the SYN/ACK from the server. Furthermore, we assume a SYN packet is nonconforming with probability p.

Assume we classify the nonconforming SYNs into two groups. The first group is the group that sends no ACK and the second group sends blind ACKs. We assume the rate of the first group is λ_c and the rate of the second group is λ_b. Furthermore we assume that the rate of legitimate SYNs is λ_l. Obviously

$$\lambda_S = \lambda_l + \lambda_b + \lambda_c \tag{6.7}$$

and

$$p = p_b + p_c \tag{6.8}$$

where p_b is the probability of a blind SYN and p_c is probability of a SYN with no ACK. Note that $p_b = \lambda_b/\lambda_S$ and $p_c = \lambda_c/\lambda_S$.

For analyzing the performance of the test, we define two sets of random variables: the first set corresponds to the SYN packets we drop. If the the total number of packet drops by the test is m, we define a sequence of Bernoulli random variables $X_1, X_2, ..., X_m$ corresponding to the m SYN packet drops. X_i is defined to be 1 if an ACK packet corresponding to the i^{th} dropped SYN is observed in the same chip interval that the packet is dropped, and it is 0 otherwise. Thus,

$$X_i = \begin{cases} 1 \text{ with probability } p_b \\ 0 \text{ with probability } 1 - p_b \end{cases} \tag{6.9}$$

The second set of random variables corresponds to the SYN packets that we observe and do not drop during the test. Assume we observe n such packets. We define a sequence of random variables $Y_1, Y_2, ..., Y_n$. The value of Y_i corresponds to the i^{th} non-dropped SYN, and it is 1 if we do not observe the corresponding ACK during the same chip interval in which we observe the SYN packet. Then

$$Y_i = \begin{cases} 1 \text{ with probability } p_c \\ 0 \text{ with probability } 1 - p_c \end{cases} \tag{6.10}$$

Lemma 3: The value of the matched filter output defined by equation (6.2) can be written as:

$$\alpha = \frac{1}{2}(m - \sum_{i=1}^{m} X_i + \sum_{i=1}^{n_1} Y_i - \sum_{i=n_1+1}^{n} Y_i) \tag{6.11}$$

in which n_1 is the number of SYN packets observed and not dropped when the perturbation signature has value 1.

Proof: We order the corresponding random variables of SYNs that we

Table 6.1: The summary of all possible cases for a SYN packet for proof of Result 1

SYN	ACK	$S(t)$	X_i	Y_i	contribution to matched filter output
Passed	Observed	1	n/a	0	$0 = Y_i/2$
Passed	Not Observed	1	n/a	1	$+1/2 = Y_i/2$
Dropped	Observed	1	1	n/a	$0 = (1 - X_i)/2$
Dropped	Not Observed	1	0	n/a	$+1/2 = (1 - X_i)/2$
Passed	Observed	0	n/a	0	$0 = -Y_i/2$
Passed	Not Observed	0	n/a	1	$-1/2 = -Y_i/2$

observe and do not drop when $s(t) = 1$ by $Y_1, Y_2, ..., Y_{n_1}$ and let $Y_{n_1+1}, Y_{n_1+2}, ..., Y_n$ represent the rest of the Y random variables. Note that during the normal conditions of the network X and Y random variables are 0, so $\alpha = m/2$, which is consistent with equation (6.6).

To prove the lemma we group SYNs into six categories. Grouping is based on three properties. First: whether the SYN was followed by an ACK, second: the value of $s(t)$, and third: whether we dropped the SYN packet for test. The summary of cases is shown in Table 6.1. Note that each SYN packet arriving during the test corresponds to exactly one row entry of this table. To complete the proof, we add the values in the last column of the table for all $m + n$ SYNs that arrive during the test.

Theorem 6: If the number of zeros and ones in the binary signature sequence $\{c_1, c_2, ..., c_N\}$ are equal, and the average of SYN arrival process $\lambda_S(t)$ is stationary during the test interval, then the following forms an unbiased estimator of $p = p_b + p_c$:

$$\hat{p} = \frac{1}{m}(\sum_{i=1}^{m} X_i - \sum_{i=1}^{n_1} Y_i + \sum_{i=n_1+1}^{n} Y_i) = 1 - 2\alpha/m \tag{6.12}$$

Proof: To prove the above result, it suffices to verify $E[\hat{p}] = p_b + p_c$. By using the fact that $X_1, X_2, ..., X_m$ is an iid sequence and the same fact holds for $Y_1, ..., Y_n$, we can write:

$$E[\hat{p}] = \frac{1}{m}(mp_b - E[n_1]p_c + E[(n - n_1)]p_c) \tag{6.13}$$

Note that in writing the above equation, we have considered the fact that m is a deterministic number, but n and n_1 are random variables whose characteristics depend on the SYN arrival process $\lambda_S(t)$. Now recall the fact that the average of $\lambda_S(t)$ is stationary during the test interval $[0, NT_c]$. Assume λ_S represents the stationary average of $\lambda_S(t)$. Therefore, we expect to observe an average total number of $NT_c\lambda_S$ SYN packets during the test interval, from which m are dropped. Hence:

$$E[n] = NT_c\lambda_S - m \tag{6.14}$$

$$E[n_1] = \frac{1}{2}NT_c\lambda_S - m \tag{6.15}$$

The last equation is a result of the fact that we have assumed the number of zeros and ones in the binary sequence $\{c_1, c_2, ..., c_N\}$ are equal, and the fact that packet drops for the test are only performed during the intervals where $s(t) = 1$. (To make this fact more clear, note that the expected total number of SYNs arriving when $s(t) = 1$ is $NT_c\lambda_S/2$). By substituting values of n and n_1 in equation (6.13) we have:

$$\begin{aligned} E[\hat{p}] &= \frac{1}{m}(mp_b - (\frac{1}{2}NT_c\lambda_S - m)p_c + \frac{1}{2}NT_c\lambda_S p_c) \\ &= p_b + p_c = p \end{aligned} \tag{6.16}$$

QED.

The preceding theorem shows that the simple function $1 - \alpha/m$ of the matched filter output can be used to estimate the nonconforming proportion of an aggregate of SYN packets.

Another important property of the estimator (6.12) is the fact that such an estimator is robust to the possible interference made by any other router that performs simultaneous perturbation. This fact is true as long as the perturbation signatures of the routers are orthogonal. In general, if an aggregate experiences perturbations of M different routers each dropping with rate $r_k(t) = A_k s_k(t)$, and the average rates of $\lambda_l(t)$, $\lambda_b(t)$ and $\lambda_c(t)$ are stationary over the test interval, then we can state the following theorem:

Theorem 7: Assume the overall perturbations applied to an aggregate by K routers, $r(t) = \sum_{k=1}^{K} A_k s_k(t)$, are small enough so that the probability of dropping both a SYN and its retransmitted SYN after the timeout is negligible. Also assume every pair of perturbing routers have orthogonal signatures. Then the matched filter output defined by equation (6.2) at each of the routers is independent of the perturbations performed by the other routers.

Proof: To avoid complexity of notation, we prove the claim of theorem for two routers ($K = 2$) and call them Router 1 and Router 2. Furthermore, we assume the aggregate passes through Router 1 and then through Router 2. Assume λ_S^i and λ_A^i denote the observed arrival rate of SYNs and ACKs at the i^{th} router, respectively.

$$\lambda_S^1(t) = \lambda_l(t) + \lambda_b(t) + \lambda_c(t) + (1 - p_b - p_c)(r_1(t - \tau) + r_2(t - \tau)) \qquad (6.17)$$

in which the last term of the above equation is due to retransmission of the SYN packets dropped by Routers 1 and 2, and we have assumed that the retransmission of a SYN happens in about τ seconds after it is dropped, which is 3 seconds in average if we neglect the probability of multiple drops of retransmissions of the same SYN. Similarly, we can write the following for the rate of ACKs in Router 1:

$$\begin{aligned}\lambda_A^1(t) = {} & \lambda_l(t) + \lambda_b(t) - (1 - p_b - p_c)(r_1(t) + r_2(t)) \\ & + (1 - p_b - p_c)(r_1(t - \tau) + r_2(t - \tau)) \qquad (6.18)\end{aligned}$$

Note that dropping SYNs with rate $r_1(t)$ makes only the conforming SYNs stop sending ACKs, and blind ACKs are not affected by perturbations. Also, from $r_1(t) + r_2(t)$ SYN drops per second made by the two routers, the rate of drops received by conforming SYNs is $(1 - p_c - p_b)(r_1(t) + r_2(t))$, and this generates the third term of the above equation. The difference of the rates of

SYNs and ACKs at the first router can be written as:

$$\lambda_S^1(t) - \lambda_A^1(t) = (1 - p_c - p_b)(r_1(t) + r_2(t)) + \lambda_c(t) \tag{6.19}$$

This leads us the the following value for the matched filter output of Router 1:

$$\begin{aligned} E[\alpha_1] = (1 - p_c - p_b) \int_0^{NT_c} & (r_1(t) + r_2(t)) s_1^a(t) dt \\ + \int_0^{NT_c} & E[\lambda_c(t)] s_1^a(t) dt \end{aligned} \tag{6.20}$$

To continue, note that $\int_0^{nT_c} r_1(t)s_1^a(t) = NA_1T_c/4$, and $\int_0^{nT_c} r_2(t)s_1^a(t) = 0$. Furthermore, by using the assumption that the average of $\lambda_c(t)$ is constant over the test interval $[0, NT_c]$, it follows that the second term of the above equation is zero since $\int_0^{NT_c} s_1^a(t)dt = 0$. This gives the following value for the output of the matched filter at Router 1:

$$E[\alpha_1] = (1 - p_c - p_b)NA_1T_c/4 = (1 - p_c - p_b)m_1/2 \tag{6.21}$$

in which m_1 represents the total number of packet drops made by Router 1 during the test. The above value of the matched filter output is the same as the value of the matched filter output in the case Router 2 had not perturbed the traffic.

To complete the proof, we need to show that the matched filter output at Router 2 is not affected by the test at Router 1. Similar to before, we can write the following for the rate of SYN packets at Router 2:

$$\begin{aligned}\lambda_S^2(t) &= \lambda_l(t) + \lambda_b(t) + \lambda_c(t) - r_1(t) \\ &+(1 - p_c - p_b)(r_1(t-\tau) + r_2(t-\tau))\end{aligned} \tag{6.22}$$

An important observation about the blind ACKs that Router 2 observes is that if Router 1 drops a SYN which is followed by a blind ACK, then Router 2 does not count the blind ACK of that SYN, and as a result, the ACK does not contribute to the rate of ACKs measured by Router 2. Therefore, we have the following rate of ACKs at Router 2:

$$\begin{aligned}\lambda_A^2(t) &= \lambda_l(t) - (1 - p_c - p_b)r_2(t) + \lambda_b(t) - p_b r_1(t) \\ &+(1 - p_c - p_b)(r_1(t-\tau) + r_2(t-\tau))\end{aligned} \tag{6.23}$$

Hence, the total difference of SYNs and ACKs at Router 2 will be:

$$\lambda_S^2(t) - \lambda_A^2(t) = (1 - p_c - p_b)r_2(t) + \lambda_c(t) + (p_b - 1)r_1(t) \tag{6.24}$$

By using the orthogonality assumption of the signatures and the fact that the average of $\lambda_c(t)$ is constant, we get the following expected value for the output of the matched filter at Router 2:

$$E[\alpha_2] = (1 - p_c - p_b)m_2/2 \tag{6.25}$$

in which, $m_2 = NA_2T_c/2$ is the total number of packet drops made during the test by Router 2. Equation (6.25) shows perturbation of Router 1 has not affected the matched filter output of Router 2. **QED.**

The significance of Theorem 7 is that the perturbation tests can be performed in a distributed manner by different routers without being affected by the possible interference resulting from different simultaneous tests.

We need to make an important note about the orthogonality assumption of the signatures. For this property to hold, the routers need to synchronize the starting times of their tests, which may be impractical. The same problem holds in the spread spectrum CDMA communication. Our suggestion to solve this problem is to make use of the offered solution in communication theory. There are codes (such as pseudo random binary codes), that are close to orthogonal to each other if the length of code is chosen to be long enough; codes generated by this scheme are approximately orthogonal even with variable offsets.

6.2 Extension of Approach to Data Packets

In this section, we extend the conformance test of TCP SYN packets for all packets that use retransmission schemes in response to packet losses. Note that this includes the TCP data packets as well as certain non-TCP packets such as DNS queries. We make the following assumption about the packets for which we apply the test:
A1: Dropped packets are retransmitted within a finite timeout period.
A2: There is an identifier field that can be utilized to identify the retransmission of a given packet.
Note that Assumption **A2** is true for many protocols. For TCP packets, we

make use of sequence numbers, which are 32-bit randomly generated numbers at the start of each connection and incremented based on progress in transmission of data octets for subsequent packets; for DNS queries we can use the 32-bit field known as transaction ID for the purpose of our tests.

To continue the generalization of this scheme, we introduce a method for detection of packet retransmissions on a link during the test interval.

6.2.1 Detecting Packet Retransmissions

To detect packet retransmissions we make use of a short term memory in which we can set a flag for every packet we observe. For every incoming packet on a link, we extract the packet ID, and the packet ID is used as the index to memory. The location in the memory that the ID is pointing is set upon receiving each packet, and that location is reset when we observe retransmission of that packet.

The above book-keeping scheme may need a huge memory if we want to consider a location for every packet ID. For example, if the packet ID is the 32-bit sequence number of TCP packets, we need $2^{32} = 4Gbits = 500MB$ of memory which may be impractical or expensive in many applications. Instead we suggest using a shorter hash of packet IDs to solve this problem. However making the hash too small increases the probability of false hits (collisions) by mapping too many sequence numbers to the same location in the memory.

In general, the worst case probability of a false hit for each packet can be written as:

$$p_{fh} = 1 - (1 - \frac{1}{L})^{\frac{TW}{B}} \tag{6.26}$$

in which, p_{fh} denotes the probability of a false hit, which is the probability that a new packet is wrongly identified as a retransmission of a previously seen packet during the test interval. L is the memory size in bits, T is the total duration of the test, W is the link speed in bits/sec, and B is the average packet size in bits. In order to have a small false hit rate, we need $\frac{1}{L} << \frac{TW}{B}$, and if this happens, we have:

$$p_{fh} \approx \frac{TW}{LB} \tag{6.27}$$

The above equation gives the minimum required memory size for a low false hit probability. For example, if $T = 4Sec$, $W = 1Gbps$, $B = 1KB$, and L=$32MB$, then the probability of a false hit is about 0.002.

6.2.2 Using Retransmissions for Detecting Nonconforming Component of Traffic

In this subsection we present a method that gives the nonconforming proportion of data packets in an aggregate. We assume that the rate of transmitted packets is $\lambda_D(t)$, of which $\lambda_R(t)$ are retransmitted packets. If we drop the arriving packets with perturbing function:

$$r(t) = A \sum_{j=1}^{N} c_j p_{T_c}(t - (j-1)T_c) = As(t) \tag{6.28}$$

then we expect that $\lambda_R(t)$ to show the shape of function $r(t)$. Similar to before, we use the deviation of $r(t)$ from the above nominal response to estimate proportion of nonconforming packets, that includes packets with no retransmission upon drops, or those that blindly retransmit without drop. Now if we apply the following transformation:

$$\begin{aligned}\lambda_S(t) &= \lambda_D(t) \\ \lambda_A(t) &= \lambda_D(t) - \lambda_R(t)\end{aligned} \tag{6.29}$$

then this case becomes mathematically equivalent to the case of SYN and ACK packets, and the results we obtained for the TCP SYN packets automatically extend to this case. Similar to before we define the following value as the matched filter output:

$$\alpha = \int_0^{NT_c} \lambda_R(t) s^a(t) dt \tag{6.30}$$

It is straightforward to verify that the nominal value for the matched filter output in this case is:

$$\alpha = \frac{m}{2} \tag{6.31}$$

in which m is the total number of packets dropped during the test interval. Deviation of the observed matched filter output from the above value gives an estimator for finding the fraction of nonconforming packets:

$$\hat{p} = 1 - 2\alpha/m \tag{6.32}$$

In the same way as before, it can be shown that the above estimator is unbiased.

6.3 Notes on Practical Issues

In this section we discuss the practical issues of the proposed methods offered in the previous sections.

6.3.1 Independence from Routing

The first advantage of modified CAPM that we offered in the previous sections is the fact that it does not depend on the assumption that the paths between a DDoS source and the victim are symmetric. Note that both the TCP SYN packets and the ACK packets being sent from a TCP client to a server travel on the same path. This is also true for the TCP data packets and their retransmissions.

6.3.2 Blind Transmissions of ACKs

The other issue that needs some further discussion is the fact that CAPM can easily take into consideration the effect of blind transmissions of ACKs. In other words, if a DDoS attack is launched against a server or a network, the sources can send an ACK packet corresponding to every SYN packet they send to the victim to mislead the intermediate routers that perform the CAPM test. However, using signature based tests avoids possible false negatives generated by such blind ACKs. As our mathematical results show, the modified

CAPM detects both SYN packets which are blindly followed by ACKs and those that are not followed by ACK packets. The mathematical explanation of this property of modified CAPM is that sending blind ACKs causes a DC shift in $\lambda_S(t) - \lambda_A(t)$, and it does not make any change in the component of the signature appearing on this signal in response to perturbations.

We use a simple example to illustrate the above fact. Assume a router receives 10000 SYN packets per second, and it uses a perturbation signature generated by $\{1, 0, 0, 1\}$, $N = 4$, $A = 100$, and $T_c = 1$ second to screen them. The router drops 1 percent of arriving SYN packets when $s(t) = 1$. We consider the following three cases:

Case 1: In the first case there is no nonconforming SYN in the above aggregate, and both the SYN rate and the ACK rate are almost 10000 packets per second when there is no perturbation; however, when we apply the perturbation, the ACK rate decreases to $\lambda_A(t) = 9900$ packets per second. In this case the nominal value of the matched filter output is $\alpha = 100$.

Case 2: In this case assume that 40 percent of generated SYNs are not followed up by an ACK. This causes the rate of ACK packets to be 6000 when we do not perturb the traffic, and the rate reduces to 5940 packets per second when we randomly drop 100 packets per second. This is because from the 100 per second dropped SYN packets for perturbation, only 60 packets per second belong to the conforming SYNs, and the other 40 packets per second are not followed by ACK packets. It can be seen that the matched filter output in this case reduces to 60, which shows 40 percent decrease compared to the normal case.

Case 3: In the last case assume that 40 percent of SYNs are followed up by blindly retransmitted ACKs. The other 60 percent is composed of legitimate SYNs. In this case if we do not drop any SYN, we have 10000 ACKs per second, and in the intervals that we drop SYNs, we have 9940 ACKs per second. This is because from 100 per second packet drops made for the test, 40 of them belong to the group that send an ACK regardless of the server response. In this case, the output of matched filter is 60 again, and similar to the previous case, the matched filter output estimates 40 percent nonconforming traffic.

The above examples show that blind transmissions of ACKs cannot mislead the modified CAPM in detecting nonconforming SYN packets. The summary of cases is shown in Figure 6.1. As can be seen in this figure, in cases (b) and (c) the shape of the signature appears on the signal $\lambda_S(t) - \lambda_A(t)$ with strength 60, whereas in the normal condition it appears with strength 100.

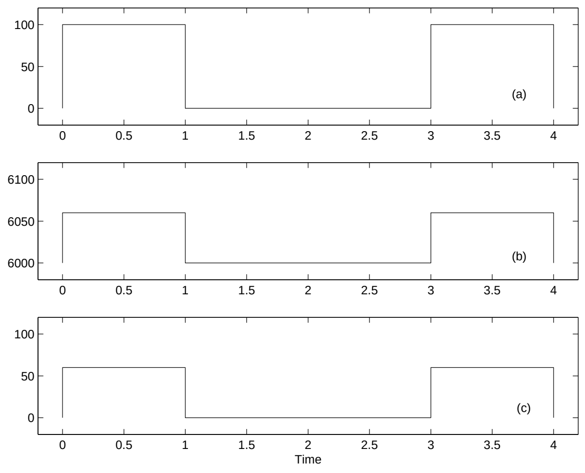

Figure 6.1: Difference of rate of SYNs and ACKs, $\lambda_S(t) - \lambda_A(t)$, for three cases: (a) Normal Conditions, (b) 40 percent of SYNs do not send ACK, and (c): 40 percent of SYNs send blind ACKs.

Chapter 7

Simulation Experiments for CAPM

In this chapter we present results of simulations that confirm the efficacy of the methods introduced in previous chapters. We present the results in three sets of experiments. In the first set, we evaluate the fair congestion control scheme that uses CAPM. In the second set, we explore performance of tests that perturb the rate of SYN packets and examine the rate of ACK packets for the purpose of conformance test. In the third set we study performance of the method that perturbs data packets and observes their retransmissions for conformance purposes. We have used *ns2* [83] network simulator for the experiments.

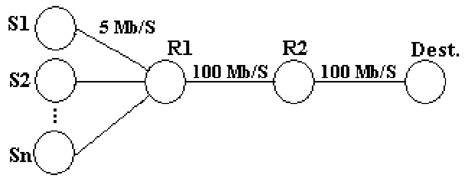

Figure 7.1: The network topology used for simulation

7.1 Using CAPM for Fair Congestion Control

For this set of simulations we have used a network with fixed topology as in Figure 7.1. The nodes $S_1, S_2, \ldots, S_n$ are n distributed sources of TCP traffic. The traffic of the sources pass through Router 1 (R_1) and Router 2 (R_2) to the destination. The propagation delay of the link between each source and Router 1 in Figure 7.1 is different from a source to another source, and it has been chosen such that the round trip time of packets is uniformly distributed between 50 and 100 milliseconds under low congestion conditions. The flows at the sources are generated according to an on-off process. Each source starts a TCP flow, and that flow ends after a random time uniformly distributed between 0 and 0.15 seconds. That source starts a new flow after waiting another random time uniformly distributed between 0 and 0.3 seconds. The packet size is constant equal to 1 Kbyte for all flows. In this topology, the link between Router 1 and Router 2, and also the link between Router 2 and destination are bottleneck links. The capacity of these bottleneck links is 100 Mbps, that translates to 12500 packets per second.

The Basic Experiment: In the first experiment we show how an aggregate responds to the signature based perturbations. For this experiment we run the simulation for two cases. In the first case the aggregate does not experience any perturbation; in the second case only Router 1 perturbs the aggregate by using drop rate $r_1(t) = A_1 s_1(t)$, in which $s_1(t)$ is the normalized drop signature generated by substituting the code $(1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1)$ in equation (5.3), and $A_1 = 160$ packet drops/sec. In the simulation, the number of sources is 50, $T = 32$ seconds, $N = 16$, $T_c = 2$ seconds. Figure 7.2-(a) shows the rate of the aggregate when no perturbation is performed. In Figure 7.2-(b) the rate of aggregate is shown when Router 1 perturbs the aggregate by using $r_1(t)$, and Figure 7.2-(c) shows two periods of the normalized drop signature $s_1(t)$. By inspecting Figure 7.2-(b) we can see that the shape of the drop signature of Router 1 has appeared in the rate of the aggregate – with 180 degrees of phase shift. In other words, when $s_1(t) = 1$ (e.g., around $t = 14$), the rate of aggregate decreases, and when $s_1(t) = 0$ (e.g., around t=10), the rate increases.

Experiment with Simultaneous CAPM Test of Two Routers: In the second experiment we explore the typical response of aggregate when two routers perturb it simultaneously. In this experiment Router 1 and Router 2 perturb the aggregate by using different CDMA drop signatures. In the simulation, the number of sources is 50, $T = 32$ seconds, $N = 16$, $T_c = 2$ seconds. The code of Router 1 is $(1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1)$, and that for Router 2 is
$(0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1)$. Under this assignment $s_1^a(t)$ and $s_2^a(t)$ are orthogonal. Two periods of the resulting normalized drop signatures for Router 1 and Router 2 are shown in Figure 7.3-(b) and 7.3-(c) respectively. The amplitude of drop signatures for the two routers, A_1 and A_2, are the same and equal to 120 drops per second. Figure 7.3-(a) shows the rate of aggregate

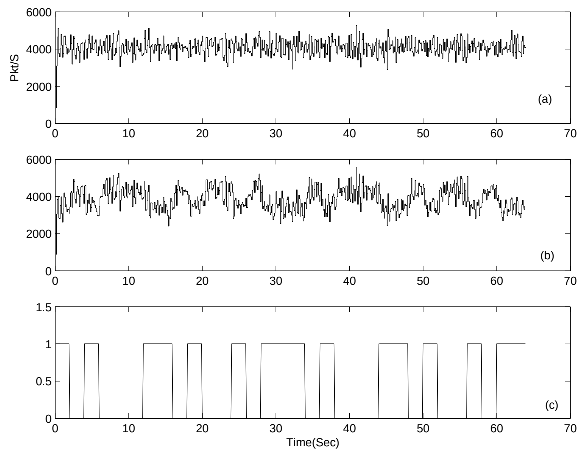

Figure 7.2: a: The aggregate rate without any perturbation, b: The aggregate rate with perturbation of Router 1, and c: the normalized drop signature of Router 1.

when Router 1 and Router 2 perturb the aggregate simultaneously. It can be seen that the additive shape of the two drop signatures appears on top of the aggregate rate– with 180 degree phase shift again. In other words, the two drop signatures modulate the aggregate rate additively. For example, at around time $t = 40$, the amplitude of both drop signatures is zero, and this shows up as an increase in the rate of aggregate as can be seen in 7.3-(a) at $t = 40$. On the other hand, at time $t = 15$ or $t = 31$, the amplitude of both drop signatures is nonzero, and this shows up as a decrease in the rate at these two times.

Robustness To Interference: The purpose of the next experiment is to verify that under orthogonality definition of (5.8), the matched filter output of a router defined by (5.18) is not affected by perturbations done by the other routers. We proved this fact in Theorem 5. In this case we use the same CDMA drop signatures as in the previous experiment, but we change A_1 and

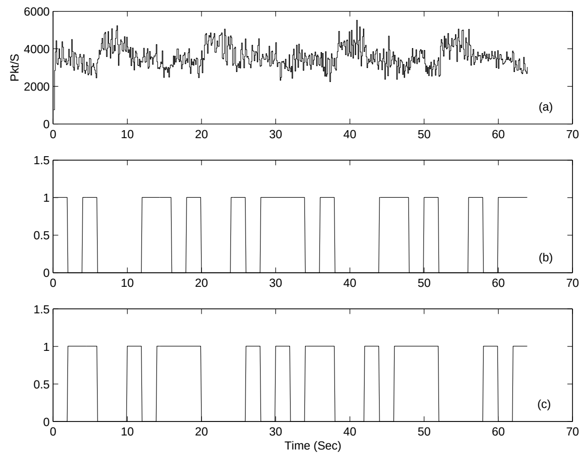

Figure 7.3: a: The aggregate rate under two simultaneous perturbations, b: the normalized drop signature of R1, and c: the normalized drop signature of R2.

A_2, the amplitude of the drop signatures of the two routers. Figure 7.4-(a) shows y_1, the output of matched filter for Router 1 as it is defined by equation (5.18), when A_1 changes from 0 to 160 drops per second. In this figure, each + represents a test in which Router 1 perturbs the aggregate with drop signature $r_1(t) = A_1 s_1(t)$ and at the same time Router 2 is also perturbing traffic with drop signature $r_2 = A_2 s_2(t)$, and $A_1 = A_2$. For each value of A_1 several tests have been done, and the average over multiple tests has been plotted by the solid line. It can be seen that the deviation of y_1 for each individual test from the average value shown by solid line is relatively small; this means that the matched filter output shows a small variance. The other observation about 7.4-(a) is linearity in amplitude of drop signature A_1.

In the other part of this experiment we turn off the perturbations done by Router 2 by setting $A_2 = 0$, and do the same multiple test and measurement of y_1 for each value of A_1. The dashed line in Figure 7.4-(a) shows the average of multiple tests for each value of A_1 for this case. It can be seen that the dashed line is very close to the solid line showing that perturbations of Router 2 do not affect the output of matched filter of Router 1. Figure 7.4-(b) is the same as Figure 7.4-(a) for the second router.

Using Response Coefficients for Congestion Control: In the next experiment we show how the response coefficients can be used to do congestion control in a fair way. So we define two aggregates that pass through Router 1. In this experiment the sources in the simulation network are divided into two groups. The on time of a flow generated by a source in group 1 is uniformly distributed in $[0.15, 0.3]$ seconds, and after ending a flow, the source starts another flow after being idle for a random time uniformly distributed in $[0, 0.3]$. There are 50 sources in group 1. Sources in group 2 generate larger flows. The on time of a flow in group 2 is uniformly distributed in $[0.45, 0.9]$ seconds, and the idle time between flows is uniformly distributed in $[0, 0.5]$ seconds. There are 20 sources in group 2. We define the traffic generated by group 1 as aggregate 1 and traffic generated by group 2 as aggregate 2.

First we find the response coefficient of each of the two aggregates by using equation (5.19). The experiment shows that $K_1 = 77.3$, and $K_2 = 588.1$. The value of response coefficients have been found by several tests and averaging the results. The higher value of the response coefficient of aggregate 2 is easy to explain by considering the fact that the flows belonging to this aggregate are larger, so they show a higher rate decrease when they experience packet drops. The experiment shows that λ_1, the average rate of aggregate 1, is 5234 pkt/sec, and λ_2, the average rate of aggregate 2 is 4547 pkt/sec. The total rate of the traffic is 9781 pkt/sec.

In the next step of this experiment, we doubled the number of sources in each group, so aggregate 1 is about $2 \times 5234 = 10468$ pkt/sec; aggregate 2 is 9094 pkt/sec, and the total demand is 19562 pkt/sec. This total demand is more than the link capacity which is 12500 packet/sec. The simulation results show that under drop tail condition in the forwarding queue of Router 1, about 4% of incoming packets are dropped, and as a result of it the arrival rate of the traffic is reduced to about 12570 packets/sec. Under this drop policy the rate of aggregate 1 reduces to 8012 pkt/sec and the rate of aggregate 2 reduces to 4558 pkt/sec. The above data means that the rate reduction of aggregate 1 is 2456 pkt/sec or 25% of its demand, while the rate decrease of aggregate 2 is 4536 pkt/sec that is 52% of its demand. Aggregate 2 shows much higher rate decrease as a result of having a higher response coefficient.

To keep the fairness in rate reduction between the two aggregates, we use the fairness scheme explained in Section **??** to assign drop probabilities. By using equations (5.21) and (5.22) we find the drop probability $p_1 = 0.067$ for aggregate 1, and $p_2 = 0.010$ for aggregate 2, so the total drop rate of the traffic is still 4%. With these drop probabilities, the rate of aggregate 1 reduces to 6522 pkt/s, and rate of aggregate 2 reduces to 5523 pkt/sec. In this case the rate reduction of aggregate 1 is 3946 pkt/sec or 39% of its demand, and the rate reduction of aggregate 2 is 3571 pkt/sec or 42% of its demand. It is seen that the rate reductions are much closer to each other than the previous experiment, and we were able to do a fair congestion control.

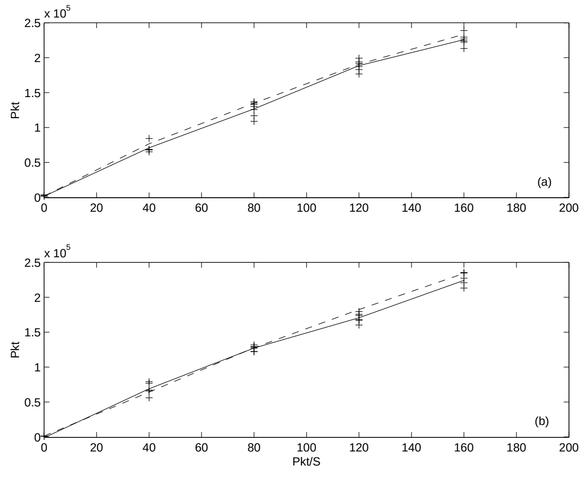

Figure 7.4: Matched filter output versus the amplitude of drop signature. Solid lines show the results of Router 1 when Router 2 performs the test simultaneously, and dashed line shows the results when Router 2 does not perform the test. b: The corresponding data for Router 2

Effect of Congestion on Tests: In the last experiment we study how congestion can affect the measurement of response coefficients. For this purpose we used an aggregate like aggregate 1 with the same conditions that were stated in the previous experiment. The response coefficient of this aggregate was measured in independent simulation runs with different link utilizations of the bottleneck links. To increase the link utilization we increased the number of sending sources in group 1. The results have been shown in Figure 7.5. In this figure each + shows the response coefficient versus the link utilization. The response coefficient shows an almost flat behavior with reasonable variance up to the point where the link utilization is about 90%. After that the measurement of the response coefficient is not accurate, however, the measurements are still good approximations up to the point where the link utilization is about 95%. In this figure the solid line shows the average of the response

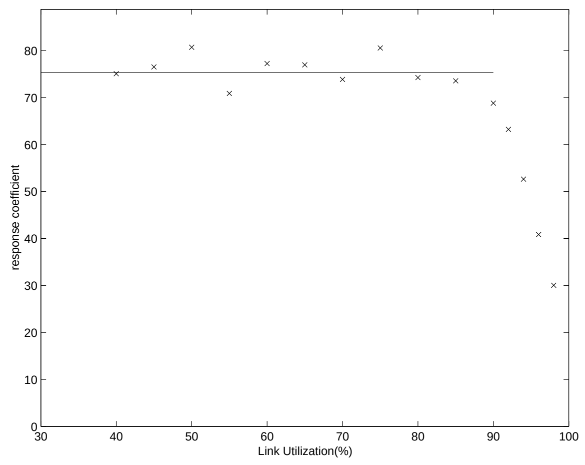

Figure 7.5: The value of response coefficient of aggregate 1 measured at different link utilizations

coefficient over the experiments for which the link utilization is less than 90%; this average is about 75.

The degradation in performance of CAPM in the presence of severe congestion is easy to explain; heavy congestion causes the aggregates to experience high rate of drops and as a result of that the aggregates shrink their rates. This causes them to become less responsive to the packet drops made by APM or CAPM. Although long term congestion is one factor that may degrade the performance of APM or CAPM, the APM and CAPM show a good performance in a wide range of link utilization before very heavy congestion happens. This can be one of the strong points about APM and CAPM since these methods can be applied proactively to prevent congestion.

7.2 Experiments with SYN and ACK Rates

In this section, we evaluate the approach we offered to estimate the proportion of nonconforming SYNs in an aggregate. In this approach we perturb the traffic by dropping SYN packets with a small probability and examine the reaction of rate of incoming ACK packets.

The Base Experiment: In the first simulation scenario, we generated different TCP connections to a server from 2500 distributed sources. Each source generated one TCP connection per second and the average number of data packets exchanged per connection was 10. Additionally, the round trip time of flows was randomly distributed in $[0, 100]$ milliseconds. We used a perturbation signature that corresponds to the binary sequence $\{1, 0, 0, 0, 1, 1, 0, 1\}$; $N = 8$, $T_c = 1$ second, and SYN packets were dropped with probability 0.01 at the intervals where the signature value was 1. Since the signature is 1 only 50 percent of the time, this introduces 0.5 percent drop probability in average. Although the drop probability may be chosen to be smaller in practice, we have deliberately chosen a larger value for it to illustrate the response of the traffic.

Figure 7.6 shows the SYN rate $\lambda_S(t)$ and the ACK rate $\lambda_A(t)$ in the experiment, and Figure 7.7 shows their difference under perturbations. It can be seen in the top plot of Figure 7.7, the difference follows the pattern of the perturbation function, and during the intervals where the SYN packets were dropped, the difference of the two arrival rates tends to be a higher value. For visualization of the data, the number of arrived SYNs and ACKs were

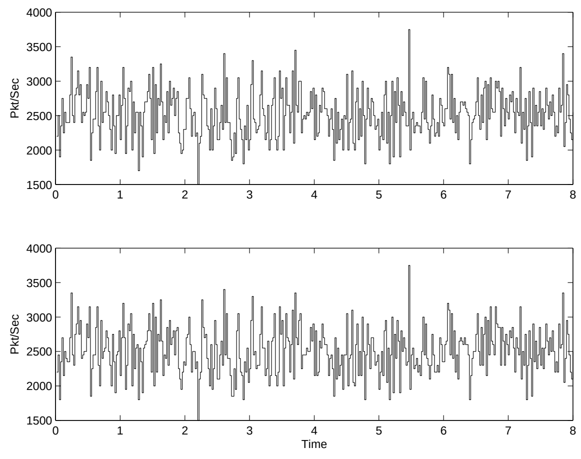

Figure 7.6: SYN rate (top plot) and ACK rate (bottom plot) in the base experiment

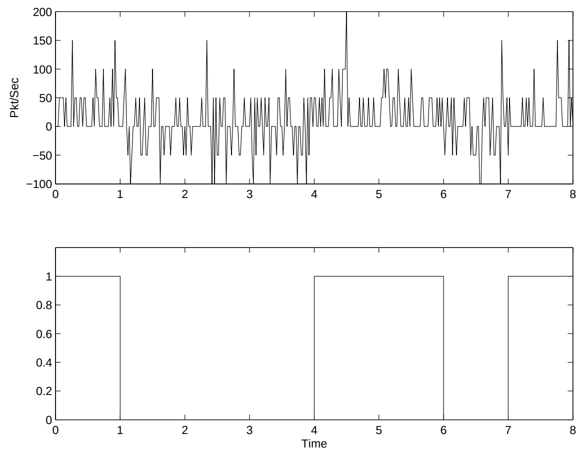

Figure 7.7: Top: difference of rate of SYNs and ACKs, $\lambda_S(t) - \lambda_A(t)$, under perturbation. Bottom: Normalized perturbation signature.

counted over intervals of length 0.02, and the measured quantities and their difference over each interval were divided by 0.02 to show instantaneous rate corresponding to that interval. For this reason, plots in Figure 7.6 and Figure 7.7 have quantization of 50.

Adding Nonconforming Traffic: In the next experiment we add nonconforming SYN packets to the aggregate and use the estimator given by equation (6.12) to estimate the proportion of nonconforming SYNs. In this scenario, the SYN packets were generated by the sources at the average rate of 2500 per second, and the rate of nonconforming SYN packets was 0, 1000, and 2000 packets per second respectively. The results are shown in Table 7.1. Each row of this table shows a single experiment. The first column of this table shows rate of legitimate SYNs, the second column shows rate of nonconforming SYNs; breakdown of the nonconforming SYNs is shown in the next two columns. The third column shows the rate of SYNs that send no ACK, and the fourth column of the table shows the rate of SYNs that send ACKs blindly. Fifth column of Table 7.1 shows the actual ratio of nonconforming SYNs to total SYNs, and the last column shows the result obtained by calculating the matched filter output and using estimator given by equation (6.12). As can be seen, the estimator shows good performance, and its deviation from the actual value in most of the cases is less than 2 percent.

7.3 Experiments with Data Packets

In this section we show the effectiveness of the method in which we drop packets from an aggregate and examine the signal that represents the rate of

Table 7.1: Evaluation of estimator of nonconforming SYN packets for different types and proportions of nonconforming SYN packets

Legitimate SYN Rate	Nonconforming SYN Rate	No-ACK SYN Rate	Blind ACK SYN Rate	Actual (%)	Estimate (%)
2500	0	0	0	0	-0.26
2500	1000	1000	0	28.6	27.3
2500	1000	500	500	28.6	28.9
2500	1000	0	1000	28.6	27.6
2500	2000	2000	0	44.4	43.8
2500	2000	1000	1000	44.4	45.8
2500	2000	0	2000	44.4	44.9

retransmitted packets in order to estimate the proportion of nonconforming traffic.

The Base Experiment: Similar to the previous set of experiments, different TCP connections were established from 2500 distributed sources to a server. Each source generated one TCP connection per second and the average number of packets generated per connection was set to be 20. Round trip times of connections in the experiments were picked randomly in the interval of $[0, 100]$ milliseconds. Generated traffic was passed through two different routers, and each router performed its tests independently on the aggregate. For generating drop signatures, the binary sequence of $1, 1, 0, 0, 1, 1, 0, 0$ was assigned to the first router and the sequence $0, 0, 1, 0, 1, 1, 1, 0$ was assigned to the second router. It can be verified that the signatures generated by these two codes are orthogonal. Similar to before, $N = 8$, and $T_c = 1$.

The top plot of Figure 7.8 shows the total packet rate, and the bottom plot of this figure shows the normalized drop signature of Router 1. During the test period, 1 percent of packets were dropped at the time intervals when the drop signature was 1. The middle plot of Figure 7.8 shows the detected retransmissions of data packets, and as can be seen, this signal obviously shows the shape of the drop signature on it.

Performing Simultaneous Tests by Two Routers: In the next experiment we let the second router perform the test at the same time as the first router. The typical value of the retransmissions at the first router when the second router performs the test is shown in Figure 7.9. The perturbation signatures of the two routers are shown in the middle and the bottom plots of this figure, and the overall retransmissions detected by the first router is shown in the top plot. As can be seen in this case, the detected retransmission rate shows additive form of the two signatures. For example in intervals such as $[3, 4]$ when none of the routers perturb, the detected rate of retransmissions is about zero. However, in intervals such as $[1, 2]$ when only one of the routers perturbs the traffic, the rate of retransmissions is about 500 per second. Finally, in intervals such as $[5, 6]$ when both routers perturb, the retransmission rate increases to about 1000 per second.

Adding Nonconforming Traffic: In the next experiment, we add nonconforming data packets to the aggregate and use the estimator given by equation (6.32) to estimate the proportion of nonconforming data packets. Nonconforming data was added from 0 to 60 percent in 10 percent steps, and on each step five different experiments were performed and the proportion of nonconforming traffic was estimated in each case. Figure 7.10 shows the results obtained by Router 1 when Router 2 is not performing any test. Each asterisk in this figure shows the result obtained by one experiment and the solid line shows the actual proportion of nonconforming data packets. The experiment

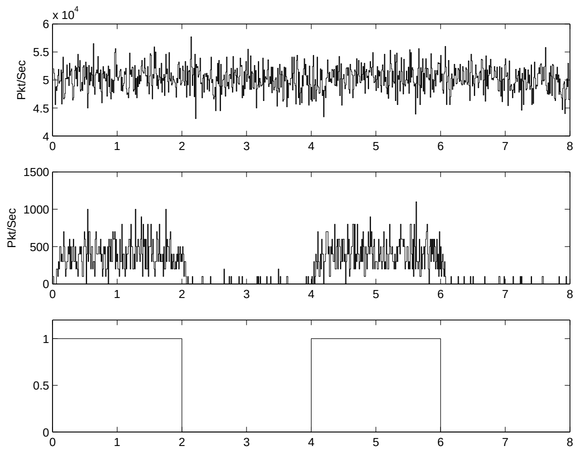

Figure 7.8: Top: Total packet rate $\lambda(t)$. Middle: The detected retransmission rate when perturbing the aggregate. Bottom: the drop signature.

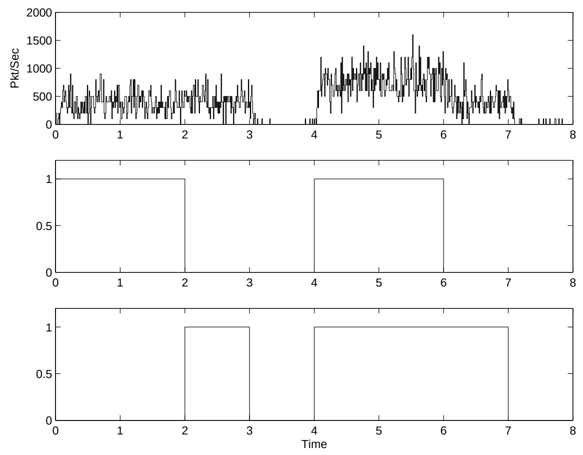

Figure 7.9: Top: the detected retransmission rate at Router 1. Middle: Signature of Router 1. Bottom: Signature of Router 2.

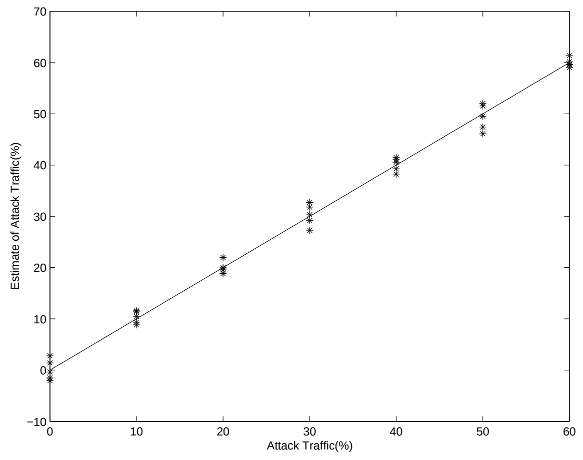

Figure 7.10: The estimate of nonconforming traffic by Router 1 when Router 2 does not perform the test. Solid line: actual percentage of nonconforming data packets. (*) Estimated percentage of nonconforming data packets.

shows that the variance of estimator in this case is 1.56 percent. The small value of variance has resulted in small deviation of estimated values from the actual values.

Results for Two Perturbing Routers: In the next experiment we examine performance of the estimator when the two routers perform tests simultaneously. The results of the tests for different proportions of nonconforming packets are shown in Figure 7.11. Similar to before, the nonconforming data was added from 0 percent to 60 percent in 10 percent steps; at each step five different experiments were performed, and for each experiment the proportion of nonconforming traffic was estimated by each of the routers independently. The top plot of Figure 7.11 shows the estimates of Router 1 and the bottom plot shows the corresponding values obtained by Router 2. The experiment shows that the variance of estimator in this case is 1.96 percent for

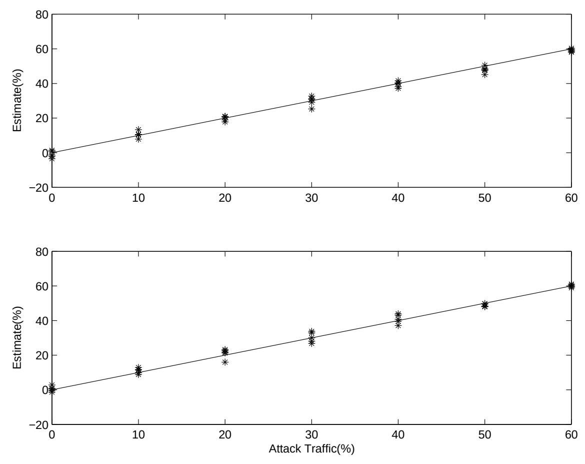

Figure 7.11: Top: estimate of nonconforming traffic by Router 1 when Router 2 performs the test simultaneously. Solid line: the actual percentage of non-conforming data packets. Each (*) shows the estimated percentage of non-conforming data packets performed by Router 1. Bottom: The corresponding plot for Router 2.

Router 1 and 2.05 percent for Router 2. Again, the small value of variances has resulted in small deviation of estimated values from the actual values for both router. Another important result of this experiment is that the simultaneous tests of the two routers has not affected the quality of the estimators of either of them.

Experiments with More than Two Routers In this experiment, we study performance of the estimator when more than two routers perform tests and estimations simultaneously. The traffic is similar to the scenario explained in the previous experiments, but the number of perturbing routers in the path between the sources and the server is varied to be 1, 2, 4 and 8. We increased the code length to 32 in order to have enough orthogonal codes to be assigned to the different perturbing routers. $T_c = 1$ second, and each router drops packets with probability 0.005 during the intervals where its signature value is equal to 1.

The results of these experiments are shown in Table 7.2. Each row of this table shows one experiment. The first column of this table shows number of simultaneously perturbing routers, and its second column shows the proportion of nonconforming traffic in the aggregate. The third column shows the average of estimated values of nonconforming traffic for all routers, the fourth column shows the worst error of the estimator among all the perturbing routers. As the data in the table shows, increasing the number of simultaneously perturbing routers does not affect the quality of estimator, and the largest error of the estimator when 8 routers were simultaneously running the tests was 4.52 percent, which is a reasonable performance. It is important to note that the interference conditions of this experiment are worst case since in practical situations it is unlikely that 8 routers perform a test on the same aggregate simultaneously.

Another interesting result of this experiment is the average increase in data transfer time of connections. We define the data transfer time as the elapsed time between the moment that the SYN packet of the connection is generated and the moment that the last data packet of the connection is received. We measured the average amount of this value among all connections during the test when there was no perturbation and this value was about 265 milliseconds. The last column of Table 7.2 shows the amount of increase in data transfer time in milliseconds compared to the case with no perturbation. It can be seen that in all cases the increase is in order of a few milliseconds, and it is negligible compared to the average normal data transfer time of the

Table 7.2: Results for different numbers of simultaneously perturbing routers.

Number of Perturbing Routers	Proportion of Nonconforming Traffic (%)	Average of Estimates of Routers (%)	Worst Case Error Among All Routers (%)	Avg. Inc. in Data Transfer Time (msec)
1	0	0.82	0.82	1.9
1	30	30.98	1.42	1.7
1	60	58.81	-1.57	0.9
2	0	1.15	1.65	2.6
2	30	31.86	2.38	2.1
2	60	58.88	-1.83	1.7
4	0	2.31	3.48	3.7
4	30	28.47	-3.92	3.0
4	60	58.87	-2.89	2.6
8	0	1.37	4.52	6.1
8	30	30.55	3.82	4.8
8	60	59.80	4.02	3.9

connections.

Effect of Congestion In the last experiment we study the effect of congestion on the performance of the estimator. Similar to before we generated conforming TCP connections from 2500 distributed sources to the server. Each source generated one TCP connection per second and the average number of packets generated per connection was set to 20. The Maximum Segment Size (MSS) of each packet was set to 1500 Bytes, and round trip times of connections were picked randomly in the interval of $[0, 100]$ milliseconds. The capacity of the link between the the victim and its first hop router was limited to be 1.0 Gbps (duplex link). Simulations show that under the above conditions, slightly less than 70 percent of the link capacity is used for the conforming traffic. We added nonconforming traffic to the aggregate in 10Mbps steps during different simulation runs. The size of forwarding buffer of the router next to victim was set to 1 MB which makes it suitable for storing up to 667 packets with the size of MSS. The results are shown in Fig. 7.12. The solid line in this figure shows the actual proportion of nonconforming traffic

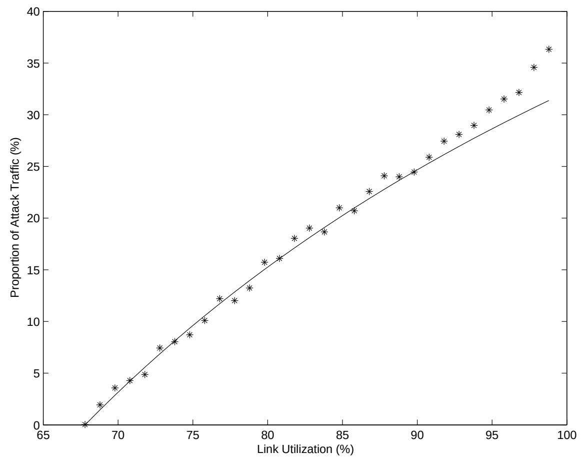

Figure 7.12: Evaluating performance of the estimator in presence of congestion. Link utilization changed from 70 percent to close to 100 percent by adding nonconforming traffic. Solid line shows the actual proportions of nonconforming traffic, and asterisks show the estimated proportions.

and each asterisk shows the estimates of the router next to the server. As can be seen in this figure, the estimator show a very reasonable error up to the point where link utilization is more than 95%, and when the link utilization is close to 100% the error of estimator is about 5%.

Another experiment of this case included increasing the number of perturbing routers in presence of congestion. We repeated this experiment when the number of perturbing routers were 2, and 4 routers. The results of the experiments were similar to the above case when we have only one perturbing router: the estimator shows an accurate performance over a wide range of link utilizations up to the point where we have 95 percent link utilization, and when the link utilization was more than 95 percent, the error was smaller than 5 percent in all cases.

Chapter 8

Conclusion

8.1 Summary

In this work we introduced a the framework of a new mathematical tool for optimization of routing, topology design and energy utilization in wireless sensor networks. We introduced a vector field formulation that models communication in the network, and routing is performed in the direction of this vector field at every location of the network. The magnitude of the vector field at every location represents the density of amount of data that is being transited through that location. We define the total communication cost in the network as the integral of a quadratic form of the vector field over the network area.

We introduced a mathematical machinery based on partial differential equations very similar to the Maxwell's equations in electrostatic theory. We showed that in order to minimize the cost, the routes should be found based on the solution of these partial differential equations. In our formulation, the sensors are sources of information, and they are similar to the positive charges in electrostatics, the destinations are sinks of information and they are similar to negative charges, and the network is similar to a non-homogeneous dielectric media with variable dielectric constant (or permittivity coefficient).

In one of the application of our mathematical model based on the vector fields, we offered a scheme for energy efficient routing. Our routing scheme is based on changing the permittivity coefficient to a higher value in the places in the network where we have a high residual energy of the nodes, and setting it to a low value in the places of the network where the nodes do not have much energy left. Our simulations show that our method gives a significant increase in the network life compared to the shortest path and weighted shortest path schemes.

First, we focused on the case where there is only one destination in the network. Then we extended our approach to the case where there are multiple destinations in the network. In the case of multiple destinations, we partition the network into several areas known as regions of attraction of the destinations. Each destination is responsible for collecting all messages being generated in its region of attraction. The complexity of the optimization problem in this case is how to define regions of attraction for the destinations and how much communication load to assign to each destination to optimize the performance of the network. We used our vector field model to solve the optimization problem for this case. We define a vector field that is conservative and hence it can be written as the gradient of a scalar field (also known as a potential field). Then we showed that in the optimal assignment of the communication load of the network among the destinations, the value of that potential field should be equal at the locations of all the destinations.

We introduced another application of our vector field model, which was to find the optimal locations of the destinations in the network. We showed that the vector field gives the gradient of the cost function with respect to the locations of the destinations, and hence during the design phase of a network, iterations can be performed to relocate the destinations to reduce the cost function. Performance of our proposed schemes was confirmed by several examples and simulation experiments.

In another part of this work we focused on the notions of responsiveness and conformance of TCP traffic in communication networks. We introduced the notion of responsiveness for TCP aggregates and defined it as the degree to which a TCP aggregate reduces its sending rate to the network as a response to packet drops. We defined metrics that describe the responsiveness of TCP aggregates, and then we offered two methods for determining the values of these quantities. The first method is based on a test in which we drop a few packets from the aggregate intentionally and measure the resulting rate decrease of that aggregate. This kind of test is not robust to multiple simultaneous tests performed at different routers. In the second method, we made the test robust to multiple simultaneous tests by using ideas from the CDMA approach to multiple access channels in communication theory. Based on this approach, we introduced tests of responsiveness for aggregates, and called it CDMA based Aggregate Perturbation Method (CAPM). We used CAPM to perform congestion control. A distinguishing feature of our congestion control scheme is that it maintains a degree of fairness among different aggregates.

In the next step we modified CAPM to offer methods for estimating the proportion of an aggregate of TCP traffic that does not conform to protocol specifications, and hence may belong to a DDoS attack. We introduced methods that intentionally perturb the aggregate by dropping a very small number of packets from it and observing the response of the aggregate. We offered two methods for conformance testing. In the first method, we applied the perturbation tests to SYN packets being sent at the start of the TCP 3-way handshake, and we used the fact that the rate of ACK packets being exchanged in the handshake should follow the rate of perturbations. In the second method, we applied the perturbation tests to the TCP data packets and used the fact that the rate of retransmitted data packets should follow the rate of perturbations. In both methods, we use signature based perturbations, which means packet drops are performed with a rate given by a specific function of time. We used analogy of our problem with multiple access communication to find proper signatures. Specifically, we assign orthogonal CDMA based signatures to different routers in a distributed implementation of our methods. As a result of orthogonality, the performance of such an implementation does not degrade because of cross interference made by simultaneously testing routers. We showed the efficacy of our methods through mathematical analysis and extensive simulation experiments.

8.2 Future Work

Our routing method based on the vector fields in its current state is a centralized approach. In other words, a central node needs to know the value of $r(x, y)$ for all locations of the network, and based on this information, it solves the set of partial differential equations and advertises the resulting routes in the network. While this centralized approach may be applicable in some situations, the electrostatic framework will be a more practical approach if we can find a way to decentralize it.

One way to introduce a decentralized approach is to make use of the *superposition property*. All the partial differential equations that give the paths and routes are linear, and especially, this linearity holds with respect to the value of $r(x, y)$. So if a sensor node knows its location with respect to the destinations, it can solve the set of PDEs by itself and find the incremental value of $\vec{D}$ corresponding only to the values of the function $r(x, y)$ in its vicinity. After that, it can broadcast that $\vec{D}$ to the other nodes in the network. The other nodes, add the received incremental $\vec{D}$ from that sensor node to update their total $\vec{D}$ value, and accordingly, compute the routes. This decentralized and incremental procedure could also be used to update $\vec{D}$ (and hence the routes) when the load density function $r(x, y)$ changes in some part of the network.

Another direction of extending our framework based on vector field is to extend it to the case where we have dynamics in the system. The current setup of our approach assumes the load density function $r(x, y)$ and the positions of the access points are static. If the load density function changes, then another frozen time snapshot of the function is used to compute new routes. However, if environmental conditions are changing rapidly as in a forest fire, it may be the case that the load density function changes rapidly with time. In this case, it should be viewed as a time-varying function $r(x, y, t)$, and the framework should be extended to explicitly take the load dynamics into account.

In the framework based on vector fields, $r(x, y)$ is analogous to charge density in electrostatics, so $r(x, y, t)$ should correspond to time-varying charge density. This suggests that we pursue an analogy with *electrodynamics*. Electrodynamics deals with moving electric charges or time-varying electric fields due to time-varying charges. Special rules govern electrodynamics, and every charged particle propagates waves in space that carry information about its charge. Such waves are modelled by time-dependent partial differential equations. Usually, the propagation depends on some factors such as the permittivity of the media.

In the part of conformance and responsiveness test of TCP aggregates, we assumed that the codes of different routers are orthogonal to each other. One issue of future research will be the effect and analysis of non-zero cross-correlation of the codes at the different routers. While it is possible to assign orthogonal codes to different routers, orthogonality will be maintained only if the drop signatures at all routers are implemented synchronously–i.e., all routers initiate their perturbations at the same time. Since it is not practical to synchronize perturbations, we need to use codes that are nearly orthogonal even when time-shifted.

A promising approach to solve the above problem is based on the use of *Pseudo Random Sequences*. Such sequences show a very low cross-correlation if the length of the signature sequence is long enough, and perturbations do not need to be synchronized to achieve the low correlation. However, making the length very long would cause the test to be expensive and slow. Consequently, investigating the trade off of speed and performance of the pseudo random codes is an interesting research problem.

An important problem is how to define the aggregates for the CAPM tests. While we suggest hash-based definition of aggregates, an important issue is proper definition of the hashes that helps best localization of polluted aggregates in the case of a DDoS attacks. Different IP and TCP header fields can be used for defining aggregates, and there is a trade off between the granularity of aggregates and the number of tests needed to achieve enough information on each of them.

Bibliography

[1] B. McQuistan, “Scalar and Vector Fields: A Physical Interpretation,” John Wiley & Sons, 1965.

[2] J. Jackson, “Classical Electrodynamics,” Third Edition, John Wiley & Sons, 1999.

[3] P. Matthews, “Vector Calculus,” Springer 1998.

[4] D. Cheng, “Field and Wave Electromagnetics,” Second Edition, Addison-Wesley, 1989.

[5] D. Bertsekas, and R. Gallager, ”Data Networks ,” Second Edition, Prentice-Hall, Inc., 1992.

[6] X. Hong, K. Xu,, and M. Gerla, “Scalable Routing Protocols for Mobile Ad Hoc Networks,” IEEE Network Magazine, July-Aug, 2002, pp. 11-21.

[7] C. Huitema,“Routing in the Internet,” Second Edition, Prentice Hall PTR, 1999.

[8] A. Iwata, C. Chiang, G. Pei, M. Gerla, and T. Chen, “Scalable Routing Strategies for Ad hoc Wireless Networks,” IEEE JSAC, Aug. 1999, pp 1369-79.

[9] G. Pei, M. Gerla, and T. Chen, "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks," Proc. of ICC, June 2000.

[10] C. Perkins, and E. Royer, "Ad Hoc On-Demand Distance Vector Routing," Proc. of IEEE WMCSA, Feb 1999.

[11] Y. Ko, and N. Vaidya, "Location-aided Routing (LAR) in Mobile Ad Hoc Networks," Proc. of MOBICOM, October, 1998.

[12] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," Proc. of MOBICOM, October 1998.

[13] C. Chiang, and M. Gerla, "Routing and Multicast in Multihop, Mobile Wireless Networks," Proc. of IEEE ICUPC, Oct. 1997.

[14] N. Riza, "Reconfigurable Optical Wireless," LEOS'99, Nov. 1999.

[15] E. Leonardi, M. Mellia, and M. Marsan, "Algorithms for The Logical Topology Design in WDM All-Optical Networks," Optical Network Magazine, pp. 35-46, Jan 2000.

[16] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," In Proc. of 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications DIAL M99, pages 48-55, 1999.

[17] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of MultiHop Wireless Ad Hoc Network Routing Protocols," In Proc. of the Fourth Annual ACM/IEEE International Conferenceon Mobile Computingand Networking, pages 85-97, Dallax, TX, October 1998.

[18] C. Chiang, H. Wu, W. Liu, and M. Gerla, “Routing in clustered multihop, mobile wireless networks,” in Proc. of IEEE Singapore International Conf. on Networks (SICON), pp. 197-211, 1997.

[19] D. De Couto and R. Morris, “Location proxies and intermediate node forwarding for practical geographic forwarding,” Tech. Rep. MIT-LCSTR-824, MIT Laboratory for Computer Science, June 2001.

[20] D. Johnson, and D. Maltz, ”Dynamic Source Routing in Ad Hoc Wireless Networks,” in Mobile Computing, Kluwer Academic Publishers, 1996.

[21] G. Finn, “Routing and Addressing Problems in Large Metropolitan-scale Internetworks,” Technical Report ISI/RR-87-180, University of Southern California, March 1987.

[22] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu, “Geometric spanner for routing in mobile networks,” In MobiHOC, Long Beach, CA, USA, October 2001.

[23] T. Hou, and V. Li, “Transmission range Control in Multihop Packet Radio Networks,” IEEE Transactions on Communication, Vol. 34, 1986.

[24] T. Imielinski, and J. Navas, “Gps-based addressing and routing,” RFC2009, Computer Sciece, Rutgers University, 1996.

[25] R. Jain, A. Puri, and R. Sengupta, “Geographical routing using partial information for wireless ad hoc networks,” IEEE Personal Communications, pages 48-57, February 2001.

[26] R. Jain, A. Puri, and R. Sengupta, “Geographical Routing and Geocasting for Wireless Ad Hoc Networks,” Available online at: http://citeseer.nj.nec.com/ko00anycasting.html.

[27] D. Johnson, “Scalable and Robust Internetwork Routing for Mobile Hosts,” In Proceedings of the 14th International Conference on Distributed Computing Systems, 1994.

[28] E. Kranakis, H. Singh, and J. Urrutia, “Compass Routing on Geometric Networks,” In Proceeding of Canadian Conference on Computational Geometry, 1999.

[29] W. Liao, J. Sheu, and Y. Tseng, “GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks,” In Telecommunication Systems, Volume 18, pages 37-60, 2001.

[30] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, “Geometric Ad-Hoc Routing: Of Theory and Practice,”In the Proceedings of the 22nd ACM Symposium on the Principles of Distributed Computing (PODC), Boston, Massachusetts, USA, July 2003.

[31] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris, “A Scalable Location Service for Geographic Ad Hoc Routing,” ACM Mobicom 2000.

[32] A. Qayyum, L. Viennot, and A. Laouiti, “Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks,” INRIA res. rep. RR- 3898, 2000.

[33] P. Jacquet, P. Muhlenthaler, and A. Qayyum, “Optimized Link State

Routing Protocol," draft-ietf-manetolsr- 05.txt, Internet Draft, IETF MANET Working Group, Nov. 2000.

[34] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I Stocia, "Geographic Routing without Location Information," Proc. of MOBICOM, September 2003.

[35] H. Takagi, and L. Kleinrock, "Optimal Transmission Ranges for Randomly Distributed Packet radio Terminals," IEEE Transactions on Communications, Vol 32, issue 3, pp 246-257, 1984.

[36] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," Proceedings of 5th annual IEEE/ACM international conference on Mobile Computing and Networking, pp. 263-270, 1999.

[37] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor network," ACM Wireless Networks, vol. 8, no. 2-3, pp. 169-185, 2002.

[38] F. Ye, A. Chen, S. Liu, and L. Zhang, " A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks," Proceedings of 10th international conference on computer communication and networks, pp. 304-309, 2001.

[39] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network," IEEE Personal Communications, Vol. 7, Issue 5, pp. 16-27.

[40] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks ," IEEE Communications Magazine," Vol. 40, No. 8, pp. 102-116, August 2002.

[41] P. Rentala, R. Musunnuri, S. Gandham, and U. Saxena "Survey on Sensor Networks ," Available online at: http://www.mdpi.net/sensors.

[42] L. Subramanian and R. H.Katz, "An architecture for building self-configurable systems," in *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing.* Boston, MA, Aug. 2000.

[43] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. IT-46, no. 2, March 2000.

[44] A. Agarwal and P. R. Kumar, "Improved capacity bounds for wireless networks," *Wireless Communications and Mobile Computing*, vol. 4, 2004.

[45] A. Agarwal and P. R. Kumar, "Capacity bounds for ad-hoc and hybrid wireless networks," *ACM SIGCOMM Computer Communications Review, Special Issue on Science of Networking Design*, vol. 34, no. 3, July 2004.

[46] O. Dousse, P. Thiran, and M. Hasler, "Connectivity in ad-hoc and hybrid networks," in *Proceedings of IEEE Infocom 2002.* New York.

[47] I. Rubin, R. Zhang, and H. Ju, "Topological performance of mobile backbone based wireless ad-hoc network with unmanned vehicles," in *Proceedings of IEEE Wireless Communication and Networking Conference*, 2003.

[48] B. Liu, Z. Liu, and D. Towsley, "On the capacity of hybrid wireless networks," in *Proceedings of IEEE Infocom*, 2003.

[49] U. Kozat and L. Tassiulas, "Throughput capacity of random ad hoc networks with infrastructure support," in *Mobicom*, 2003.

[50] S. Lindsey, C.Raghavendra, and K. Sivalingam, "Data gathering in sensor networks using energy delay metric," in *International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing.* San Francisco, CA, Apr. 2001.

[51] A. Manjeshwar and D. P. Agrawal, "Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks," in *2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing.* Ft. Lauderdale, FL, Apr. 2002.

[52] S. Bandyopadhyay and E. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Proceedings of INFOCOM 2003, Vol. 3, pp. 1713-1723.*

[53] M. Kalantari and M. Shayman, "Energy efficient routing in sensor networks," in *Proceedings of Conference on Information Sciences and System.* Princeton University, March 2004.

[54] M. Kalantari and M. Shayman, "Routing in wireless ad hoc networks by analogy to electrostatic theory," in *Proceedings of IEEE International Communications Conference (ICC-04).* Paris, France, June 2004.

[55] S. Toumpis and L. Tassiulas, "Efficient node deployment in massively dense sensor networks as an electrostatics problem,"

in *Proceedings of IEEE Infocom 2005.* available online at: http://userver.ftw.at/ toumpis/publications/infocom05.pdf.

[56] N. T. Nguyen, A. A. Wang, P. Reiher, and G. Kuenning, "Electric field routing: A reliable framework for routing in manets," in *Proceedings of ACM SIGMOBILE Mobile Computing and Communications Review, 2004.*

[57] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[58] H. Schulzrinne, "Long-term traffic statistics," *http://www.cs.columbia.edu/ hgs/internet/traffic.html/.*

[59] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *Computer Communications Review 32:3,*, July 2002. [Online]. Available: http://www1.cs.columbia.edu/ smb/papers/pushback-CCR.pdf.

[60] J. Ioannidis and S. Bellovin, "Pushback: router-based defense against DDoS attacks," in *Proceedings of NDSS*, Feb. 2002.

[61] T. Peng, C. Leckie, and K. Ramamohanarao, "Defending against distributed denial of service attack using selective pushback," in *Proceedings of IEEE International Conference on Telecommunications*, 2002.

[62] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network support for IP traceback," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, June 2001.

[63] S. Bellovin, M. Leech, and T. Taylor, “The ICMP traceback messages,” in *Internet draft, work in progress*, Feb. 2003.

[64] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountiob, S. Kent, and W. Strayer, “Hash-based IP traceback,” in *Proceedings of ACM Sigcomm*, Aug. 2001.

[65] M. Adler, “Tradeoffs in probabilistic packet marking for IP traceback,” in *Proceedings of the ACM Symposium on Theory of Computing*, 2002.

[66] D. Dean, M. Franklin, and A. Stubblefield, “An algebraic approach to IP traceback,” in *Proceedings of the Network and Distributed System Security Symposium*, Feb. 2001.

[67] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, B. Schwartz, S. Kent, and W. Strayer, “Single-packet IP traceback,” *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, Dec. 2002.

[68] A. Mankin, D. Massey, C. Wu, S. Wu, and L. Zhang, “On design and evaluation of intention driven ICMP traceback,” in *Proceedings of the International Conference on Computer Communications and Networks*, Oct. 2001.

[69] H. Burch and B. Cheswick, “Tracing anonymous packets to their approximate source,” in *Proceedings of the USENIX Large Installation Systems Administration Conference*, Dec. 2000, pp. 319–327.

[70] T. Daniels and E. Spafford, “Network traffic tracking systems: folly in the large?” in *Proceedings of the Workshop on New Security Paradigms*, Feb. 2001.

[71] R. Stone, “Centertrack: an IP overlay network for tracking dos floods,” in *Proceedings of the USENIX Security Symposium*, Aug. 2000.

[72] D. Song and A. Perrig, “Advanced and authenticated marking schemes for IP traceback,” in *Proceedings of the IEEE Infocom*, Apr. 2001.

[73] K. Park and H. Lee, “On the effectiveness of probabilistic packet marking for IP traceback under a denial of service attack,” in *Proceedings of the IEEE Infocom*, Apr. 2001.

[74] M. Goodrich, “Efficient packet marking for largescale IP traceback,” in *Proceedings of the ACM Conference on Computer and Communication Security*, 2001, pp. 117–126.

[75] P. Ferguson and D. Senie, “Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing,” *RFC 2827*, 2000.

[76] J. Mirkovic, “D-ward: Source-end defense against distributed denial-of-service attacks.” [Online]. Available: citeseer.ist.psu.edu/mirkovic03dward.html

[77] S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, 1999.

[78] J. Widmer, D. Denda, and M. Mauve, “A survey on TCP-friendly congestion control,” *IEEE Network*, May/June 2002.

[79] S. Floyd, M. Handley, and J. Padhye, "A comparison of equation-based and AIMD congestion control," unpublished; Preliminary version available online. [Online]. Available: http://www.aciri.org/floyd/papers.html

[80] S. Floyd and K. Fall, "Router mechanisms to support end-to-end congestion control," *LBL Tech. Report*, 1997. [Online]. Available: http://www.nrg.ee.lbl.gov/nrg-papers.html

[81] W. Feng, K. Shin, D. Kandlur, and D. Saha, "The Blue active queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 513–528, 2002.

[82] S. Reddy, "LRU-RED: An active queue management scheme to contain high bandwidth flows at congested routers," in *Proceedings of Global Telecommunications Conference*, 2001.

[83] Network Simulator, ns2, *http://www.isi.edu/nsnam/ns/*.