A Simple Metric for Ad Hoc Network Adaptation

Stephen F. Bush, *Senior Member, IEEE*

***Abstract*—This paper examines flexibility in ad hoc networks and suggests that, even with cross-layer design as a mechanism to improve adaptation, a fundamental limitation exists in the ability of a single optimization function, defined *a priori*, to adapt the network to meet all quality-of-service requirements. Thus, code implementing multiple algorithms will have to be positioned within the network. Active networking and programmable networking enable unprecedented autonomy and flexibility for ad hoc communication networks. However, in order to best leverage the results of active and programmable networking, metrics that indicate the nature and location of required flexibility need to be developed. The primary contribution of this paper is to propose a metric that couples network topological rate of change with the ability of a generic service to move itself to an optimal location in concert with the changing network. This metric points to a fundamental tradeoff among adaptation (changing service location), performance (sophistication or estimated minimum code size of the service), and the network's ability to tune itself to a changing ad hoc network topology.**

***Index Terms*—Active networks, ad hoc networks, communications systems, complexity theory, computer network management, network performance, network quality-of-service (QoS), optimization methods.**

I. Introduction

AD HOC NETWORKS are a subset of variable topology networks. The goal of variable topology networks is to maintain message delivery as network topology varies. Nodes should be able to dynamically form transient networks. Nodes, which may be located on rapidly moving platforms such as aircraft, should be able to join, leave, and rejoin networks which may form at any time. Networks spontaneously form and their topologies can change rapidly. An additional challenge imposed by airborne and heterogeneous air and ground environments is the ability to provide predictable and optimized quality-of-service (QoS) for information transmission over highly dynamic topologies. This paper proposes a service location metric, the beta metric,[1] which quantifies the ability of a service to maintain an optimal location in such highly dynamic networks. If the service improves the operation of the network, then another benefit is the ability to aid in the determination of when, where, and what degree of flexibility needs to be added to the network, given the network's topological rate of change.

Optimal QoS in ad hoc networks requires that the network architecture support adaptation to a rapidly changing infrastructure. The degree to which the network must adapt is dependent on the rate of change of topology. QoS requirements are most often stated in the form of an optimization problem with a cost function that is optimized by adaptation within the network. An applicable result from complexity theory, the no free lunch theorem (NFL) [8], expresses a limit on the ability of any single algorithm, or protocol, to meet all QoS requirements. Across all optimization functions, the average performance of all algorithms is the same. Thus, if an algorithm performs well on one set of problems, then it will perform poorly (worse than random search) on all others. The NFL has proven that all algorithms perform the same as they search for an extremum when averaged over all cost functions. If a potentially good algorithm appears to outperform a poor algorithm on some cost functions, then there exist exactly as many cost functions where the apparent poor algorithm will outperform the good algorithm. In other words, no single algorithm, or in this specific case, ad hoc protocol, can optimize over all possible QoS constraints as shown in the first NFL theorem (1), where f, m, and a_n are the cost function, number of iterations, and algorithms, respectively. d_m^y is the cost of data of sample size m. $P(f)$ is the probability that f is the actual cost function being used. As discussed in [8], $P(f)$ represents the knowledge one may have about a particular class of problems, for example, a particular class or type of optimization problem

$$\sum_f P\left(d_m^y|f,m,a_1\right) = \sum_f P\left(d_m^y|f,m,a_2\right). \tag{1}$$

In this paper, optimization of a single service is considered. A service is defined simply as any unit of code that provides a service to clients. The service may be an application or network service such as reduced jitter, increased compression rate, improved forward error correction, etc. In addition, it is assumed that service code is capable of moving from one node to another and contains within itself the ability to determine when and where it should move. The movement algorithms are distinct from one another; each possible movement algorithm cannot be placed on all nodes in a resource-constrained network. Coordination of multiple services of the same type would involve complicating communication among the services and is a future extension of this work. Clients utilize a QoS metric that is both suitable and consistent with respect to the service. The expected value of all client QoS metric values at a particular instant is the fitness of the service. For such a service in a resource-constrained ad hoc network there are two interesting parameters in the above theorem: 1) the size and number of data samples (assumed to be polled from each node in order to allow services to decide when and where to migrate) and 2) the number, size, and placement of algorithms within an ad hoc network. In particular, if the algorithm is a service location algorithm that allows a service to be optimally positioned and whose decision is based

Manuscript received June 11, 2003; revised April 8, 2005. This work was supported in part by Lockheed Martin Shared Vision.

The author is with the GE Global Research Center, Niskayuna, NY 12309 USA (e-mail: bushsf@research.ge.com).

Digital Object Identifier 10.1109/JSAC.2005.857204

[1]Originally referred to as the Bush Metric.

upon sampled data to be preprocessed, aggregated, and compressed, but the network nodes are resource constrained such that they cannot either hold or execute all possible service and movement algorithms, then a tradeoff is necessary. The bottom line is that multiple algorithms (services) will be required to perform optimally over all cost functions in order to meet all client QoS requirements. The problem is managing when and how to locate algorithms (services) in a dynamically changing network. Two high-level frameworks that are flexible and customizable enough to allow change in algorithmic content within networks are: active networks [2] and programmable networks [4]. Programmable networks allow the control software of a network to be dynamically reprogrammed. Active networks are an extreme form of programmable networks that allow code and data to travel through and modify the function of the network. Active packet code can execute on any node along the path that the packet travels.

Current research tends to search for a fixed algorithm that has enough degrees of freedom that optimal operation can be found by tuning a fixed set of parameters. This may be due in part to the difficulty in breaking away from the passive (as opposed to active) architecture of the Internet protocols have a strong grip on the mind-set of most researchers. There is potential in examining the latter form of adaptation (dynamically changing algorithms), particularly in light of the implication of the NFL which suggests that simply tuning a fixed algorithm will not be as optimal as changing to a better algorithm given a different cost function, and nodes cannot hold all possible algorithms.

Because this work is focused on ad hoc networks, a metric is required that relates the degree of dynamism within the ad hoc network to QoS. By dynamism, we mean the complexity and rate of change of a variable topology. More specifically, a critical requirement is how quickly the system can adapt as a function of the complexity and rate of change of the topology. It can be argued that only a subset of QoS requirements are likely to be requested by a reasonable user under reasonable operating conditions. However, the envelope of what will be considered reasonable, particularly under military conditions, will constantly expand. An ad hoc network has the goal of eliminating inefficiency caused by competing fitness requirements. For example, one protocol layer attempts to reduce congestion while another attempts to maximize throughput. In addition to the problem of multiple users with potentially competing fitness requirements within a single protocol or application, there are also multiple protocol layers in the same protocol suite that show inefficiencies due to competing fitness requirements. As a specific example, maximizing quality at the link layer could entail dropping links that degrade in quality, while rapidly adding new ones that form. If link changes are highly variable and the link layer places too high a demand on link quality, then network layer overhead rises as it tries to efficiently maintain routes over changing links. An optimal transport layer will minimize congestion, but will react improperly when delays are due to link changes rather than actual congestion. These mismatched approaches to achieving disparate fitness requirements within each layer become highly exacerbated in an ad hoc environment.

Adaptation is defined as autonomic change in network state designed to maintain optimal communication performance. Most network functionality falls under some form of adaptation. The problem with adaptation as defined above is that algorithms defined *a priori*, that is, before installation within the network, are developed to meet high performance standards for specific predefined cases within the network. These algorithms do not have the flexibility, or adaptability, required to meet unanticipated conditions. However, to go a step further, the definition of adaptation should include self-directed change by an algorithm designed to maintain optimal performance. Adaptation is self-directed change in an algorithm, or program code, designed to maintain optimal ad hoc network performance.

Section II considers the impact of node mobility in the form of vector fields upon network topology. The rate of change of network topology and its impact on the ability of a service to move from node to node is defined as the beta metric and the relation between movement algorithm sophistication; the corresponding impact on code size is briefly discussed in the context of a complexity estimation technique known as minimum description length (MDL). Section III relates QoS and node movement patterns. Section IV presents the results of the proposed metric in ad hoc network simulations and also considers the relation to the cluster coefficient from the simulations of optimal service location. In Section V, the beta metric is experimentally tested in an ad hoc network testbed. Section VI summarizes the results.

II. Dynamically Changing a Network Service

Changing an algorithm requires changing code. It is anticipated that services will require capability to adjust location within an ad hoc network in order to maximize QoS to all clients. Code that remains resident to only one node will not be guaranteed to reside in an optimal location given ad hoc movement patterns. Installing all potentially useful code on all nodes requires too much overhead, particularly, for low capability ad hoc nodes and sensors. Eventually, we anticipate self-composing services within the network. The location and movement of the service will be critical to its performance. The factors influencing service migration include not only basic route acquisition delays, but also the size of the service code. Later, in this paper, the results of routing delays upon service migration performance are simulated. Route acquisition delays have the impact of potentially providing false network topology information and could cause the service to migrate toward an incorrect location. Large service code size might be due to a complex service or it may be due to the overhead of the service location algorithm itself. Specifically, we consider the case of a single service injected into an ad hoc network. Because the nodes are mobile, and because the direction and velocity of the nodes can be highly variable, it is often the case that nodes will be unreachable. In this case, a penalty must be assigned for "no QoS." A metric is required that captures the precision, including overhead, with which the network is able to maintain an optimal location for the service that maximizes overall QoS normalized by the amount of dynamism in the network topology. Symbols used in the reminder of this paper are defined in Table I.

The metric proposed in this paper regards the ability of the ad hoc network to maintain a single service in an optimal location maximizing fitness as the network topology changes. By

TABLE I
SERVICE CODE MIGRATION SYMBOL DEFINITIONS

Symbol	Meaning
P_c	(Probability of Connection) The ratio of the number of actual links to the total number of possible links in a network topology.
n	(Number of Nodes) The number of nodes in a network graph.
d	(Diameter) The diameter of a network graph is defined as follows. Let $d(u,v)$ be the graph distance, that is, the length of the shortest path between two nodes, u and v. The graph diameter is $\max_{u,v} d(u,v)$ between all nodes.
$\hat{q}$	(QoS Estimate) A Quality of Service estimate based upon number of nodes and diameter of a service graph topology.
QoS_α	(Client QoS) The quality of service as seen by client α.
β	(Service Location Performance) The ratio of the service migration rate to rate of change in network diameter.
η_s	(Service Migration Performance) Transmission rate of a service in hops per second.
η_d	(Rate of Change in Network Diameter) The rate of change of the network diameter in hops per second.
$len(S)$	(Service Code Size) The service code size in bytes.

means of simulation, the optimal location can be determined at all times. It is assumed that the service is equally valuable to all clients and that the optimum service to all clients is achieved when the variance in the number of hops from the service node to all client nodes is minimized. The graph center is defined as the nodes in the graph with minimum eccentricity, where eccentricity is the maximum distance between a given node and all other nodes in the graph. This places the service in the graph center, where the graph vertices are nodes and graph edges are network links. This is defined as an optimal location for the service. Edge weights may be either uniform or weighted to represent either link capacities or client priorities for a service. When edge weights are nonuniform, the center of mass of the graph represents the optimal service location. Rate of change of network topology is measured as the average rate of change of network diameter η_d, or graph center, in units of hops per second. Let S denote the network service program and $len(S)$ denote the size in bytes of the network service program. The average hop rate of a service η_s is also measured in hops per second.

The ratio shown in (2) is a dimensionless service location performance metric, referred to as β. The relation $\beta \geq 1$ is a necessary condition for optimal service migration because this relation indicates that the service can move rapidly enough relative

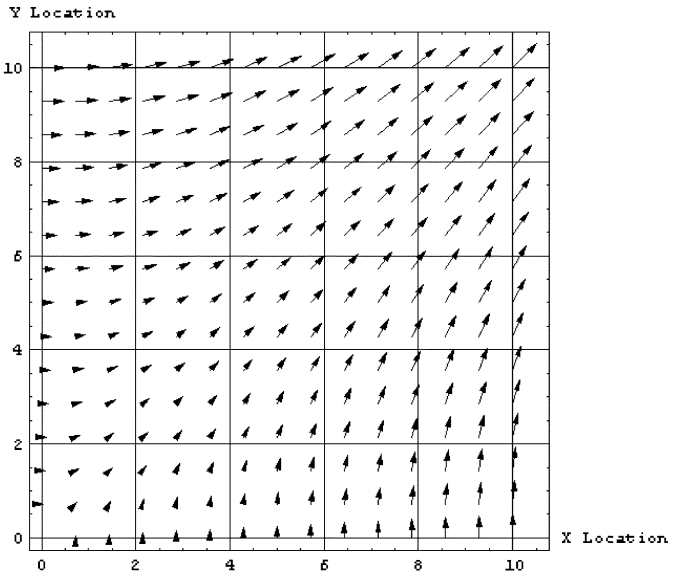

Fig. 1. Gradient vector field $\vec{v} = (xi, yj)$ inducing node movement.

to changes in network topology to maintain an optimal location. An underlying assumption is that the service chooses the most direct route from its current location to its latest knowledge of the optimal location and does not attempt random moves or use *a priori* knowledge to anticipate node movement. The relation $\beta < 1$ indicates that service location will be suboptimal because, even with perfect knowledge and movement strategy, the service cannot move quickly enough relative to changes in topology. When $\beta = 1$, one would expect that the service could maintain an optimal location only if initially placed near the optimal location. If the service were not initially located in a near-optimal position, then, with $\beta = 1$, the service could not move quickly enough relative to the changing network topology to maintain optimal QoS. Note that η_s is a function of service code size $len(S)$ and link transmission rate ρ, as shown in (3)

$$\beta = \frac{\eta_s}{\eta_d} \tag{2}$$

$$\eta_s = \frac{\rho(\text{bits}/\sec)/\text{hop}}{len(S)}. \tag{3}$$

A specific example demonstrates the application of β. Fig. 2 shows node movement for ten nodes randomly located on a Cartesian coordinate plane acted upon by forces described by the vector field shown in Fig. 1. Each arrow represents a new location after one-second intervals. Each node has a radius of communication of 0.5 units. Nodes located along the diagonal accelerate away from the origin faster than nodes along the x and y axes. As nodes move with varying velocities and directions, their relative positions change, inducing a change in network topology. A single service, residing on a single node, may reside in different subnetworks over time as the network topology changes. For example, a network may split apart into smaller subnetworks and the service node may end up in a smaller subnetwork or become isolated, depending upon its direction and velocity relative to other nodes. Fig. 3 shows the rate of change of diameter, in hops per second, as a function of subnetwork containing a specified service node for the example in Fig. 2. A larger service will require more transmission time

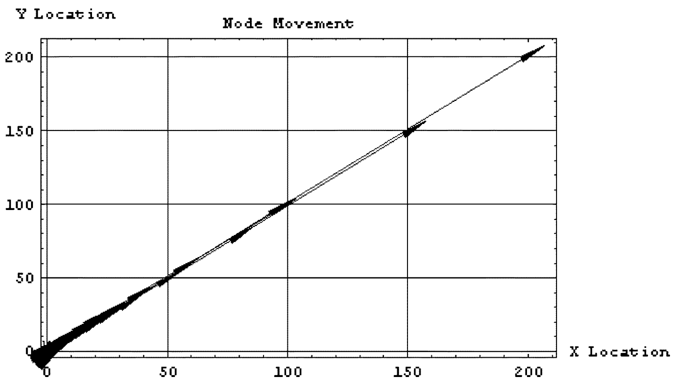

Fig. 2. Node movement for which β is computed. Each arrow represents a new location after one-second intervals.

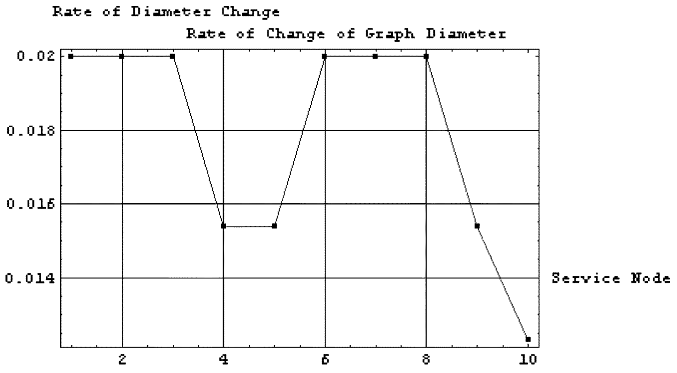

Fig. 3. Rate of change in network diameter (hops/s) as a function of service node for simulated node movement shown in Fig. 2.

for migration, while faster links will reduce transmission time. Fig. 4 shows β as a function of service program length (bytes) and expected link rate (bps) in this specific case. For example, Fig. 4 can be used to determine service lengths that can be supported at a given link rate for an acceptable value of β.

Decreasing $len(S)$ increases η_s, thus, there is motivation for small service implementations. The advantage of β is that it represents a measurable quantity with values readily available in an operational ad hoc network. A disadvantage of β is that it does not consider the effectiveness of the strategy utilized by the service location code itself, it only considers whether movement capability exists. In this analysis, it is assumed that location strategy is part of the mobile service; the strategy moves with the service. A future goal will be to examine the tradeoff between service location code size and accuracy.

Service migration is dependent upon an accurate internal model that indicates the best location for the service. A sophisticated movement algorithm would tend to require more code and maintain more state, thus reducing the movement rate of the service. A less sophisticated algorithm will tend to be implemented with less code; the smaller code size will enable increased service movement rate but the choice of location may not be as accurate. This is a classic MDL problem [1]. MDL ([5] and [7]) relates algorithmic code size (model) to its performance (error). Let D_x be a binary string representing x. Let H_x be a hypothesis, in algorithmic form that attempts to explain how x is formed. MDL states that the sum of the length of the shortest encoding of a hypothesis about the model generating the string and the length of the shortest encoding of the string encoded by the hypothesis will estimate the Kolmogorov Complexity of string x, $K(x) \approx K(H_x) + K(D_x|H_x)$. A method for determining $K(x)$ separates randomness from nonrandomness in x by incorporating nonrandomness, which is computable, as the shortest encoded program that represents the original string. The random part of the string represents the error, that is, the difference between the original string and the output of the encoded program. Thus, the goal is to minimize $l(H_e) + l(D_x|H_e) + l(E)$, where $l(x)$ is the length of string x, H_e is the hypothesis used to encode the string (D_x), and E is the error in the hypothesis, $D_x - (D_x|H_e)$. The more accurately the hypothesis describes string x, the shorter the encoding of the string. However, a large hypothesis works against achieving the MDL. In the case of service migration, the hypothesis is the algorithm that determines the optimal service location. The data is the state information observed to make the decision. We propose a relatively simple hypothesis based upon maintaining location within the graph center as explained later.

A diameter-based QoS metric is defined as the ratio of the number of nodes n in the largest fully connected service subnetwork, which assumes all nodes are clients, and the diameter d incremented by one, of the same subnetwork. These values are then normalized by the maximum diameter metric $\hat{q}_{\max}$, that is, the maximum number of clients and smallest diameter, as defined in (4). Fig. 5 shows a comparison between the diameter-based QoS metric and the measured QoS (expected inverse hop count to reach all clients nodes) for a series of random network topologies with ten nodes with a probability of connection ranging from zero to one in 0.05 increments

$$\widehat{q} = \frac{\frac{n}{(d+1)}}{\widehat{q}_{\max}}. \tag{4}$$

The actual QoS is computed based upon inverse average hop-count to all nodes in the subnetwork. Note that all links have a weighted value of one when computing shortest path. Nodes that are disconnected from the service node in an ad hoc subnetwork suffer a hop count penalty that is set to be arbitrarily large, in this case 100. Both the actual QoS and the diameter-based QoS metric can vary from a minimum of zero in which a node is an isolated service node to one where all clients exactly one hop away from the service node. The diameter-based metric appears to provide a reasonable indication of QoS.

One would expect the probability of connection P_c defined as the proportion of links that actually exist divided by the total number of possible links to play a role in the optimal location of a service node. However, it is not just the probability of connection, but the actual topology that plays a critical role. In particular, the length of the longest shortest path distance in terms of hops is the network diameter d. The number of nodes in the network is n. If the rate of change of network diameter per second is defined in (5) and the number of nodes in the induced service graph per second as shown in (6), then the ratio of nodes to network diameter is shown in (7)

$$\eta_d = \mathrm{d}/\sec \tag{5}$$

$$\omega = \mathrm{n}/\sec \tag{6}$$

$$\frac{\omega}{\eta_d} = \frac{n}{d}. \tag{7}$$

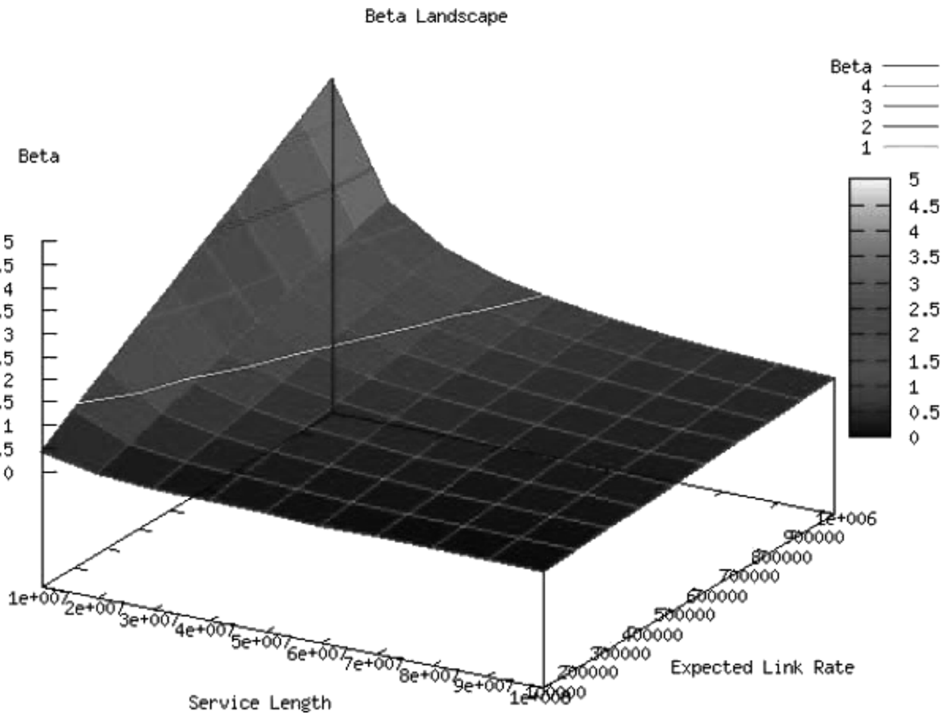

Fig. 4. A β surface as a function of service code length (bytes) and expected link rate (bits/s). Faster service migration and slower changing diameter result in a higher value of β.

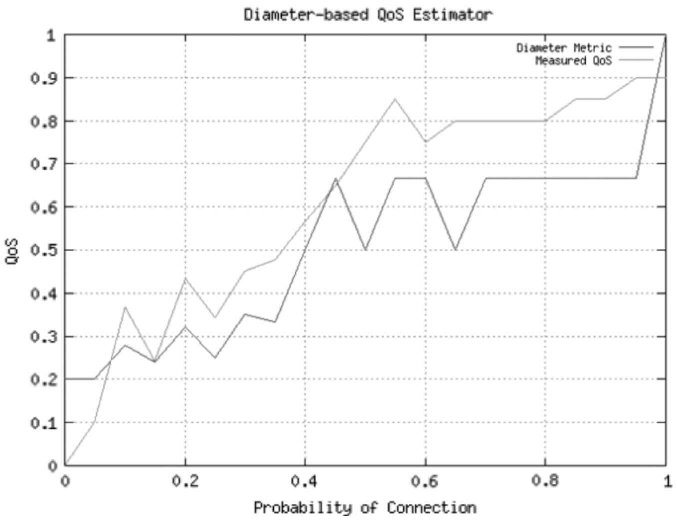

Fig. 5. Diameter-based metric is compared with measured QoS values for randomly generated network topologies.

It is possible for a service to reside on a node in a subgraph that is disconnected from the rest of the network. In this case, d and n refer to the largest connected subgraph in which the service resides and an estimate of the best service capability is n/d. The next section considers the impact of node movement patterns on QoS.

III. Dynamics of Network Topologies

The assumption of a single service and uniform interface capability implies that service quality is inversely related to the number of hops between client and service nodes. This is supported by [6] which examined the impact of varying packet size, beaconing interval, and route hop count on route discovery time, communication throughput, end-to-end delay, and packet loss. Throughput was found to be more dependent on packet size (bytes) and route length (hops) with the exception of very high beaconing frequencies. Route discovery time was more dependent on channel conditions (bit-error rate) and route length (hops) than variations in beaconing intervals. In this work, fitness is the average distance (hops) between the service node and each reachable client node. Nodes are assumed to be mobile with transient links; later in this paper, rather than explicitly modeling node movement, network connectivity varies based upon a specified probability of connection. This section develops the relationship between node movement and link connectivity.

The simplest case is to consider the fitness of the ad hoc network when a single server is located at an optimal service node location and attempts to provide service to all nodes in the network. Finding the optimal service location in an ad hoc network in a low-overhead manner is a challenge given the constantly changing network topology. Fig. 6 shows an analytically determined QoS based upon expected inverse hop count to all client nodes with a penalty for disconnected clients, given a single service that can move itself onto any of 20 nodes of a randomly connected network.

The network topology induced by changes in node movement can be examined via vector fields. Consider a two-dimensional (2-D) coordinate system, however, instead of (x, y) coordinates, consider an (i, j) coordinate system, where i and j are functions of x and y. First consider the simple case of constant vector field $\vec{v} = (1i, 0j)$. This shows a constant movement along the x direction and no change in the y direction. Fig. 7 shows node movement for five nodes placed randomly within

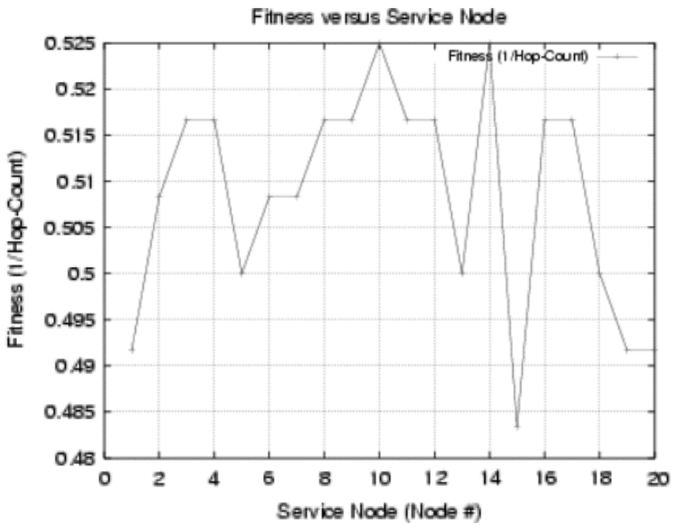

Fig. 6. Fitness is plotted as service code migrates within a randomly generated constant topology network.

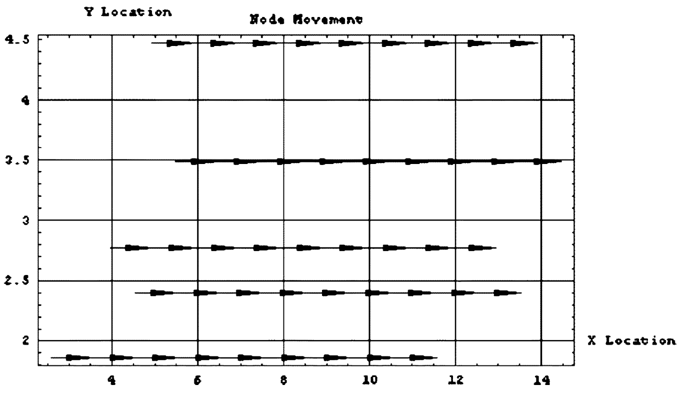

Fig. 7. Node movement is plotted at one-second intervals for vector field $\vec{v} = (1i, 0j)$.

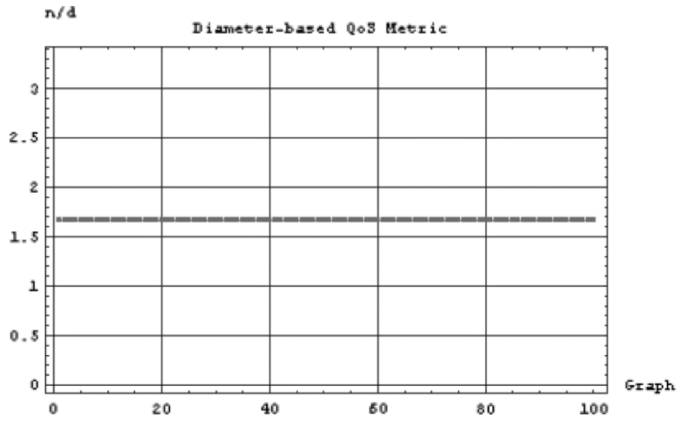

Fig. 8. Diameter-based metric values for the vector field defined by $\vec{v} = (1i, 0j)$.

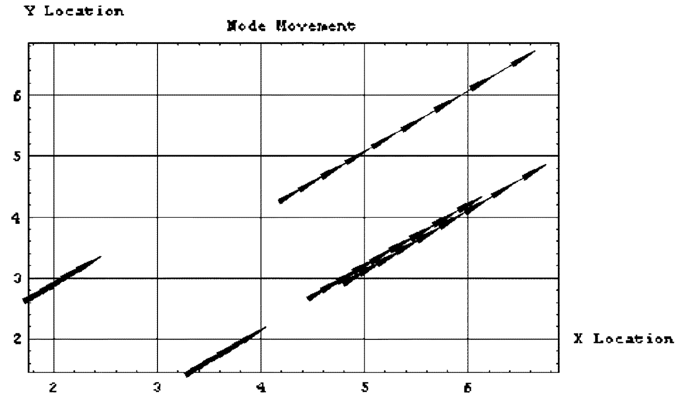

Fig. 9. Movement pattern for ten randomly placed nodes in vector field $\vec{v} = (xi, yj)$ is plotted at one-second intervals.

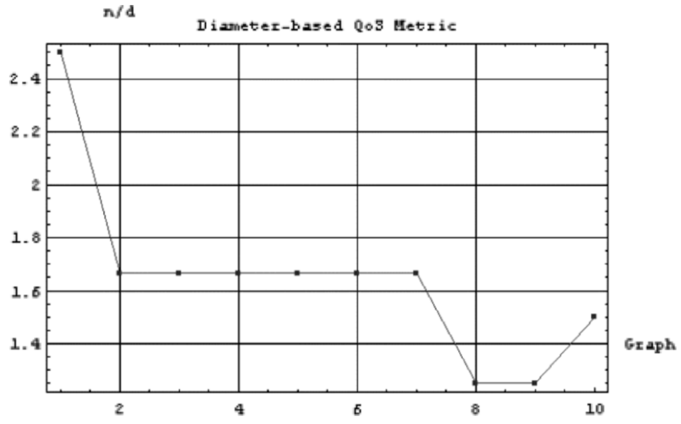

Fig. 10. Diameter of the network topology for node motion in vector field $\vec{v} = (xi, yj)$.

the field. Fig. 8 shows the diameter-based metric as the nodes move given a communication radius of two units. In this vector field, the nodes maintain a constant relative position; therefore, the network topology remains constant.

Consider a slightly more interesting vector field corresponding to $\vec{v} = (xi, yj)$. This field yields an increase in velocity as x and y increase. The gradient field plot was plotted previously in Fig. 1. Nodes located in the upper right of the gradient pattern will move away faster than nodes located closer to the origin. The movement pattern is shown Fig. 9; the longer arrows to the right of Fig. 9 indicate faster movement as expected. This should cause the network diameter to stretch, possibly causing a break in connectivity at some point. The diameter-based QoS estimate is shown in Fig. 10 for a communication radius of two units. As expected, the QoS estimate shows a negative slope. At the last point in Fig. 10, the fastest node disconnects from the network reducing the network diameter among the existing clients causing a slight increase in QoS. A QoS surface is shown in Fig. 11; QoS rises with the probability of connection P_c though not always at the same rate for a given service location.

The curl $(\nabla \times \vec{v})$ at various locations throughout the area of movement provides interesting information regarding irrotational locations. When the curl is zero at a specific point in the field, there is no net flow through that point. Thus, nodes moving into zero curl points tend to remain in those locations. The curl is shown in Fig. 12. The slope in the plane of the curl indicates the rate at which the network will be stretched.

A more interesting example is shown for $\vec{v} = (\sin(x)^2 i, \cos(y)^2 j)$. In Fig. 13, the gradient is shown. In this field, nodes will tend to clump around a pattern of locations. In Fig. 14, a node movement trace is shown that verifies this fact. In Fig. 15, the network diameter-based QoS metric is

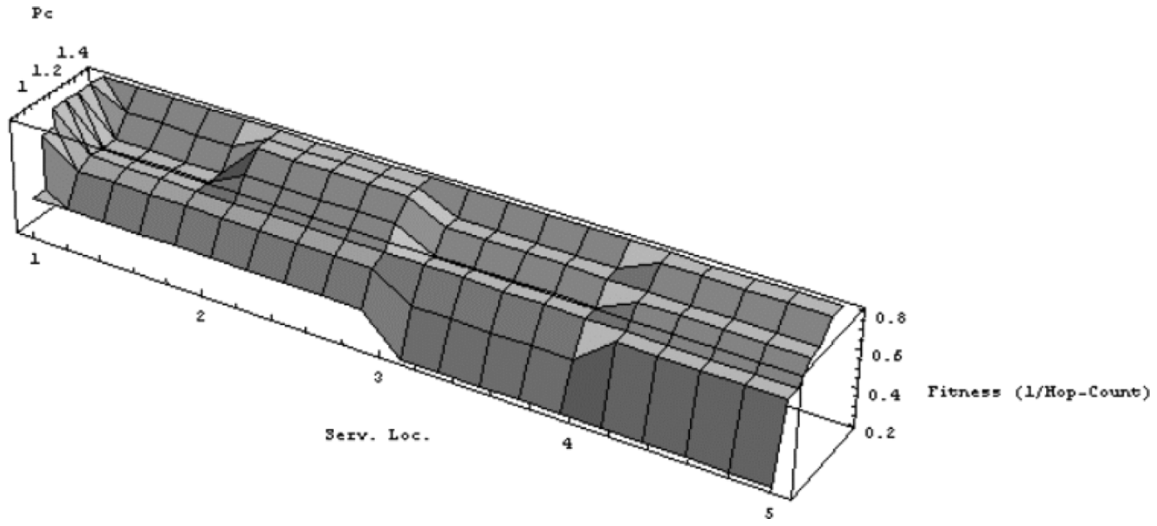

Fig. 11. QoS surface is shown for vector field $\vec{v} = (xi, yj)$.

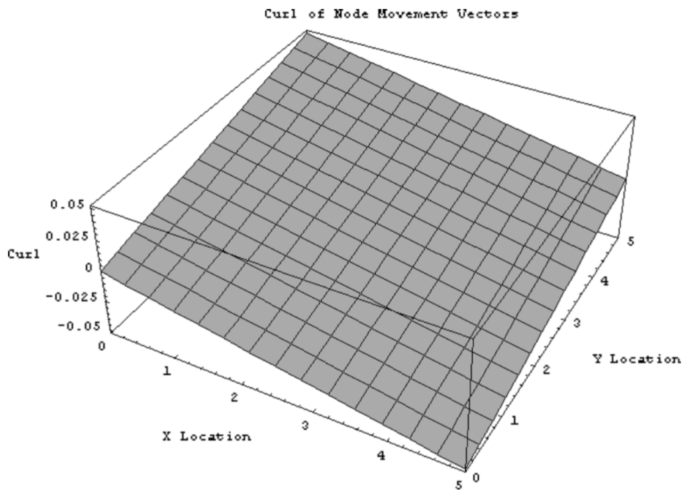

Fig. 12. The curl of vector field $\vec{v} = (xi, yj)$.

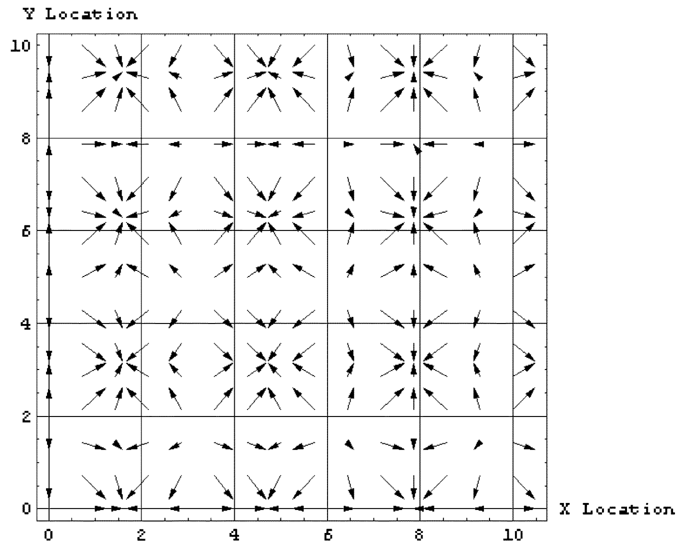

Fig. 13. Gradient of the vector field $\vec{v} = (\sin(x)^2 i, \cos(y)^2 j)$.

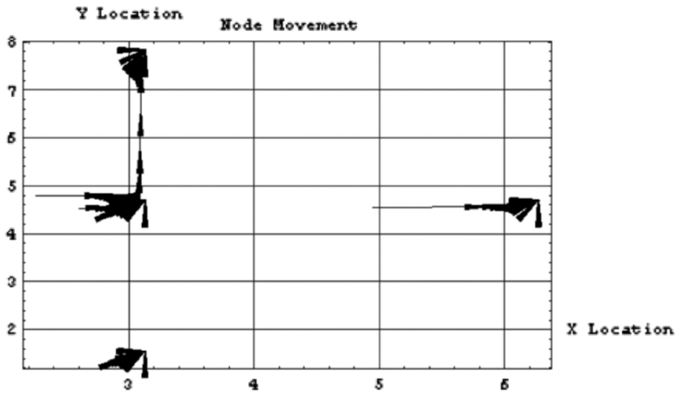

Fig. 14. Movement pattern for ten randomly placed nodes in vector field $\vec{v} = (\sin(x)^2 i, \cos(y)^2 j)$.

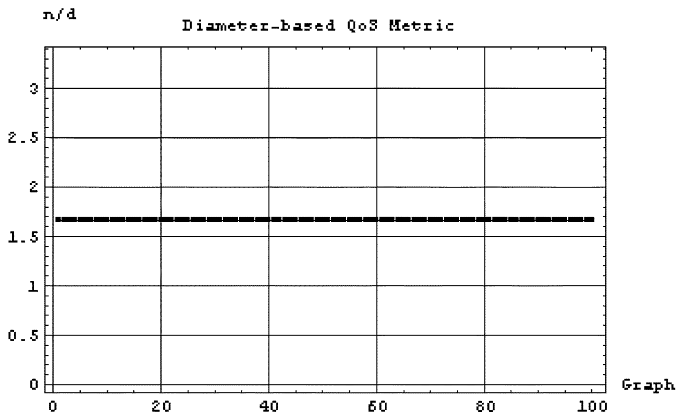

Fig. 15. Diameter of the network topology for nodes moving in vector field $\vec{v} = (\sin(x)^2 i, \cos(y)^2 j)$.

shown. The communication radius remains at two units; the clustering behavior caused the nodes to maintain a constant diameter. In Fig. 16, the QoS surface is shown; since the node movement pattern caused nodes to be attracted to clusters that were within communication radius, the connectivity remained relatively constant. Finally, in Fig. 17, the curl is shown for $\vec{v} = (\sin(x)^2 i, \cos(y)^2 j)$. The topology remained relatively constant as nodes were pulled within irrotational areas (zero

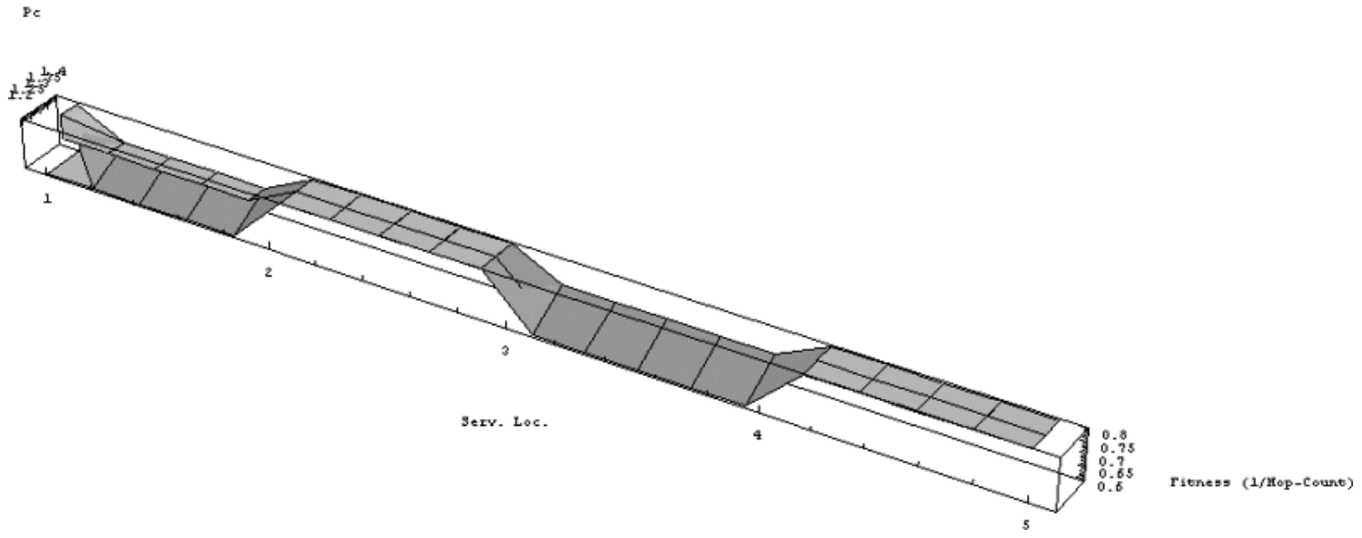

Fig. 16. The QoS surface for nodes moving within vector field $\vec{v} = (\sin(x)^2 i, \cos(y)^2 j)$.

TABLE II
SERVICE MIGRATION CASE STUDIES

Case	Type	Description
Base	No Service Migration	Single service remains upon a single, specific node
Naïve	Naïve Service Migration	Single service polls all network nodes in order to determine optimal next move
Perfect	Perfect Service Migration	Single omniscient service always moves to the best location without inducing any overhead

curl) of the field. The next section analyzes simulation results with regard to the impact of node movement on QoS.

IV. AD HOC SERVICE LOCATION SIMULATION

Swarm is a software framework for the simulation of multiagent simulation models (also called ABMs, short for Agent-Based Models). Low-level actors interact within the Swarm framework. A common goal of Swarm simulations is to provide agents with simple preprogrammed behavior and view the overall patterns that emerge. The intent of this paper is to consider ad hoc network QoS characteristics that emerge from a suitable abstraction of routing and transport. In keeping with the concept of simple interacting agents, each node has an attribute of mass and moves in response to applied force vectors. The space upon which the nodes move is a 2-D toriod. Collisions among nodes are possible. There is currently no damping; the full force of a collision is transferred to the movement of colliding nodes.

An abstraction of a proactive routing protocol is modeled via simple rules within each node. The rules include the following which occurs at every simulation time step: each node updates its location based upon applied forces, determines its nearest neighbor, exchanges routing information between itself and its nearest neighbor, and if any change has occurred, that information is then propagated to all nodes currently reachable by both nodes. Thus, link state changes are proactively propagated.

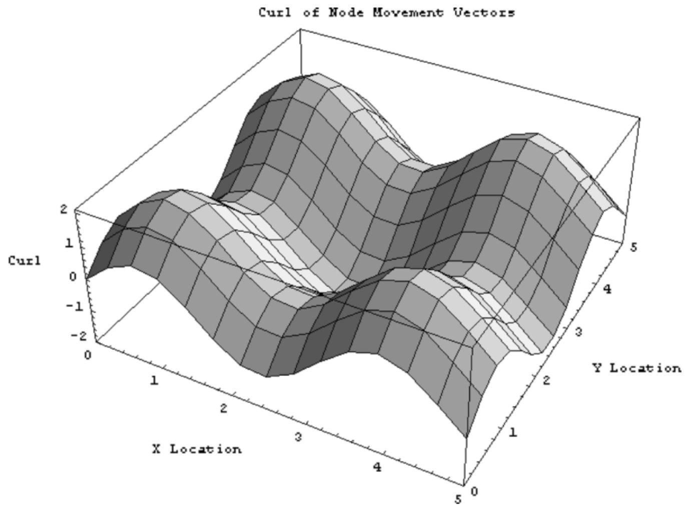

Fig. 17. The curl of the vector field $\vec{v} = (\sin(x)^2 i, \cos(y)^2 j)$.

A. Simulation Results

In order to test the ad hoc performance metric proposed in this paper, three service migration cases are considered, as shown in Table II.

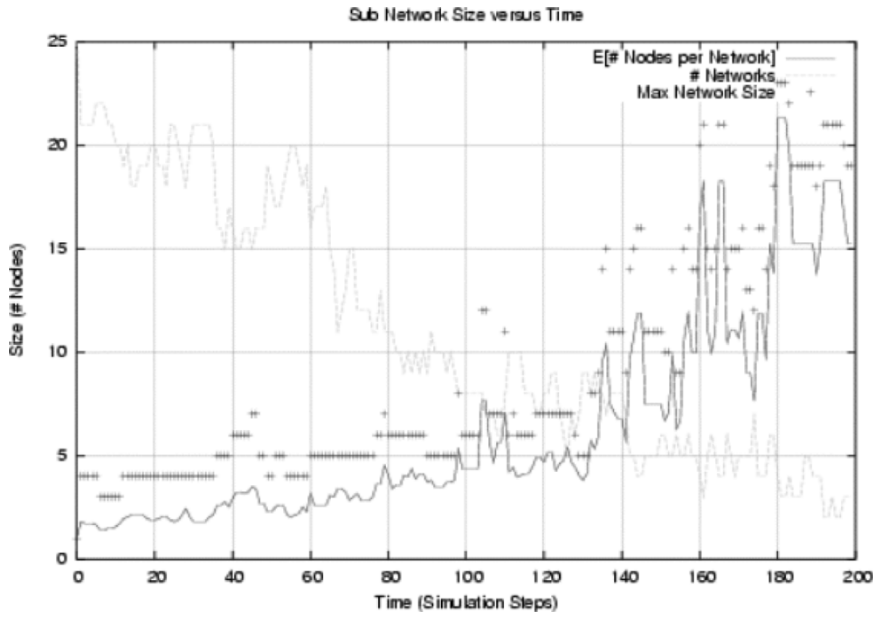

Fig. 18. Maximum subnetwork size and number of subnetworks as nodes moved based upon applied forces during the simulation applied to all cases in Table II.

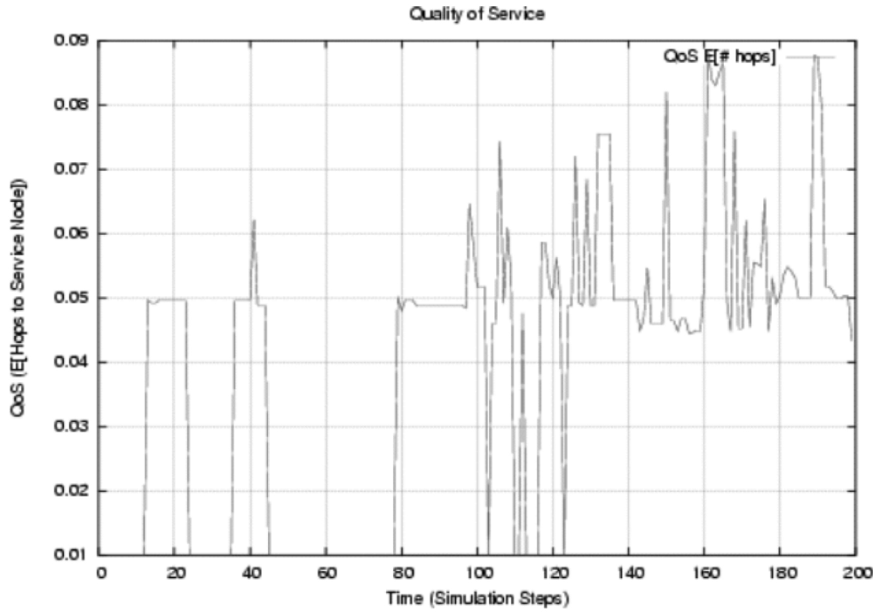

Fig. 19. The QoS for a single nonmigratory service measured for the perfect case in Table II.

Simplifying assumptions in this simulation are the following.

1) The QoS is inversely related to expected hop count to all client nodes.
2) Polling overhead adds load to the network. The polling overhead is dependent upon poll rate ξ, polling packet size, assumed to be 1% of link capacity, and number of hops to the queried node whose maximum value is limited by δ.

A Swarm simulation was developed using autonomous agents to represent network nodes. A Swarm agent is an autonomous entity; there is no global simulation control. The simulation constructed for this model enables each node to move and communicate within a specified radius. Nodes that move within range of another node automatically join that node's ad hoc network. The simulation is purposely simplified in order to discover the most important attributes in an ad hoc environment. In this simulation, nodes periodically announce their existence and, if in range of another node, exchange routing information. At each simulation time step each node moves and checks whether it is in range of any other node. This is equivalent to announcing a "hello" message. For each node within range, the node that just moved exchanges routing information forming an ad hoc network. Many simulation parameters exist for each node such as: communication radius, hello message announcement rate, and speed and movement characteristic. The size of the area and number of nodes are also parameters of the simulation.

Included in the simulation is the capability for each Swarm agent to host a network service. Currently, there is only one service per simulation and the service can migrate from node to

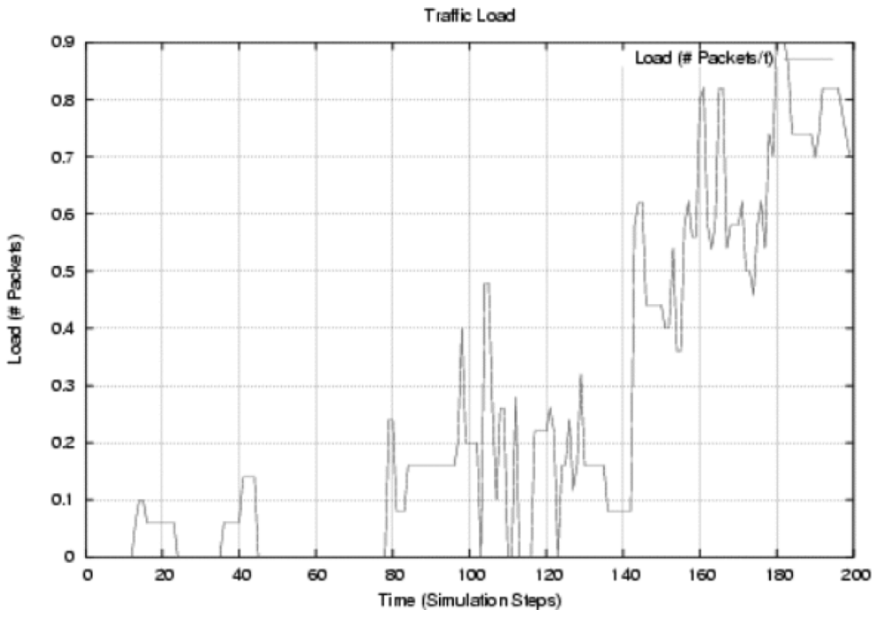

Fig. 20. Routing and service location overhead observed in the simulation for the naïve case in Table II.

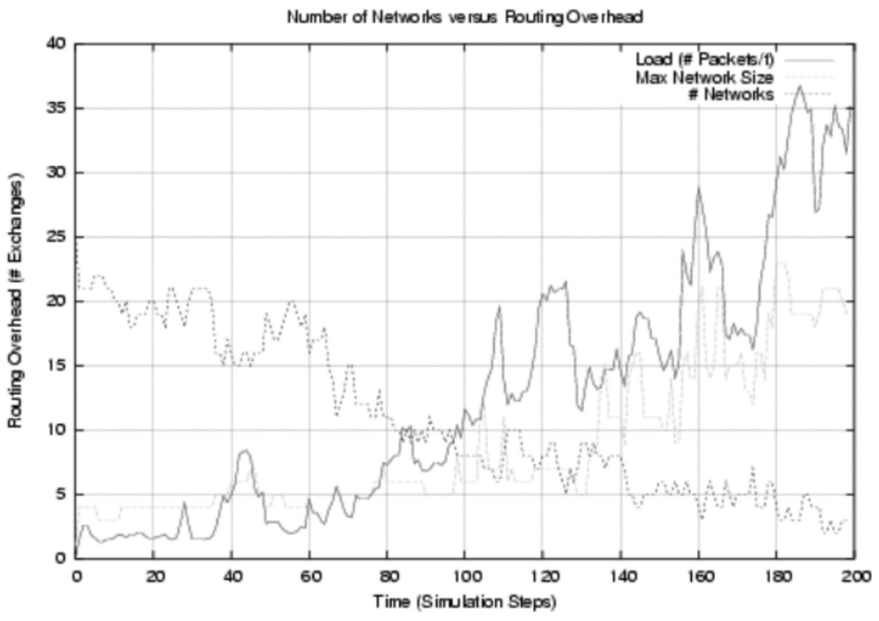

Fig. 21. Routing and location overhead versus network topological characteristics such as number of subnetworks and maximum subnetwork size applied to all cases in Table II.

node. QoS is measured by computing the expected hop-count from the node currently hosting the service to every reachable node.

There are many probes in Swarm to collect data. Routing, hello announcement, and service location overhead each measure the number of their respective messages that are exchanged between nodes. Expected network size, maximum network size, and number of networks are continuously monitored during the simulation.

B. Ad Hoc Network Topologies

In order to study the impact of transmission radius, the transmission radius is allowed to increase as the simulation runs. As the radius increases, the maximum and expected subnetwork size increases, while the number of subnetworks decreases. There are 25 mobile nodes each with a radius of communication of five units, where the radius grows by one unit every ten units of time. The nodes move with a speed and direction governed by an initial set of (x, y) velocity vectors. Six randomly chosen nodes have velocity vectors of (1.0,1.0), (1.0, -1.0), (-1.0, 2.0), (0.5, 1.0), (2.0, -1.0), (2.0, -2.0), and (1.5, 1.5). The area of movement is a toriod of area 80 $\times$ 80 units. Fig. 18 shows the maximum and average subnetwork size, as well as the total number of subnetworks during the simulation. Initially, with a small communication radius, very few ad hoc networks form. The result is a large number of small subnetworks.

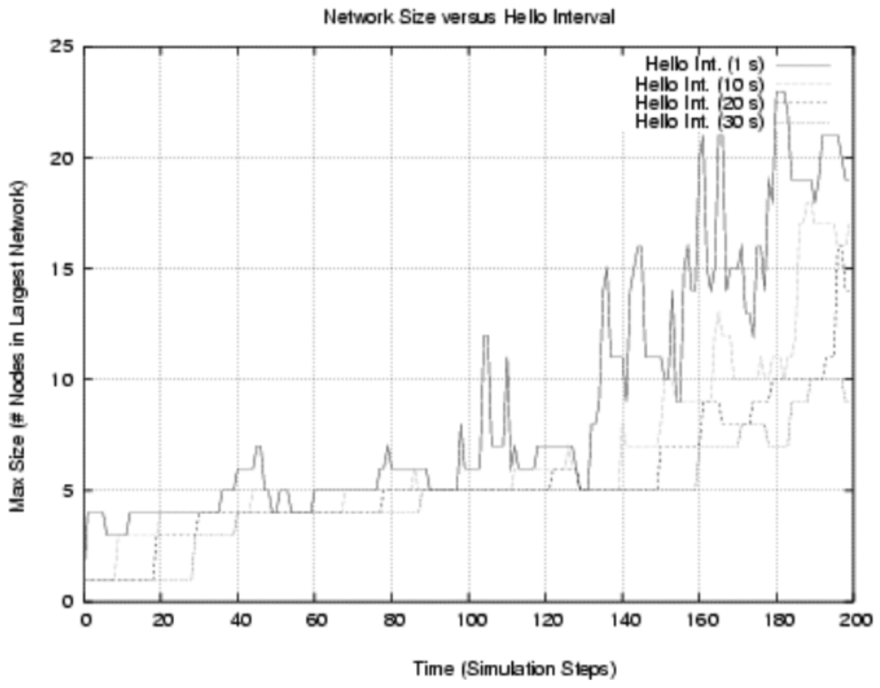

Fig. 22. Maximum network size versus the hello routing update interval applied to all cases in Table II.

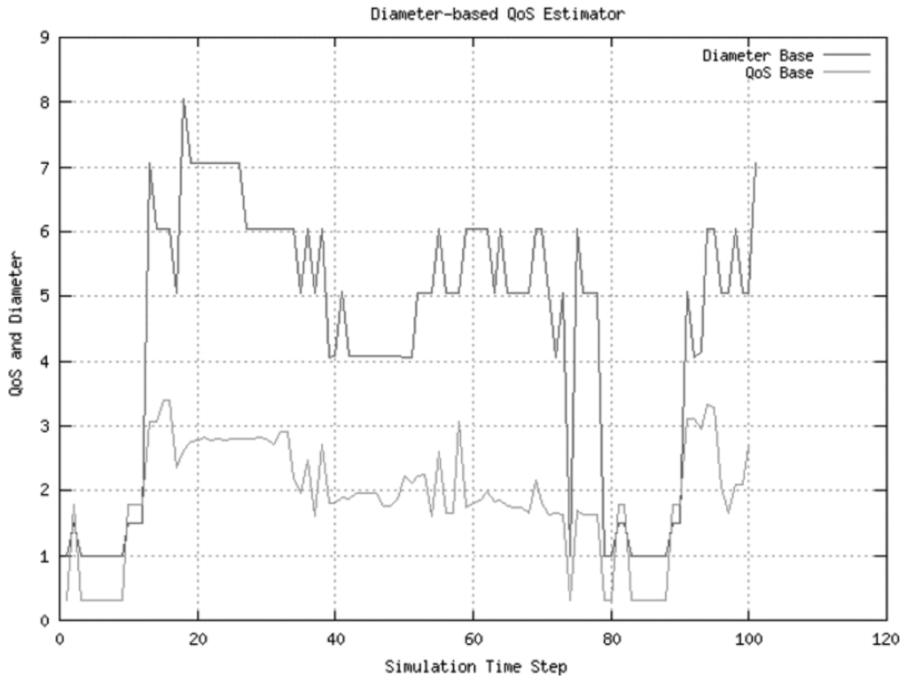

Fig. 23. Simulated base case in Table II (no code migration) QoS versus the analytically derived diameter-based QoS metric values.

Fig. 19 shows the QoS for a single service injected at random into the ad hoc network. The QoS improved when more nodes joined the subnetwork in which the service was located and also as the service migrated toward the center of mass of the network topology. Compared with Fig. 18, it is clear that the larger networks formed after time step 120 allowed for a higher QoS. Note that the service location algorithm used in these graphs is a "perfect" mechanism. The service polls every reachable node continuously in order to move as a quickly as possible to the optimal location. A single node service has a QoS that improved with larger subnetwork size. It is instructive to compare Figs. 18 and 19. It can be seen that as larger ad hoc networks began to form in Fig. 18 and as the service happened to reside in these larger networks, the QoS increased, as shown in Fig. 19. Note that it is possible for large subnetworks to form without the service node. This accounts for the occurrence of large networks without a corresponding improvement in QoS.

Fig. 20 shows the traffic load as a function of simulation time. Note that the only traffic load in this simulation is a broadcast from the service node to the client nodes in order to estimate

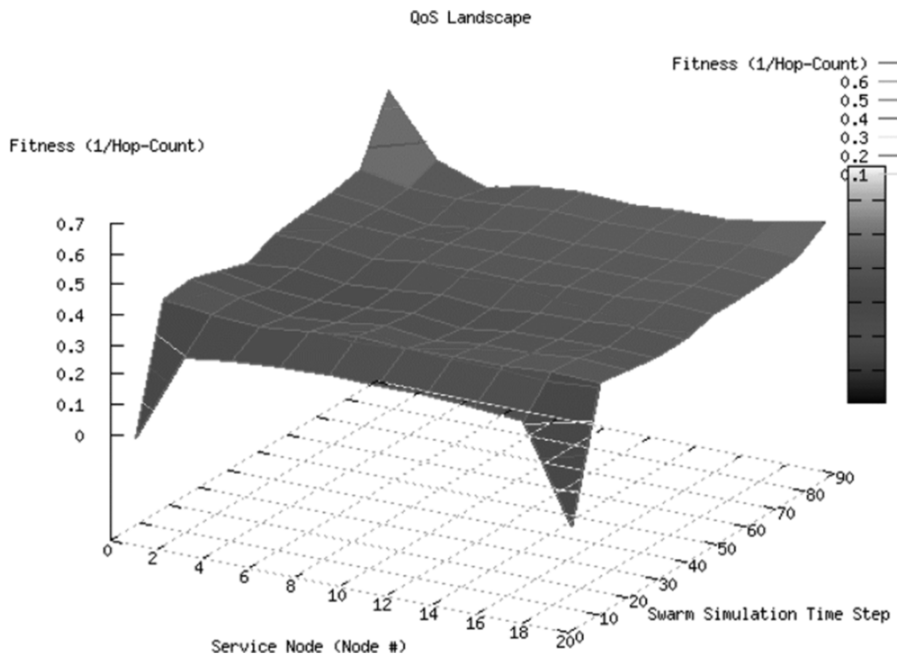

Fig. 24. Swarm simulation QoS surface for initial vector sum (0.0, 0.0).

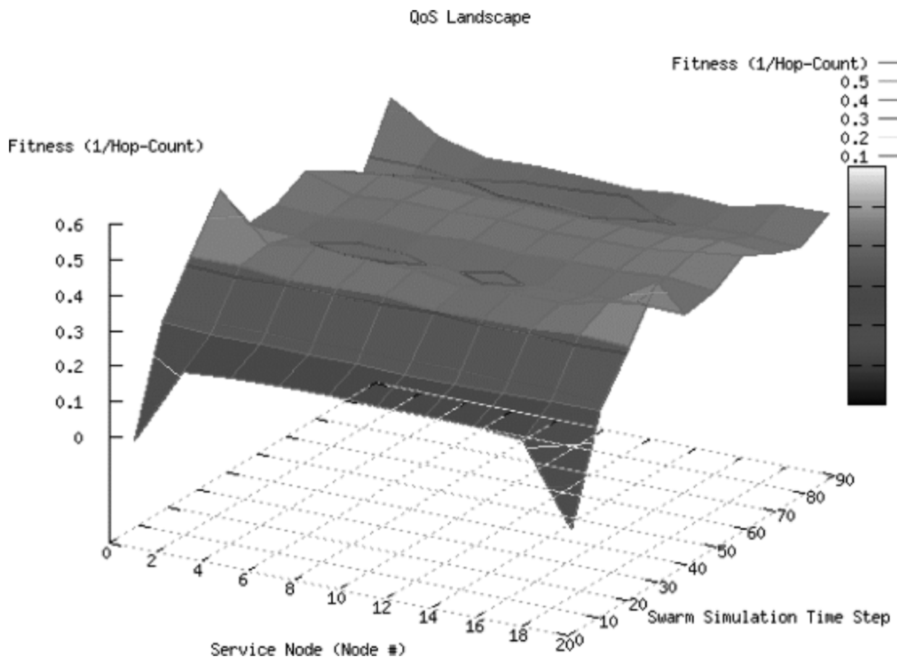

Fig. 25. Swarm simulation QoS surface for initial vector sum (5.0, 2.5).

their QoS. The traffic increases with the size of the networks. This measurement is distinct from routing overhead shown in Fig. 21.

Fig. 21 shows routing overhead as a function of network size. When the network was partitioned into as many uniform network sizes as possible, the routing overhead was minimal. As subnetworks formed, the routing overhead increased.

Ad hoc networking algorithms require a means of discovering when the topology changes. This discovery is often implementing by broadcasting an announcement of some type, often called a beacon, or hello, message. Frequent broadcasts improve accuracy, however, they also consume bandwidth. This becomes extremely critical in a highly dynamic network because link changes can be frequent. Fig. 22 shows the change in subnetwork sizes as the Hello Interval is increased. Instantaneous announcements shown in the solid curve show perfect topological knowledge. As the rate of announcement messages generation is reduced, the effect on network topology can be seen. Essentially, the high-frequency topological changes are not captured. The partitioning of networks into subnetworks increases and the maximum network size decreases, as shown in Fig. 22. Although greater partitioning occurs, the benefit is that hello announcement bandwidth usage is reduced.

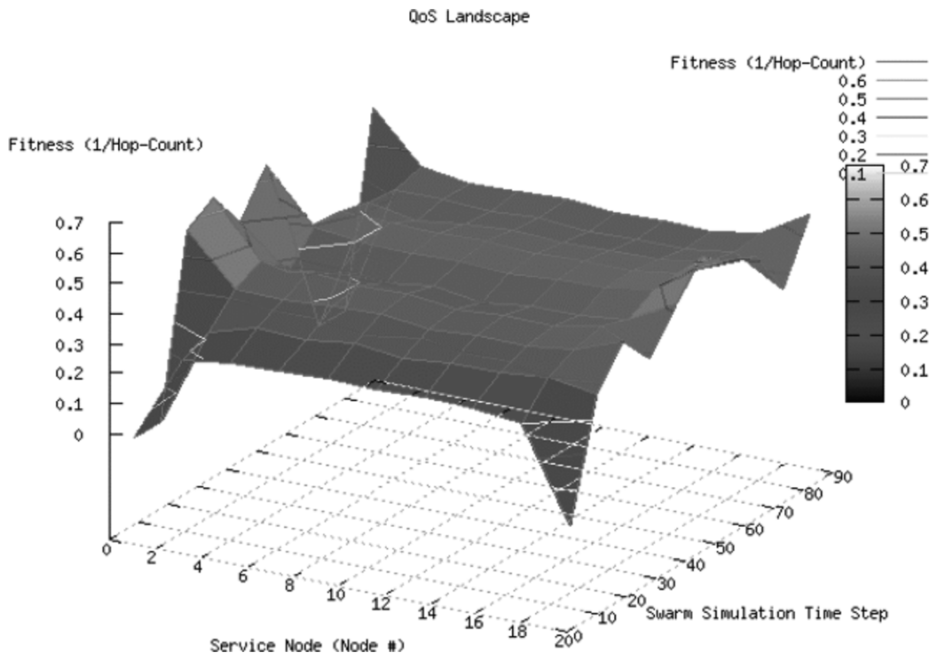

Fig. 26. Swarm simulation QoS surface for initial vector sum (7.5, 9.0).

TABLE III
INITIALLY APPLIED SIMULATION VECTORS

Node	Applied Vector Set A	Applied Vector Set B	Applied Vector Set C
Node 1	(1,1)	(1, -1)	(1, -1)
Node 2	(1, -1)	(-1,1)	(-1,1)
Node 3	(-1,2)	(2, -2)	(2, -2)
Node 4	(0.5,1)	(0.5,0.5)	(0.5,0.5)
Node 5	(2, -2)	(-0.5, -0.5)	(-0.5, -0.5)
Node 6	(1.5,1.5)	(2, -2)	(-2, -2)
Sum	(5, 2.5)	(7.5,9)	(0,0)

In Fig. 23, the QoS from the Swarm simulation is measured versus the diameter-based QoS metric from (4) for the base case in which no service migration occurs. In this case, no effort is made by the service to move to an optimal location although the node may happen be in an optimal location. A large penalty is applied to nodes that are disconnected from the service subnetwork, thus, the diameter-based metric tends to be larger.

If node movement vector functions were determined for a given scenario, then a low or zero value of curl $(\nabla \times \vec{v})$ at various locations throughout the area of movement would provide interesting information regarding irrotational, or conserved movement. The Swarm simulation applies force vectors only to initial node movement. The relationship among the force vectors in this mobility model should have an impact upon the QoS surface. A vector summation is used to characterize the movement. The intuition behind this characterization is that applying initial vector components of similar size and direction will tend to keep the network in a similar topology, much like the constant gradient field shown previously in Fig. 7. In contrast, vectors with randomly size and directions will also have a uniform impact on the network topology, as shown in Fig. 23. From a service migration point of view, as long as the network maintains a high probability of connectivity and the diameter remains low, the service will have an opportunity to migrate to an optimal location. This can occur whether the nodes are relatively static or moving rapidly in random patterns, that is, with a low curl. In other words, nodes appearing to move randomly will also tend to yield network topologies that provide a high diameter-based QoS metric value.

The QoS surface plots in Figs. 24–26, are intended to show the smoothness of the surface versus the initially applied force vectors. The initially applied vectors in the Swarm simulation are shown in Table III. The QoS results from the Swarm simulation are examined for the effects of initially applied vectors upon QoS. The QoS surface in Fig. 24 (Applied Vector Set C) is the smoothest compared with the surfaces shown in Fig. 25 (Applied Vector Set A) and Fig. 26 (Applied Vector Set B). Using Applied Vector Set C, the applied force vectors cancel

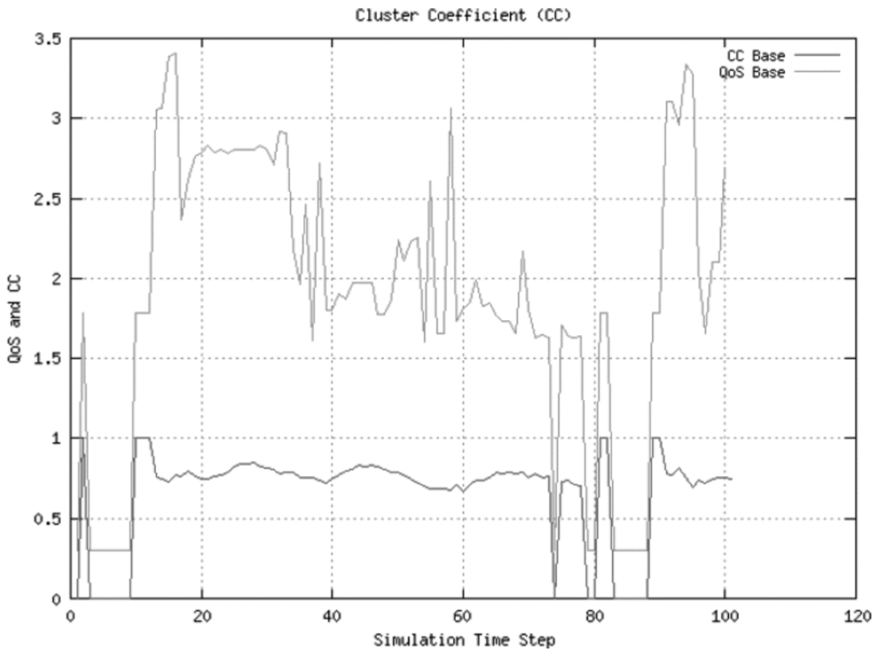

Fig. 27. Cluster coefficient and QoS for the base case (no service migration) from the swarm simulation.

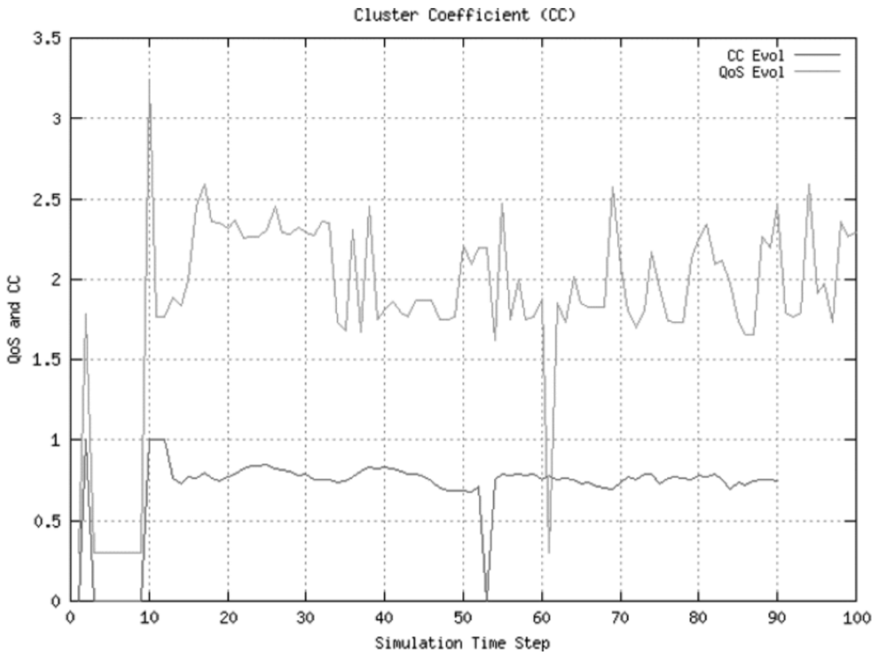

Fig. 28. Service migration QoS versus cluster coefficient. The migration mechanism maintains the QoS at a higher value relative to the cluster coefficient versus the base case shown in Fig. 27.

one another resulting in a more uniform mixing of nodes. Vector Sets A and B have increasingly larger aggregate vector summations resulting in a greater impact on network topology and increasing variation in QoS surfaces.

C. Cluster Coefficient and Ad Hoc Network QoS

The cluster coefficient is defined in (8). The cluster coefficient has been used to identify networks with small world behavior. These are networks having seemingly contradictory characteristics; neighboring nodes are very likely to be connected to one another forming local clusters, yet any two nodes in the entire network can be reached via a small number of hops. This paper does not consider whether the service network topologies form small world graphs, but rather looks at the cluster coefficient versus the measured QoS from the Swarm simulation. The total number of vertices is N. The number of neighbors in the ith vertex is k_i. The number of edges among all such neighbors is v_i. The cluster coefficient measures for each node how many neighboring nodes are connected

$$C = \frac{\Sigma_{i=1}^{N} \frac{2v_i}{k_i(k_i-1)}}{N}. \tag{8}$$

TABLE IV
SERVICE CODE MIGRATION ACTIVE NETWORK TEST ENVIRONMENT

Node Hardware	Dell Latitude C610
Wireless Speeds	Auto Mode 2 Mbs to 11 Mbs
Operating System	Redhat 9.0 Kernel 2.4.20
Fitness	Min. Var. and Exp. hop count (h), $\frac{1}{E[\{h\}+Var[\{h\}]}$
Active Execution Environment	Magician [2]

TABLE V
SERVICE MIGRATION EXPERIMENTAL VALIDATION TEST RESULTS

Rate of Change of Graph Center	0.0001392 hops/sec (hps)
Rate of Service Movement	0.308064 bps
Service Size	17,063 bytes

The cluster coefficients for the Swarm simulation results are shown in Fig. 27. Sudden drops in the cluster coefficient correspond to drops in QoS. A high cluster coefficient is necessary, but not sufficient to achieve better QoS unless the service node is in the optimal location. A comparison of cluster coefficients without service migration versus service migration can be seen in Figs. 27 and 28. The next section presents experimental validation of the beta metric and QoS on a wireless Linux testbed implementation.

V. EXPERIMENTAL VALIDATION OF THE BETA METRIC

In order to experimentally validate the proposed beta adaptation metric, an active network is run on a wireless infrastructure, described in Table IV, allowing an ideal environment for network services to migrate from one node to another. The goal of this experiment is to examine the impact of code transfer in a changing topology, measure the β value, and use it to evaluate design tradeoffs.

The topologies formed during the experiment are shown in Fig. 29. The larger, lighter circle within a node indicates the location of a service and the smaller darker circles indicate probes used to determine fitness of the service in alternative service locations. In this case, all decisions are based upon hop count as analyzed previously. The network grows from three to five nodes in a linear topology changing from Topology A to B, and then forms a cluster changing from Topology B to C. The service is challenged to maintain optimal location as these changes take place.

The results for this particular experiment are shown in Table V. Measurements of the rate of change of graph center and service transmission rate are dependent upon accurate clock measurements. NTP was used to maintain clock synchronization. More efficient wireless clock synchronization techniques are discussed in [3]. Topology formation times are dependent upon accurate and timely route update information. The relatively small network allowed for negligible delays in routing updates among the nodes, that is, each node sensed the complete network topology at the same instant of time. Note that the service size does not include protocol overhead, the service is a Java class that resides within user datagram protocol (UDP) and the active network encapsulation protocol [2]. The service contains code that determines its optimal location including creating, transmitting, and receiving probes. The service's primary function is to manipulate and intelligently relay a video traffic stream to multiple destinations.

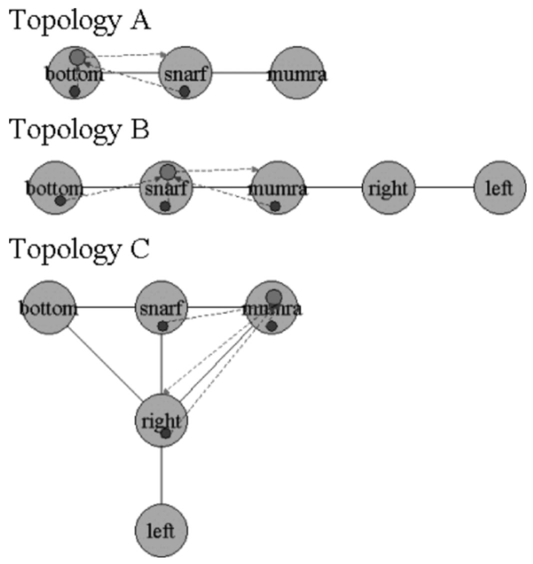

Fig. 29. Three snapshots of topologies formed within the wireless testbed.

The resulting β value is shown in Fig. 30. The value is shown on the resulting β surface. If the topological rate of change in this experiment was typical of expected usage, then the β metric indicates that a much larger service could have been supported. The tradeoff in service size and expected link rate can be obtained from this surface for a given β.

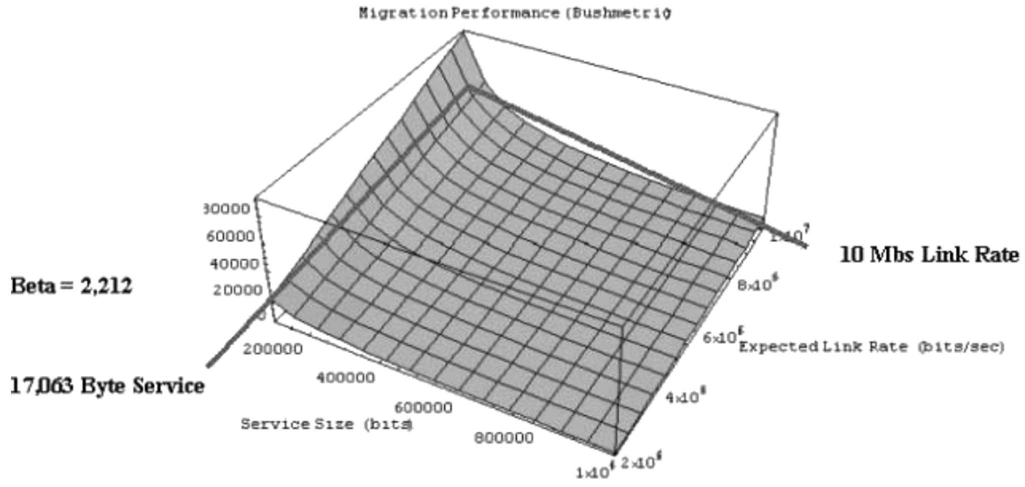

Fig. 30. The tradeoff in service size and expected link rate for a range of beta values.

VI. Conclusion

This paper has suggested that, even with cross-layer design, a fundamental limitation exists in the ability of a single *a priori* optimization function to meet all QoS requirements. Metrics that best indicate the location and nature of required flexibility need to be developed. Such a metric was proposed and evaluated in this paper and was applied to an active network. Maximizing QoS will require adaptation of algorithms, rather than simply tuning static algorithms. Algorithms can change form either *in vivo* (within a node as part of normal network operation), or they can be defined *a priori* (before or external to network operation); in either case, the code implementing the algorithm must be positioned in an optimum location within the network as the topology changes. Network topology is dependent upon patterns of node movement and those effects were examined via vector fields. It was demonstrated that vector field representation of node movement can be directly applied to the estimation QoS and the requirements for service migration performance, namely, service transmission rate as a function of code size and expected link transmission rate. Finally, the beta metric value was experimentally validated for mobile code implemented within a wireless ad hoc network testbed that indicated the performance of the code to maintain an optimal position relative to node movement.

Acknowledgment

The author thanks R. Spackmann for his tireless efforts in performing the experimental validation of the Bush Metric, S. Kupst from Lockheed Martin for her gracious support of this research, and C. Forth for her encouragement and promotion of new ideas.

References

[1] S. F. Bush, "Active virtual network management prediction: complexity as a framework for prediction, optimization, and assurance," in *Proc. DARPA Active Netw. Conf. Exposition*, San Francisco, CA, May 29–30, 2002, pp. 534–553. ISBN 0-7695-1564-9.

[2] S. F. Bush and A. B. Kulkarni, *Active Networks and Active Virtual Network Management Prediction: A Proactive Management Framework*. Norwell, MA: Kluwer, Spring 2001. ISBN 0-306-46560-4.

[3] S. F. Bush, "Low-energy sensor network time synchronization as an emergent property," in *Proc. 14th Int. Conf. Comput. Commun. Netw.*, 2005.

[4] A. T. Campbell, H. G. De Meer, M. E. Kounavis, and K. Miki, "A survey of programmable networks (1999)," in *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, Apr. 1999, pp. 7–23. ISSN:0146-4833.

[5] L. Ming and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*. New York: Springer-Verlag, 1997. ISBN 0-387-94868-6.

[6] C.-K. Toh, M. Delwar, and D. Allen, "Evaluating the communication performance of an ad hoc wireless network," *IEEE Trans. Wireless Commun.*, vol. 1, no. 3, pp. 402–414, Jul. 2002.

[7] C. S. Wallace and D. L. Dowe, "Minimum message length and Kolmogorov complexity," *Comput. J. (Special Issue on Kolmogorov Complexity)*, vol. 42, no. 4, pp. 270–283, 1999.

[8] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

Stephen F. Bush (M'03–SM'03) received the B.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, the M.S. degree in computer science from Cleveland State University, Cleveland, OH, and the Ph.D. degree from the University of Kansas, Lawrence.

He is currently a Computer Scientist at General Electric Global Research, Niskayuna, NY. Before joining GE Global Research, he was a Researcher at the Information and Telecommunications Technologies Center (ITTC), University of Kansas. He has been the Principal Investigator for many DARPA and Lockheed Martin sponsored research projects including: Active Networking (DARPA/ITO), Information Assurance and Survivability Engineering Tools (DARPA/ISO), Fault Tolerant Networking (DARPA/ATO), and most recently, Connectionless Networks (DARPA/ATO), an energy aware sensor network project. He coauthored a book on active network management, titled *Active Networks and Active Network Management: A Proactive Management Framework* (Norwell, MA: Kluwer). He is an internationally recognized researcher in Active Networking and Algorithmic Communications Networking Theory with over 30 peer-reviewed publications. He explores novel concepts in complexity and algorithmic information theory toward applications ranging from wireless network management and ad hoc networking to RNAi sequence analyses and nanotechnology related research into networks of carbon nanotubes.