AUTONOMIC NETWORK MANAGEMENT: A PROPOSED FRAMEWORK

by

DANIEL BRYAN MARCONETT
B.S. (California State University, Sacramento) 2006

A thesis submitted in partial satisfaction of the
requirements for the degree of

Master of Science

in

Computer Science

in the

Office of Graduate Studies

of the

University of California, Davis

June 2008

Approved by _____
Chairperson of Supervisory Committee

_____

_____

Date _____

Table of Contents

Daniel Bryan Marconett
June 2008
Computer Science

Autonomic Network Management: A Proposed Framework

Abstract

A thesis presented on the varying issues involved concerning Autonomic Network Management in general, and applied to multihop wireless networks in particular. A management framework is proposed for addressing network management concerns through the use of a feedback loop for state observation, virtual hierarchy for state aggregation, and a control loop for management policy dissemination and actuation. Distributed network agents, residing within individual nodes, are proposed as the main actors for actuating management policy within the network. Cross-layer optimization techniques, utilizing reinforcement learning, are proposed for improving network performance, reliability and understanding. Further, the proposed framework allows for the automation of these management tasks through self-monitoring, self-healing, self-configuring, and self-optimizing control loop processes. By leveraging varying analytical tools, from data mining to scientific workflow and machine learning techniques, the need for a multifaceted approach to autonomic management which is cognizant of the state of the network, no matter what the underlying physical technology or architecture may be employed, is conveyed. Further, this cross-disciplinary approach attempts to leverage the existing work form varying fields, such as Operations Research, Artificial Intelligence, and Control Theory to validate the potential benefits of autonomic network management.

LIST OF FIGURES

Number *Page*

ACKNOWLEDGMENTS

The author wishes to express sincere thanks to Prof. S. J. Ben Yoo for steadfast advising in the area of network management, as well as for providing much academic opportunity and intellectual freedom. Additional thanks go to Prof. V. Rao Vemuri for chairing the committee, as well as the other members, Prof. S. Felix Wu, Prof. Xin Liu and Prof. Bertram Ludaescher, for providing valuable input for this work. Finally, this thesis is dedicated to the author's future wife, who has been an unbending source of support, encouragement, and love for the past seven years.

I. INTRODUCTION

Motivation

Today's network technology environment is varied and diverse. Different technologies, such as Optical Fiber, DSL, 802.11Wireless LANs and Wi-Max, are deployed in varying localities, sometimes simultaneously, and expected to work seamlessly through the use of the TCP/IP protocol stack. While TCP/IP has allowed for substantial interoperability among such technologies, there remain limitations. TCP's incorrect inference of link congestion as the sole source of packet losses in wireless networks is one such example [1,2].

Though we focus mostly on multihop wireless technology in the ensuing discussion, we do acknowledge a few examples in a wirelined and heterogeneous environment to point to the need for fully automated network management to comprehend the underlying technologies used, their inherent limitations, so as to fully maximize the optimizations performed in terms of policy-based management decisions. Such a feat is by no means trivial to engineer. However, this proposal endeavors to present a conceptual framework, coupled with experimental design and preliminary analysis, in order to illustrate the feasibility and utility of this approach, namely taking the human out of the decision-making loop in network management.

By leveraging cross-disciplanary concepts, such as Machine Learning, Knowledge Management, and Data Mining, we propose a holistic approach to managing complex network systems, especially those which may be prone to frequent topological flux. Dynamic wireless mesh networks provide just such an environment to test these concepts.

Wireless Mesh Networks

Wireless Mesh Networking is a promising access network technology which provides relatively inexpensive network connectivity by leveraging existing IEEE 802.11 radio technology. The minimal infrastructure required for multi-hop wireless networks is ideal for rapid deployment of network communications capabilities in scenarios such as emergency disaster response, business conferencing, remote sensing, and internet access for underdeveloped communities.

A wireless mesh network is characterized as a self-configuring and self-healing mesh of wireless nodes cooperating to create or extend an IP network through the multihop relaying of data packets. At times, the topology of this network can change arbitrarily due to the mobility of nodes, which may also be power and bandwidth-constrained [3]. Wireless mesh network architectures can generally be divided into three categories; infrastructure-based, client-based, and hybrid. Infrastructure-based refers to an architecture in which wireless routers form a self-configuring backbone to which other client nodes connect. Client-based is a peer-to-peer architecture where every node functions as both a router and a client in the multihop network. The hybrid architecture is a combination of the previous two scenarios; i.e. some nodes may be designated strictly as router, while other nodes may be strictly functioning as clients which run user applications, and yet other nodes may function as both clients and routers to increase the bandwidth and robustness of the network [4]. Figure 1 illustrates this hybrid architecture, where in addition, some nodes may be fixed and others mobile (which may change over time). We consider this scenario will be the most prevalent mesh network architecture in the future, and therefore target the discussion of this paper on network layer management with this scenario in mind.

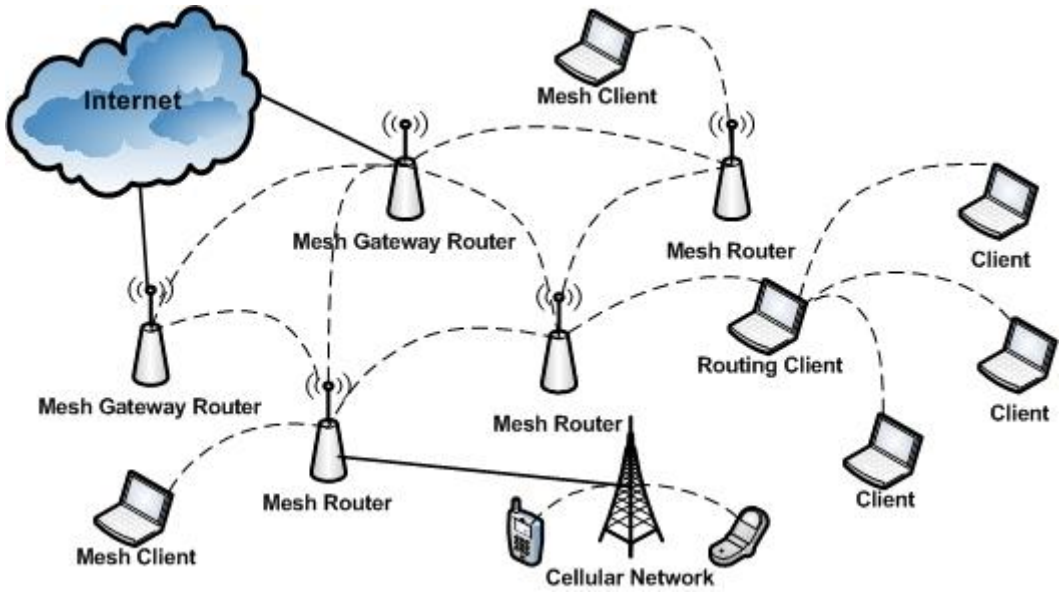


Figure 1 - Hybrid Wireless Mesh Network Architecture

II. PROPOSED FRAMEWORK

Autonomic Workflow

The general idea behind autonomic or cognitive networking is network state awareness [5]. Cognitive network element(s) follow an operational workflow which entails observation, policy modification, enacting such policies, and further observing the effects of these modifications [6]. In the case of a cognitive network layer, policies address fundamental questions, such as, "*To which outgoing link should node i forward the current data packet, and what is the result of this forwarding decision upon network QoS?*"

High level goals are formulated into policy, which in turn dictates specific actions on the part of the network management entities, which then observe the effects of these actions in the form of quantifying network performance metrics. The uncertain, dynamic environment of a hybrid mesh network, where certain nodes are mobile and existing link quality is constantly in a state of flux, is an ideal motivation for the development of a cognitive network management architecture. In general, cognitive networks involve cross-layer adaptability based on network performance, and are ultimately driven by end-to-end QoS demands [7]. The general workflow which we propose for the iterative management process is:

1.) Observe: Gather state information pertaining to the current view of the network.

2.) Refine: Perform preliminary analysis on observed metrics (visualization, data mining).

3.) Analyze: Determine the meaning of said observation by using reinforcement learning to discover optimal environmental configuration.

4.) Formulate: Create management policy based upon observational analysis over time.

5.) Actuate: Translate policy formulations into concrete actions, such as load balancing and protocol tuning.

The above five step workflow naturally lends itself to an architectural decomposition which is conveyed in Figure 2, where State Refinement occurs in the 'Knowledge Management System,' Analysis occurs in the 'Network Analysis System,' and Formulation occurs in the 'Policy Formulation System. Each of the three components are discussed in detail in subsequent sections.

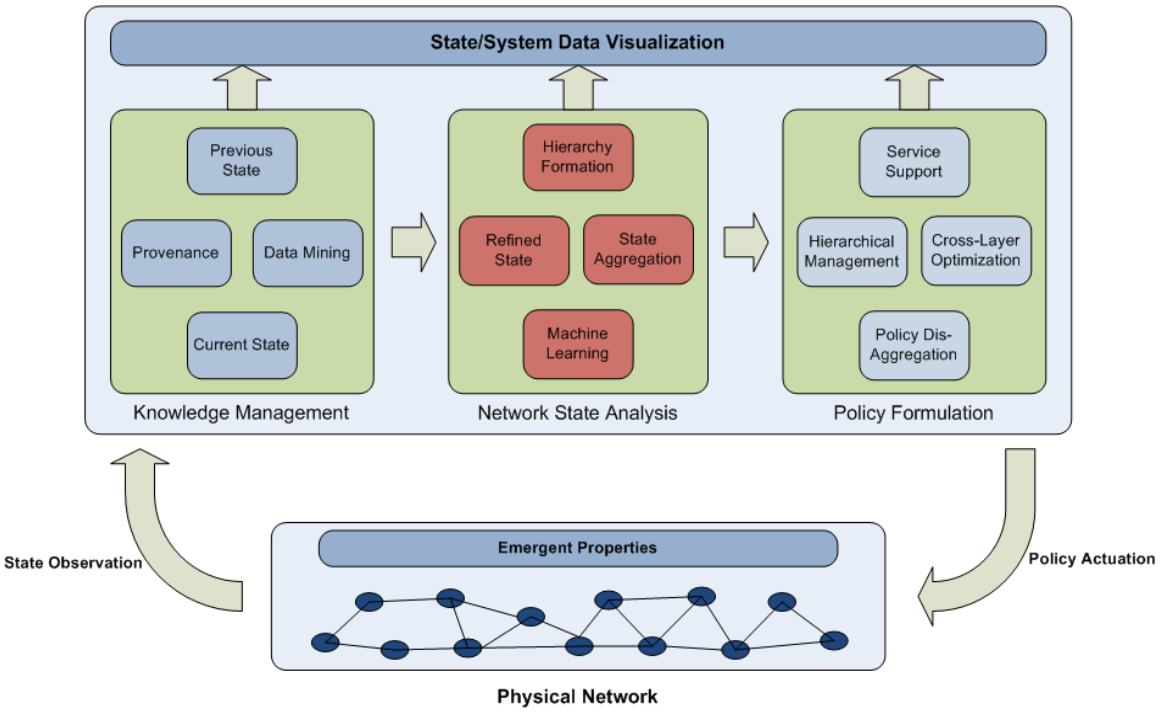


Figure 2 - Autonomic Network Management Architecture

Autonomics and The Management Plane

One of the main ideas behind the push for widespread research and deployment of WMNs is the furthering of ubiquitous computing. The creation of an unteathered wireless environment where users may seamlessly interact with surrounding access technologies, regardless of mobility, and with little or no user configuration. Key to the enablement of this operational scenario is an intelligent cross-layer management framework which maintains context in terms of client mobility and QoS requirements [8]. In the following paragraphs, we illustrate some existing network management schemes which employ situationally-aware mechanisms for policy-based management in dynamic networks.

In essence, the management framework should not only be cognizant of the requested Guaranteed Level of Service for end-users, but also work to decide whether or not such requests are feasible, and do so in a timely manner. In the current discussion, we are focused on the network-layer enablement of such an approach, while mindful of the cross-layer concerns which will eventually come into play when designing an optimal management framework. Presently, there are a few proposed frameworks which attempt to address WMN management from a somewhat cognitive perspective, and we discuss them below.

DRAMA

The *Dynamic Re-Addressing and Management for the Army* (DRAMA) [9] architecture is a policy-based management framework for highly mobile mesh networks. Identifying the need for effective link management in highly volatile, ad hoc communication networks with scarce bandwidth, this framework attempts to address this concern by designating varying levels of Policy Agents to manage the network in various domains. Policy Agents come in three forms:

- *Local Policy Agents* (LPAs): Responsible for individual node management.
- *Domain Policy Agents* (DPAs): Responsible for domain management and LPA aggregation.
- *Global Policy Agents* (GPAs): Charged with the dissemination of policy to as well as management of DPAs.

DRAMA allows for increased management scalability through the implementation of a two-level hierarchy. GPAs ensure communication of management policy to DPAs, which are charged with ensuring LPAs carry out the required actions.

Policies are defined by the authors as having four components; *Events*, *Conditions*, *Actions*, and *Scope*. Events are defined states which can occur during normal network operation. In the case where a defined Event occurs, the necessary policy evaluation takes place, and if deemed necessary, an action is specified. Conditions are defined as binary determinations of whether a particular Policy Agent is responsible for carrying out a specified action, such as a route determination or, at a higher level, resource brokering between domains. Actions are the tasks performed by the agents in response to Condition evaluations. Scope is extent to which specified actions are applicable to Policy Agents, i.e. whether globally defined actions are to be carried out by DPAs, LPAs, or both, and which domain(s) are involved in the process.

At a fine-grained level, LPAs are responsible for reporting network management information, such as congestion levels and traffic loads, to higher layer policy entities. Periodic asynchronous reporting is facilitated through the use of the Yelp Announcement Protocol (YAP). YAP periodically polls lower level policy agents for management updates, which allows proactive policy modification in the face of a changing network environment.

Newly dictated configuration information is then distributed down to the lower-layer domains. This process has been designed to take into account the roaming inter-domain behavior of mobile nodes to provide seamless support for connection maintenance. Though the literature does not propose a specific RL model for policy determination, such a mechanism can easily be integrated into the high level and/or local policy management determination. Further, this framework conveys the essential needs of any effective solution, namely an effective information reporting and adaptation mechanism, and a reasonable degree of scalability.

SPIN

A more cognitively-based approach to hierarchical network management is evident in the *Software Programmable Network Architecture* (SPIN) [10]. The authors propose implementing a "Cognitive Plane" atop the more traditional Management and Forwarding planes. The Forwarding Plane is designated as monitoring and forwarding data packets within the network. Software is used to control guaranteed levels of service, when possible, and so-called "Heuristics-of-Service" (HoS) are forwarded to the management plane. These HoS messages contain network link measurements such as jitter, packet loss, and available bandwidth.

The Management Plane is charged with the control of forwarding plane devices as well as the aggregation of HoS information. Dividing the concerns of forwarding and management provides proven fault tolerance and increased availability. This capability is seen by the authors as crucial for implementation in a dynamic wireless mesh network. As in other schemes, the cognitive plane can be seen as the layer in charge of policy formulation and refinement. By leveraging the aggregated HoS information from the management level,

policy designations are fed back down the hierarchy to be enforced by the management plane, through the definition of specific actions in support of these policies.

At a more concrete level, network demand is measured through HoS observations. To this end, the SPIN architecture strives to enable intelligent decision-making on a per-flow basis. Network elements have the ability to both function as routers and label switches, depending on higher level requests. This functionality enables forwarding plane devices to actively change the characteristics of specific data flows in "near real-time" [10]. Further, the Management Plane can dynamically determine clustering associations and cluster size based upon the performance of these data flows, to better facilitate improved network performance.

Ad Hoc Innovation

The *Ad Hoc Innovation* management scheme is one of the more direct approaches to dynamic network-layer management [11]. The goal of Innovation is to allow the network layer to adapt to network conditions by means of network-wide consensus. Network-layer reconfigurability is a decision-based process (i.e. feasible or not) subject to the concept of "Diffusion of Innovations." Diffusion of Innovations is a social science model which studies how innovations are either accepted or rejected by the society at large. The five stages of the model are defined as *knowledge, persuasion, decision, implementation,* and *confirmation*. Knowledge entails the exposure of individual management agents to the observed innovation. Persuasion is the stage where the intelligent agent attempts to make a judgment about the proposed innovation based upon internal and/or external observation. Decision is the binary choice of the agent to either accept or reject the innovation. Implementation is the act of the agent putting the innovation to use. Finally, Confirmation is the reinforcement of the

decision to implement the innovation based upon observed performance, whether positive or negative.

This five-stage model is the basis for the adoption of certain network-layer management polices through the use of global consensus. The more agents adopt the proposed solution or policy, the more likely it will be adopted network-wide. Additionally, an interesting characteristic of this approach is that highly connected nodes tend to converge upon a decision faster, due to an increased amount of information at their disposal. More to the point, the innovation is typically parametric tuning of the routing protocol, and the decision is whether to change the value according to the proposed policy modification. However, the authors propose using the protocol itself as a tunable parameter, changing the routing protocol as network conditions dictate.

The scenario used by Forde et al to illustrate the need for dynamic protocol deployment is shown in fig. x. The scenario consists of two fundamental geographic areas; a narrow area with high nodal degree and moderate mobility, and a larger area with nodes more sparsely distributed, lower nodal degree, and relatively low mobility. When choosing between AODV, OLSR, and DSR, observations are passed through an EWMA filter to determine their contribution to the overall protocol choice. Of course, the determination of reasonable threshold values for these critical (and potentially costly) decisions is of paramount importance.

Further work in this area by Forde et al can be found in [12] regarding Self-Stabilization in WMNs with zone-specific routing, where different routing protocols are used on a per-zone basis as described above. Specifically, the problem addressed in this work is the conflict resolution which must take place when a mobile node passes from one domain to another.

Node migration is identified as a key concern in routing domain determination and maintenance. A key point of this work is the need for neighboring nodes to use the same protocol. This fundamental realization can pose a significant management concern in dynamic protocol configuration.

III. KNOWLEDGE MANAGEMENT

System Overview

In the domain of network management, there are several data, also referred to as measurements, which need to be collected periodically to ensure an accurate depiction of network state. Whether looking at topological configuration, average load, or the makeup of route tables, the data can have complex relationships with each other that might not be readily apparent to human operators. To fully realize the vision of autonomic network management, the concept of the Management Information Base (MIB), the central respository for network state information, must be re-engineered to better facilitate knowledge discovery for more powerful network management and improved performance [13].

Existing schemes for network management, especially those designed for knowledge management in dynamic networks, i.e. wireless mesh, have professed the need for metadata and ontology management [14]. While the need for metadata and data dictionary can be easily envisioned, for the purposes of this discussion, we are mainly interested in the architecture of a proposed knowledge management system, and tasks which need to be carried out to extract knowledge from network state observations. We assume that such structured metadata and definitional constructs are handled within the MIB itself. Figure x depicts our proposed knowledge management architecture, discussed in greater detail below.

Provenance

When actuating management policy in such a complex envirionment as network systems, where actions can take place, from packet transmissions to route determinations, at varying

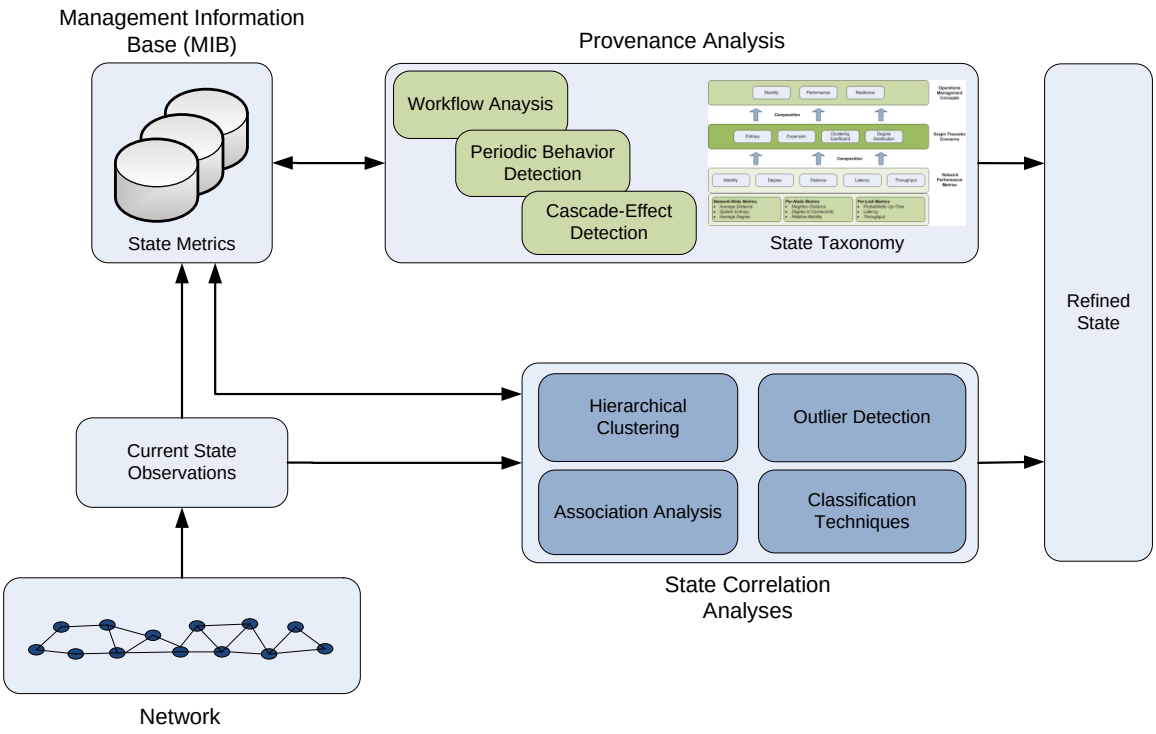


Figure 3 - Knowledge Management System

time granularities, it is important to be able to gain as much understanding about the environment as possible. This is necessary to avoid making policy decisions which may cause adverse interactions among control loops, and create less desireable performance results in the long run [15]. With this in mind, we introduce a provenance-based state capturing mechanism to provide better understanding of the complex interaction engendered by learning agents and the management policies they enact.

Node ID	X Position	Y Position	Speed	Degree	Avg. Util.	Rank
1	564.323	432.333	13	4	0.443	3
2	43.393	444.121	10	2	0.899	2
3	690.234	124.495	4	6	0.103	1
: :	: :	: :	: :	: :	: :	: :
N						

Figure 4 - Simple Provenance Schema

Provenance can be thought of as a detailed trace of execution in a program or system. More generally, it is concidered the history of how something came to be [16,17]. In the context of autonomic network management, we are concerned with the specific affects of our management decisions upon the network system, and gaining understanding for more optimal decisions moving forward. As such, we have devised a provenance analysis schema to serve as a simple MIB implementation.

The above table represents a specific snapshot of the network environment, and the MIB contains one of these for each observation made. Each node in the network is assigned a unique identification number. The X and Y cartesian coordinates of the node are captured as well, to reflect movement, or lack thereof over time. Instantaneous speed is also captured, along with degree of connectivity and average link utilization for all links adjacent to the node of interest. Finally, the rank, determind by a virtual hiearachical topololgy clustering scheme (discussed later) is noted to track premotion and demotion of nodes over time. Essentially, the

higher the rank, the more global knowledge a node has of the state of the network, as well as responsibility to formulate management policy based upon this knowledge.

Within the simulative framework, we provide query mechanisms to allow for further understanding of network behavior over time, with respect to such parameters as nodal region, nodal rank, and the average of any parameter which has been captured in the provenance model, such as utilization and speed. Further, we provide for a start and end time for the specified query so as to vary the epoch of interest for analysis at differing temporal scales.
find_avg(start_time, end_time, parameter, granularity)

- Determines the average value of a particular parameter over the specified time window (end- start).
- Granularity parameter determines how many time steps between metrics are needed for this calculation (every 1 sec., 5 sec. etc).

find_region(start_time, end_time, node_id)

- Determines the "tightest" physical bounds on a particular node, with respect to where it has been between the start and end times. (returns corner coordinates of a bounding box)

rank_history(start_time, end_time, node_list)

- Returns a time-series plot of the rank of nodes listed in node_list, over specified time period.
- Plot is 3D surface if two or more nodes are in list.

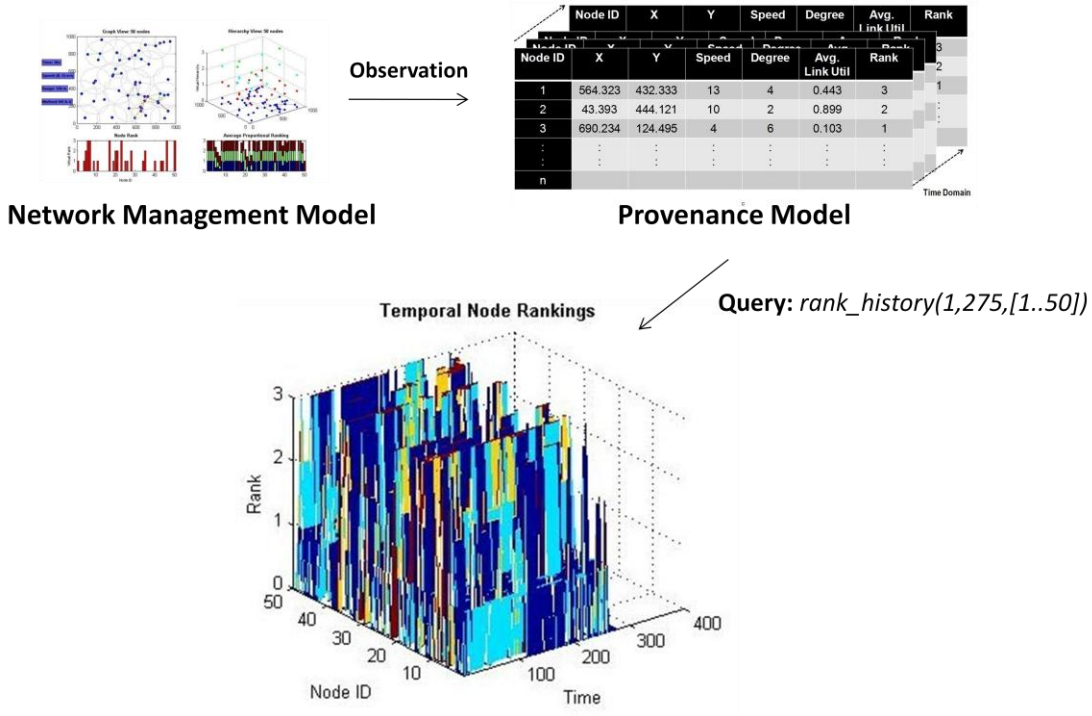


Figure 5 – Simulate-Capture-Query Workflow

Figure 5 depicts the simulation-capture-query workflow and illustrates a sample query based upon node rank, which allows for spatio-temporal visualization of nodal fitness over the desired timeframe.

While the intial work shown in this paper is a mere step in the direction of creating a fully viable decision support system, the intial results prove to be helpful and encouraging. Future work will entail the re-ordering of nodes according to logical adjacency, for a more clearer rank analysis over time. We hope this will allow for a more uniform and continuous surface when making queries to the Provenance system based on such metrics as rank history, as well as create a more cohesive view of network behavior. Logical adjancency can be found by utilizing

shortest path algorithms to create spatial adjacency over time when displaying data. Allowing for temporally sensitive data visualization at varying level of time granularity allows for a much more powerful mechanism by which to analyze and improve network management decisions. Furthermore, additional work will endeavor to explore data mining techniques which leverage this MIB schema to make correlative analyses which may not be so obvious to the human network manager, over space and time. This is clearly the direction a future Autonomic Network management framework must go in terms of knowledge management, otherwise sub-optimal performance will be realized, and automation will be seen as a hinderance rather than a useful tool for complex control systems.

IV. NETWORK STATE ANALYSIS

A key challenge in leveraging the full potential of WMN technology is the heterogeneous behavior of the nodes which make up the network, and how to manage the network in light of their typically unstable and unpredictable behaviors. Furthermore, the ability to localize certain behavioral patterns as well as aggregate similar nodes to more accurately identify emergent regional or network-wide behavior would give the management mechanism a more powerful means by which to formulate efficient policies. Additionally, it is important that such a management scheme scales with the number of nodes in the network.

A key facilitator of scalability in large mesh network deployments is clustering. Clustering is needed to leverage temporal and spatial relationships which arise from nodal dynamics. The dynamics dictate the state of the network in terms of topology and overall reachability. These dynamics in turn directly impact the performance of transport and management tasks taking place on the network. By more accurately characterizing network dynamics at node-level granularity, as well as higher-level emergent network behavior, and being cognizant of past network state, we are better able to predict more optimal clustering associations, namely choose the ideal clusterheads and reduce the overall number of clusters, as well as the overhead associated with maintaining these nodal aggregations.

In the ensuing section, we discuss the topic of network state analysis. The state of the network can be viewed in several fashions, depending upon the metric of interest. For example, network state can be thought of as the reachability of all nodes in the network relative to each other, or the speed with which each node is moving, or a snapshot of the current topology. An autonomic network management system must be able to reconcile

several perspectives and create an emergent, cohesive view of what is actually happening in this environment, so as to allow for effective policy management, whether it be load balancing, fault handling, or any other management task.

We discuss how virtual topological clustering can be used to scale network management decisions and allow for state information to be aggregated, such that the most 'stable' nodes end up being the ones to formulate management policy. Such nodes are indeed the ones promoted to the top of the hierarchy and are therefore responsible for high-level management decisions. Clustering allows for management decisions to be scaled, observational state to be aggregated temporally and spatially, and optimal policy to be fed back down to low-level actuating nodes who make the actual decisions in the network. We begin by discussing varying approaches to clustering in multi-hop wireless networks, followed by a description of reinforcement learning and its applicability to optimal clustering. Finally, we conclude with a discussion of how virtual hierarchical clustering aids in autonomic management of highly energetic network environments.

Clustering Techniques

Typically, in clustering schemes there are three types of nodes; clusterheads, cluster gateways, and ordinary cluster member nodes. Clusterheads are charged with the maintenance of their given cluster, ordinary nodes are adjacent to clusterheads, and as such are members of a clusterhead's cluster. Cluster gateways are nodes which are members of two or more clusters. Generally speaking, clustering techniques typically fall into one of six categories [18]:

- Dominating Set-based Clustering
- Reduced Maintenance Clustering
- Mobility-Aware Clustering
- Energy-Efficient Clustering
- Load-Balancing-based Clustering
- Combined Metrics-based Clustering

The ensuing discussion will provide a brief overview of these techniques with respect to the key metrics of interest and how they are used to form clusters the multi-hop wireless domain.

Dominating Set-Based Clustering

The Dominating Set-based approach to clusterhead election is based upon the need to identify a minimal set of key nodes upon which to optimize route formation and maintenance operations, thereby effectively forming a virtual backbone across the multi-hop network [19]. Furthermore, the calculation of this minimal node set allows for less frequent routing updates, so long as the dominating set chosen is also fairly stable.

Reduced Maintenance Clustering

Since the cost of cluster reformation is far higher than mere re-affiliation, determining clusterheads based upon decreasing the frequency and cost of reformation is a promising approach to cluster formation. Specifically, the Passive Clustering algorithm [20] is a clustering technique which does not require the generation of explicit signaling packets for state

awareness and cluster formation, but rather clusters are constructed on the fly, in response to user-driven data traffic.

By piggybacking nodal status on the data packets that a source node sends, it indicates its readiness to serve as a clusterhead. Thereafter, the neighbor nodes decide whether to stay members of their current clusters, or join with the newly announced clusterhead to form a new cluster. Moreover, the selection of gateway nodes (intermediate nodes which belong to two or more clusters), a minimal set of such nodes is chosen to reduce the need for reformation due to topology changes.

Mobility-Aware Clustering

Mobility-based clustering highlights the fact that mobility is the main factor which causes topological flux within a network. By taking mobility metrics into account, such as speed and direction, more reliable clustering can take place with fewer instances of reformation and re-affiliation. A large body of work concerning mobility-based clustering focuses on mobility prediction as a temporal means of efficient nodal aggregation [21, 22]. However, this cannot always hold true since current mobility does not necessarily translate into future mobility patterns of a given node.

Energy-Efficient Clustering

The motivation behind energy-consumption aware clustering is the fact that a clusterhead tends to expend more energy than standard nodes due to the increased load and management role inherent in its role. By monitoring the amount of energy dissipation which takes place at each individual node, better clusterhead elections can take place to increase the average

residual power available on a node-to-node basis and across the network. This in turn will increase the lifetime of a network which may be severely limited by the lack of available power, especially in the case of mobile nodes powered by limited battery capacity. Several existing schemes attempt to address this performance concern by limiting the duration a node can spend as a clusterhead, based upon the energy it has consumed thus far by minimizing the size of the dominating set upon which the clustering scheme is determined [23].

Load-Balancing-Based Clustering

Load-Balancing oriented clustering is typically concerned with limiting the number nodes in a cluster to minimize the traffic load on clusterheads [18]. When clusters are too large, the clusterhead can impede transport performance of the network itself. Conversely, when the cluster size it too small, the number of clusters network-wide increases, along with the overhead of maintaining the clustering structure.

Combined Metrics Clustering

Combined metrics clustering is typified by the Distributed Clustering Algorithm (DCA) [24], in which a weighted linear combination of a subset of the above mentioned clustering metrics is used determine the overall suitability of a node to act as a clusterhead. The various parameters are normalized and weighted according to performance, as in Equation 1,

$$s_i = w_1 c_1 + w_2 c_2 + \ldots + w_n c_n \quad (1)$$

where i is the node ID, and n is the number of metrics under consideration for the nodal fitness calculation.

However useful it may be to incorporate one or more of these metrics into the general

clusterhead-determining calculation, it remains that the afore-mentioned techniques rely on static snapshots to perform spatial aggregation, or mere mobility prediction to address temporal concerns [25]. There is clearly a need to tune these calculations with the temporal state of nodes so as to better anticipate which candidates will in fact make the most ideal clusterheads in terms of long-term stability. We posit that reinforcement learning is a promising technique to achieve this goal.

Reinforcement Learning

While clustering is typically a technique used to scale routing in a large multi-hop wireless network, our goal is to leverage this technique for management of dynamic networks. By creating multi-level hierarchy and clustering based upon stability metrics, distributed network management is facilitated and varying views of emergent network behavior are apparent at varying levels of granularity, depending upon how the network state is observed and aggregated. Our particular clustering algorithm uses a network-distributed form of Reinforcement Learning to take note of a node's previous state and factor that knowledge into the current clustering determination.

Reinforcement Learning is a form of Machine Learning, characterized by the formulation of policy to achieve a particular set of goals. Reinforcement Learning problems are typically modeled by means of Markov Decision Processes (MDPs) [26]. The model is comprised of the set of potential environment states, S; the set of possible actions, A; the designated reward function, R: $S \times A \rightarrow R'$; and the policy which determines state transition, P: $S \times A \rightarrow \pi(S)$. The set $\pi(S)$ represents the set of functions over set S, which specify the actions required to

transition by means of action a ∈ A from state s to state s'. The instantaneous reward (S, A) is a result of taking action a while in a particular state.

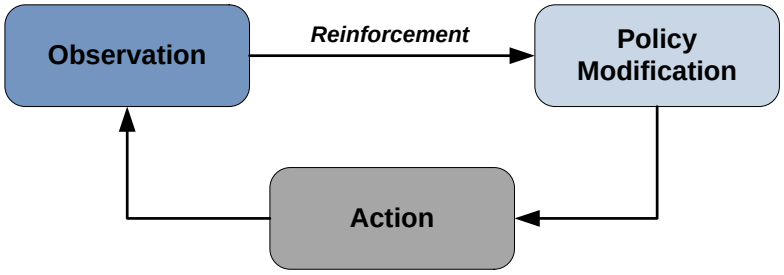


Figure 6 - The Reinforcement Learning Workflow

The specific form of Reinforcement Learning that our clustering algorithm employs is referred to as Q-Learning [27]. As a model-free reinforcement learning technique, Q-Learning is ideally suited for optimization in dynamic network environments. Model-free techniques do not require any explicit representation of the environment in which policy updates are formulated, and as such, can solely address parametric optimization to maximize long-term reward. This is a particularly important benefit of Q-Learning, since the problem addressed, namely the management of dynamic wireless mesh networks, is multi-dimensional problem with several state variables involved, which leads to a rather large potential state space.

Q-Learning

Q-Learning employs a weighted moving average calculation, to not only take note of recent policy success/failure as feedback, but also take into account the weighted average of past values observed, referred to as 'Q-values.' Q-value computation is performed via the following equation

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha \cdot Q(s',a') \quad (2)$$

Q(s,a) refers to the Q-value computed based upon state '*s*' and action '*a.*' In the current network environment considered, state is the observed behavior of nodes within the network, whether considering mobility, connectivity, nodal distance, or a The specific form of Reinforcement Learning that our clustering algorithm employs is referred to as Q-Learning [27]. As a model-free reinforcement learning technique, Q-Learning is ideally suited for optimization in dynamic network environments. Model-free techniques do not require any explicit representation of the environment in which policy updates are formulated, and as such, can solely address parametric optimization to maximize long-term reward. This is a particularly important benefit of Q-Learning, since the problem addressed, namely the management of dynamic wireless mesh networks, is multi-dimensional problem with several state variables involved, which leads to a rather large potential state space.

In the current network environment considered, state is the observed behavior of nodes within the network, whether considering mobility, connectivity, nodal distance, or a combination of parameters. Actions in this setting correspond to the decision of whether or not to employ certain nodes as clusterheads and include the one-hop neighbors of such nodes in the newly formed cluster. Variable '*r*' represents the instantaneous reward value which is derived from a measurement of the current environment based upon the current policy. The variable '*α*' is referred to as the learning rate, or the weight assigned to the current observation, between 0 and 1.

In our implementation, as conveyed by Equation 2, the learning rate also determines the weight coefficient of the previously observed Q-values, namely the '(1-α).' *Q(s',a')* is the new Q-value computed, which corresponds to the current action *a'* and the newly resultant state, *s'*.

By measuring Q-values over time, the relative fitness of a particular policy, or in our case, a choice in determining cluster formation, may be ascertained, and obligatory iterative modifications to this policy can be performed.

RL-based Clustering

Clustering techniques, which incorporate previous network state into the decision of node fitness for clusterhead election, mainly focus on mobility prediction [25,28] as a means of calculating nodal fitness. However, our RL-based clustering scheme leverages the weighted parameterization of the WCA algorithm, by combining a handful of node metrics which are relevant to measuring the stability of the node, and uses Reinforcement Learning as a means optimize this calculation by taking into account past stability values.

We dynamically tune the learning rate based upon relative mobility. This is accomplished by normalizing the speed of the node based on the max speed of all other nodes for a given iteration. This in turn becomes the current learning rate for this particular node, thereby facilitating a distributed learning mechanism which is both specific to that given node, but also take into account network-wide parameters of mobility for the normalization process. The goal is to allow the clustering management mechanism to leverage the emergent stability which exists within the network at any given time. Further, by taking into account recent nodal behavior through the use of a weighted moving average, we are able to temper short term reaction to network dynamics with a long term perspective of overall optimization and long-term reward.

The adaptive learning rate updating is a function of the relative stability which may exist in the network. Stability can mean several things in this sense; degree of connectivity, residual

power levels, relative mobility, etc. Since we are primarily interested in the physical stability of the network to facilitate adaptive nodal aggregation through clustering, we chose to use three metrics for the stability measure, and weight them according to their perceived importance; namely:

1.) c_i: the difference between ideal cluster member count and number of one-hop neighbors for each candidate node,

2.) d_i: the average distance of a candidate clusterhead with all of its one-hop neighbors

3.) s_i: the speed of each node

These metrics are then combined to form the following node fitness equation,

$$f_i = w_1 c_i + w_2 d_i + w_3 s_i \quad (3)$$

where f_i is the fitness index for node i, and weights w_1,w_2, and w_3 are determined before-hand (for simulative purposes, we set each equal to 1/3 to reflect the uncertainty of the relative importance of each measurement).

Our algorithm then takes a weighted moving average of these fitness values on a per node basis, relative to the state of the current node with respect to its mobility. In particular, the learning rate 'a' of this calculation is determined by how fast the node is currently moving, divided by the fastest node speed in the network.

$$f_i = (1-\alpha) f_i + \alpha \cdot f_i' \quad (4)$$

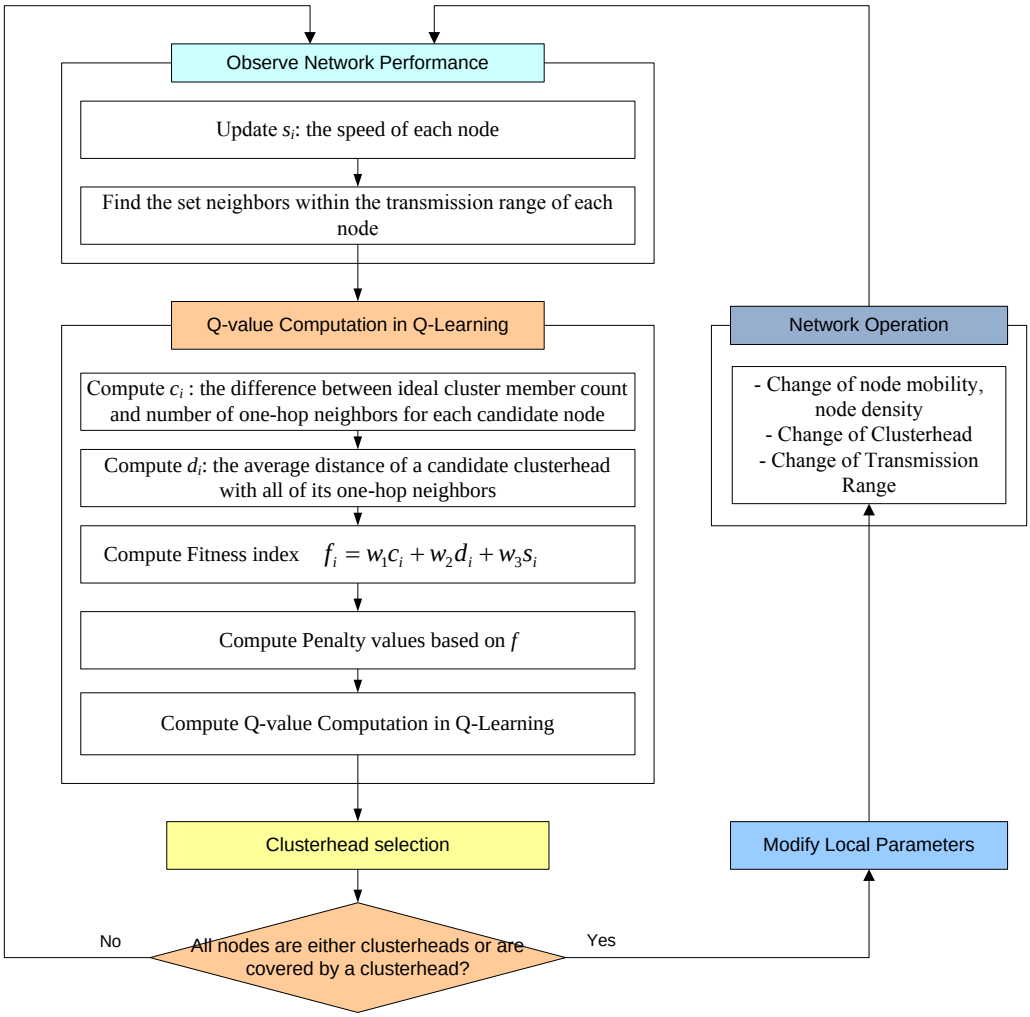


Figure 7 - RL-Clustering Workflow

By leveraging the above behavioral metrics in a weighted fashion for periodic clusterhead election, as shown in Equation 3 above, we are able to take a more holistic view of network stability. Further, we optimize our reinforcement calculation by taking into account recent stability indices for each node, weighted by the instantaneous mobility of the node itself (see Equation 4). The fitness value, becomes, in effect, the penalty values for the Q-Learning calculation, in that the lower the fitness value, the higher the chance a node will become a clusterhead. Clusters are then formed by comparing the relative fitness values, and choosing

the lowest value as a clusterhead, as well as its one hop neighbors to form a new cluster (see figure 3 for illustrative algorithmic workflow). The clustering algorithm executes as follows:

1.) For each iteration of the algorithm, if a node is already chosen as a clusterhead, or covered by a clusterhead, then it is no longer considered to be elected a clusterhead.

2.) Otherwise, each node's fitness metric is updated relative to the moving average of its past stability metrics (see eqn. 4)

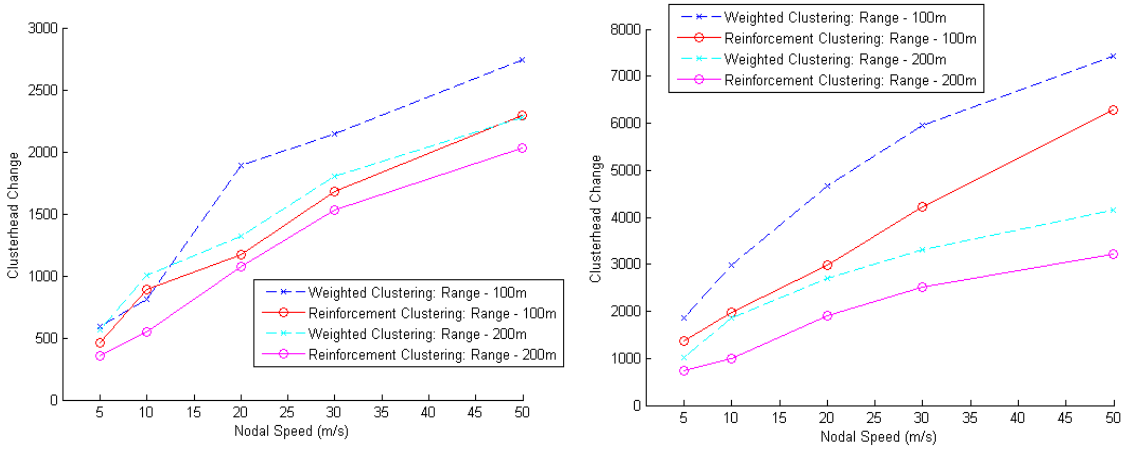
3.) From the remaining candidate clusterheads, the node with the *lowest* fitness value is chosen as a clusterhead, and its one-hop neighbors as clustermembers.

4.) This algorithm persists until all nodes are either clusterheads or are covered by a clusterhead, at which time the election process ends its current iteration.

By taking past node behavior into account, we prevent the clustering mechanism from making hasty decisions about the fitness of a node based upon instantaneous fitness measurements, thereby decreasing the overhead needed to maintain our clustering scheme.

Experimental Results

We use the stability of the clustering structure itself to determine the overall performance of the clustering scheme, with respect to such metrics as number of clusterhead changes and total number of clusters, as a function of maximum nodal speed, nodal density, and transmission range. To simulate the environment, we use a Matlab model, which employs random waypoint mobility for changing node mobility vectors. The speed and direction are uniformly chosen



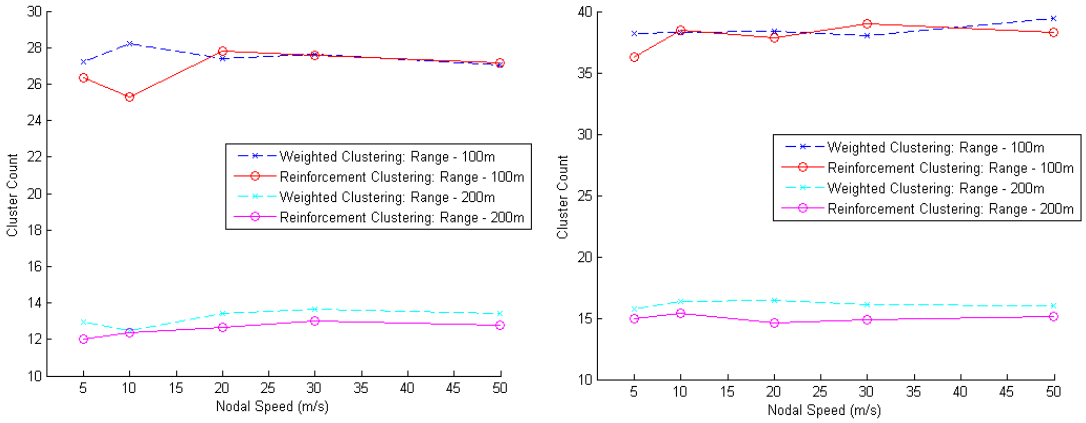
(a) 50 nodes (b) 100 nodes

Figure 8 - Degree of Clusterhead Change per Clustering Technique

with a max pause time of 5 seconds and max speeds of 5, 10, 20, 30 and 50 m/s for each simulation. The simulation duration is set at 200 seconds to give sufficient time for topological flux to take place due to mobility. The physical area of the network is a typical 1000m x 1000m, with node counts of 50 vs. 100 nodes. Node counts are varied for the same geographic area to observe the effect of nodal density on clustering stability and decision making. Further, transmission range changes from 100m to 200m for the first and second set of simulations respectively. This too is done to observe the effects of cluster expansion on the stability of cluster formation and maintenance.

Clusterhead Change

The results obtained with respect to cluster change are promising, as shown in Figure 4. Notice a decrease in the number of cluster reformations of approximately 15-20% for Reinforcement Clustering over the WCA clustering implementation. This result proves intuitive, in that, our algorithm is better able to adjust the rate at which network stability is



(a) 50 nodes (b) 100 nodes

Figure 9 - Average Cluster Count per Clustering Technique

taken into account when determining nodal fitness values. Rather than simply calculating the connected dominating set at a particular time instant, Reinforcement Clustering weights the current state observation of nodal stability by the current mobility of that node, and tuning the learning rate accordingly.

By using this relative stability calculation, we effectively reduce the number of cluster reformations which take place. This in turn leads to more efficient management of the network, but decreasing the need for explicit cluster management messages, decreasing nodal energy consumption, and increasing the accuracy of network state measurement. The distributed network management agents may then take advantage of this more accurate state representation to make more optimal policy decisions, in terms of routing, resource provisioning, and topology management.

Cluster Count

Though there is a fairly consistent reduction in the number of clusters required, as shown

in Figure 5, when using Reinforcement Clustering over simple WCA, the decrease is too minimal to be considered significant. We attribute this result to the fact that both algorithms use similar methods of cluster election. Furthermore, while Reinforcement Clustering takes past behavior into account, it does not necessarily work towards lowering the overall cluster count, but rather creating the most stable cluster configuration, as evidenced by the decrease in instances of cluster reformation, discussed in the previous section.

Implications of Range and Nodal Density

By doubling the number of nodes for a given geographic area, we effectively increase the nodal density of the network. We witness a more consistent decrease in the number of cluster reformations over the lower density 50 node count. The increased transmission range will usually improve cluster stability due to the increased spatial locality. Moreover, increased nodal density appears to also have a beneficial effect on clustering performance and stability. Regardless of which metric is considered, our Reinforcement Clustering technique yields a decrease of cluster reformation, but the amount of improvement appears to decrease with the increase in node density.

Concluding Overview

By taking advantage of emergent stability trends within a given deployment of nodes, we display that using reinforcement learning, more effective temporal and spatial aggregation of nodes into virtual clusters can take place. Further, by dynamically tuning the learning rate of the RL clustering algorithm with respect to the stability of individual nodes relative to its

peers, we are better able to assess the fitness of an individual node, in terms of whether or not it should carry out the duties of a clusterhead.

While many challenges remain insofar as reinforcement-learning-based clustering is concerned, our preliminary results yield promising indications that policy-based reinforcement learning can indeed be used in mesh environments with varying degrees of dynamicity. Future work will look at the tuning of weighted stability metrics for fitness value computation, depending upon the state characterization of the network at any given time. Additionally, using hierarchical management for varying levels of policy formation will be an important step to introduce scaling in large deployments., This in-turn will allow for a more efficient means to manage such dynamic networks through building management hierarchy, and decreasing the overhead necessary for maintaining the management structure.

V. Policy and Actuation

The key purpose of this reasearch, whether observing the state of the network or analyzing and extracting knowledge from these observations, it to effectively and efficiently tune, or actuate, the network for optimal performance. The formation and actuation of policy is sometimes referred to policy-based network management [29], and can take different forms depending upon the metric(s) of interest. In the next section, we provide a view of cross-layer design for our management framework which lays out explicit metrics to observe, as well as components to tune. We then discuss our preliminary work tuing the AODV routing protocol to achieve substantial performance improvement over the current implementation.

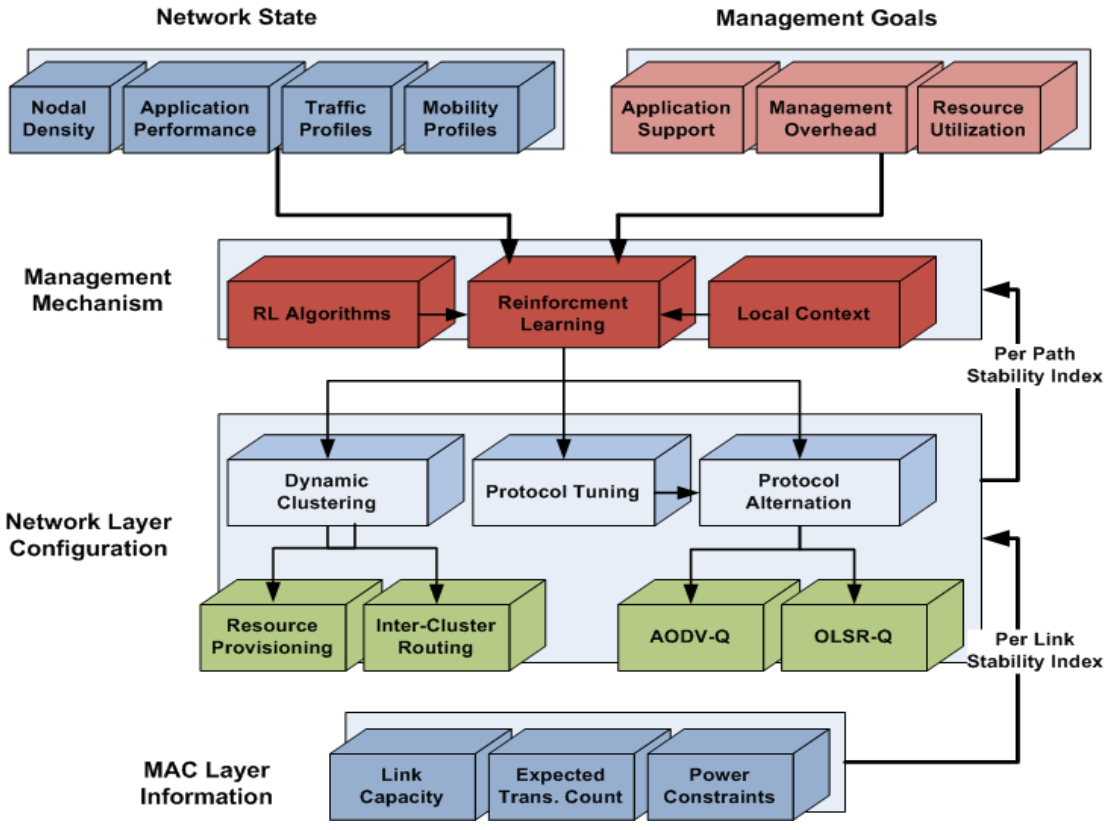


Figure 10 - Cross-Layer Interaction Model

Network Layer Solutions

Quality-Aware Routing

Fundamental to adaptive routing is the identification of relevant performance metrics. The modification of the routing protocol itself is determined by the observation and analysis of these metrics of interest. Quality-Aware Routing (QAR) [30] is a non-RL routing scheme which attempts to address this issue by using two new parameters, modified expected number of transmissions (mETX) and effective number of transmissions (ENT), to reduce the packet loss rate in wireless mesh routing [30]. The mETX is obtained by changing the previously used ETX metric to take into account the variability of radio-channel conditions, rather than the average channel behavior. A more responsive ETX is generated by calculating the combined expected value of both the average and variance of the packet error probability. The ENT considers both the overall network capacity, as well as the higher layer protocol end-to-end loss-rate concerns. The value of ENT is defined as the sum of the average bit error probability of the channel and a so-called strictness coefficient times the packet error variance for that channel, referred to as the log ENT. Specifically, a given link is said to satisfy the ENT constraints if the calculated value is below the log value of the threshold of retransmits for the underlying link-layer protocol.

Routing in QAR is performed by modifying existing reactive routing protocols (e.g. DSR, AODV) to utilize mETX and ENT values during path selection. Probe packets are periodically sent to measure the ETX of a given link. Each node sends ten probe packets to sample the link loss rate, which is estimated by using an exponentially weighted moving average (EWMA) filter. The weight of the EWMA is halved every five sample iterations. The authors specify two routing approaches, the first being the periodic calculation of mETX for

each link and assignment of the mETX value as the corresponding link cost. Standard cost minimization routing is then employed. The second approach utilizes the *log(ENT)* values, compared to *log(M)*, where *M* is the retransmit limit of the link-layer protocol. An infinite cost is assigned to links which have a *log(ENT)* value greater than *log(M)*. The ETX value is assigned as the link cost to all other links.

Q-Routing

The first RL-based routing scheme to employ Q-Learning is the Q-Routing protocol [31]. In this approach, Q-values are used to perform to actual routing, and the reward (or penalty) values correspond to the time needed to transmit a packet over a particular link, including the queuing delay before transmission. The essential question of the Q-Routing algorithm is which outgoing link to forward to current packet to, based on perceived potential delay and actual observed delay. Actions in the form of outgoing link determination will result in transmission and queuing delay, which are also the chosen penalty values.

The following notation is derived from [31]. Let *q* be the amount of time packet *P* spends in node *x's* queue, and let *s* be the transmission delay incurred by transmitting the packet over the chosen link. If node *x* is the current node, its decision of which outgoing link to forward *P* is contingent upon its own estimate, $Q_x(d, y)$, which represents the total time estimated by node *x* to deliver *P* to destination node *d* via *x*'s next hop neighbor, node *y*. Upon arrival of *P* at node *y*, *x* will receive immediate feedback in the form of:

$$t = \min_{z \in neigbors\_of\_y} Q_y(d, z)$$

where *t* is the time estimate of node *y* forwarding *P* to its estimated optimal neighbor node *z*. Node *x* will then revise its estimate of delay with the following equation:

$$\Delta Q_x(d, y) = \eta(q + s + t - Q_x(d, y))$$

where η represents the learning rate, or the degree to which new observations are factored into the cumulative Q-value. The authors recommend a value of 0.5 for this parameter. An optimal forwarding policy, *Q*(s,a)*, is determined by computing the EWMA of successive Q-values over several iterations, at each local node. The emergent behavior of these local agent policy modifications is that of a somewhat globally optimal routing policy, which may forgo short term and relatively high valued rewards in favor of longer-term optimal reward values.

Using Bellman-Ford shortest path determination, the authors found that Q-Routing significantly out-performed static routing schemes which merely take into account shortest path metrics. Further, the conditions of widely varying traffic loads tended to underscore the significant performance improvement of Q-Routing over standard hop-based static schemes. Although these preliminary results were with respect to simulation of dynamic traffic loads in wire-lined networks, there have been some promising modifications for implementation of Q-Routing in Wireless Mesh, as discussed below.

Least Squares Policy Iteration

A further enhancement to RL-based routing can be found in the *Ada*R routing protocol. AdaR uses the model-free reinforcement learning technique of Least Squares Policy Iteration (LSPI) to iteratively approximate reward values of chosen routing policies, much the way Q-Routing does. However, LSPI has been found to converge must faster upon a optimal route policy than traditional Q-Learning based routing [32]. LSPI approximates the Q-values for a particular routing policy decision. Moreover, these Q-value approximations are made through the use of a linear weighted combination of *k* basis functions [32]. Wang et al have

defined the Q-value approximating function as follows. Let $\hat{Q}^{\pi}$ be be the approximate Q-value for policy π. Then $\hat{Q}^{\pi}$ is obtained by

$$\hat{Q}^{\pi}(s,a,w) = \sum_{i=1}^{k} \phi_i(s,a) w_i = \phi(s,a)^T w$$

where w_i is the weight of the i^{th} basis function $\phi_i(s,a)$ in the above linear approximating equation. The purpose of the k basis functions is to model the known information of each (s,a) pair. Using the above equation, coupled with the information gained by probing the environmental sample space to obtain immediate reward values, the weight value w can be solved for. Therefore, a much smaller sampling of (s, a, s', r) tuples is needed than in the case of Q-Learning, to converge upon an optimal policy decision.

In the AdaR routing scheme, the current state s represents a particular node, and state s' corresponds to its chosen neighbor to which the current packet is forwarded, via action a. Though the protocol is inspired by routing in ad hoc sensor networks, it is entirely appropriate to consider its implementation for volatile and time-varying hybrid wireless mesh networks. The protocol considers four metrics of interest for QoS routing in each state-action pair (s,a):

1) $d(s, a)$: the hop count differential between s and s' to the destination.
2) $e(s, a)$: the residual energy of node s'.
3) $c(s, a)$: the number of routing paths which incorporate node s', which is determined by network state sampling.
4) $l(s, a)$: the estimated link reliability between nodes s and s'.

Basis functions for the (s, a) pairs are of the form $\phi(s, a) = \{d(s, a), e(s, a), c(s, a), l(s, a)\}$, and the values for each are normalized to a range of [-1,1]. LSPI is used to update the weight

values of the linear functions, based upon the network samplings. Weight updates are used to modify the current policy iteratively. The learning procedure takes place when packet forwarding occurs. When a packet is sent from s to s', it is "stamped" with a record $< s, a, s', \phi(s,a)>$, where $\phi(s,a)$ is the immediate reward estimated with regard to the current Q-value. After the base station receives a certain number of packets with observational data, the LSPI procedure is invoked to carry out weight updating and policy modification. After broadcast of the modified weight, w', each node adjusts its forwarding policy by choosing the action a which will yield the largest Q-value as per the modified policy π'.

Cognitive Packet Networks

A conceptually different approach to adaptive routing in ad hoc networks is displayed in the form of Cognitive Packet Networks (CPNs). The intelligence of CPN routing lies in the packets themselves rather than the network elements [33]. Specifically, smart packets (SPs) collect network performance data as they travel throughout the network. Upon arrival at a specified destination, an ACK packet is sent back to the source node containing the updated performance information. ACK packets update "mailboxes," which reside at each node they visit along the reverse path, with SP-observed network performance data. The SP's are routed throughout the network as a result of Random Neural Network (RNN) Reinforcement Learning algorithms which reside at each node. Neural computation takes place based on network status information received by the ACK packets. This scheme facilitates distributed observation of dynamic network topology. Each node makes local routing decisions based upon the user application QoS needs and network performance metrics which reside within the node as a result of SP and ACK packet information

interchange. QoS estimates leverage SP observations to optimize application performance. CPNs are designed to accept high-level user defined goals for optimal path routing, with regards to metrics such as power constraints, node mobility, and topological disposition. Network state awareness is achieved by incorporating the knowledge of SP discovery into neural computation. Weight tuning of the RNN takes place to improve the CPN attainment of stated goals. Weights are modified according to the observed success level of SP route discovery with respect to these QoS metrics.

With respect to route determination, each router contains a neural network corresponding to a specific source-destination pair which is currently active within the network. Each router autonomously determines the outgoing link for every incoming SP packet based on the corresponding RNN. At the neuronal level, the i^{th} neuron's level of excitation corresponds to the probability of the i^{th} outgoing link being utilized. The most excited neuron determines which outgoing link is used to forward a given packet. The specified QoS goal has a function value to be minimized, and the reward value is merely the inverse of this. The goal function can be derived from such concerns as loss-probability and delay.

Swarm-based Routing

A slightly different form of distributed routing management is found in a variation of the classic *AntNet* algorithm [34]. The Swarm-based [35] algorithm utilizes distributed RL agents to explore ad hoc networks to find multiple viable paths. The basic premise is that while Ants in nature are simplistic and possess limited intelligence individually, in the aggregate, colonies are able to effectively locate resources and adjust paths to these found supplies

dynamically based on localized awareness and distribution of this awareness.

The Swarm algorithm uses network probing, in the form of "ant" packets to establish optimal forwarding policy. Specifically, there are three types of packets:

- *Regular Forward Ants*: At regular intervals, a node will send forward ant packets to a randomly chosen subset of network nodes. These packets contain the fields of *SourceIP, DestIP, NextHopIP, PathStack,* and *HopCount*. The PathStack contains the IP addresses of the nodes traversed along the way to the destination.
- *Uniform Forward Ants*: To alleviate the concern of *Regular Ants* only discovering sub-optimal paths, nodes also periodically generate *Uniform Ants* which are not routed to any particular path, but rather randomly choose their next hop path based on a uniform probability distribution.
- *Backward Ants*: Upon a *Regular* or *Uniform Ant* reaching a destination, it is destroyed and a *Backward Ant* is created. The *Backward Ant* obtains the *PathStack* and bandwidth information that the Ant Acquired and uses the *PathStack* to trace the reverse path back to the source node. Upon arrival at the source, the *Backward Ant* updates the source node's routing table with the bandwidth information it carries, along with dispensing this information along every intermediate node back to the source.

The QoS metric of choice for the Swarm algorithm is available bandwidth. Backward Ants update node routing tables according to the rules below. Let P_{id} be the probability of choosing node i as the next hop to reach destination node d. Further, let P_{jd} be the probability of choosing any neighbor other than node i as the next hop to reach d. The update rules are as follows:

$$P_{id} = P_{id} + f(BW)$$
$$P_{jd} = P_{jd} - (f(BW)/N - 1)$$
$$P_{id} = 1 - \sum P_{jd}$$

where $f(BW)$ is a function of the available bandwidth along that particular link, and the value upon which the reinforcement learning takes place. Note that $0 < f(BW) < 1$ and $f(BW)$ decreases proportionally with the decrease in available bandwidth. The goal of this metric is to ensure that the neighbor node with the largest available bandwidth is assigned the highest forwarding probability in the neighbor node's forwarding table. Thereafter, data packet routing takes place by merely choosing the node along the route path with the highest probability value in the routing table.

Collaborative Reinforcement Learning

A Swarm-inspired Collaborative Reinforcement Learning (CRL) route optimization approach is the SAMPLE routing protocol [36]. SAMPLE is a model-based RL algorithm which uses past and recent network status observations to create a model of the environment. CRL uses these models for each distributed agent to compensate for having incomplete network information when determining optimal routing paths, as opposed to Q-Routing, which a model-free RL technique and only learns after completion of a given action, rather than during it.

Each node in the SAMPLE scheme caches routes which are periodically broadcasted by local neighbors. In essence, each distributed entity is advertising the success level of actions taken, in the form of packet forwarding decisions. When node i receives an update from node j, in the form of $(Q_i(s, a), r)$, the state-reward information is cached for future reference.

A significant difference between Q-routing and CRL-based SAMPLE is the fact that cached routing information decays with time if not re-advertised among nodes, reflecting the tendency of topologies in MANETs and dynamic mesh networks to change frequently. Little or no advertisement of a particular path is indicative of low quality or non-existent links which comprise that path. After the passage of enough time, the path information is considered stale and removed from the table. The route cache decay rate is conveyed by the following equation:

$$Decay(r_j) = r_j \cdot \rho^{td}$$

where r_j is the last route advertisement received from node j and td is the amount of time elapsed since the advertisement took place. Further, the rate of decay can be determined by the parameter ρ.

The goal of the SAMPLE routing scheme is to maximize both the overall network throughput and packet delivery ratio, as well as minimize the number of transmissions required per packet. The events which are measured and monitored per-link in the network are:

1) c_A: number of unicast transmissions attempted.
2) c_S: number of unicast transmissions successful.
3) c_R: number of unicast transmissions received.
4) c_B: number of broadcast transmissions received.
5) c_P: unicast transmissions promiscuously received.

An estimate of the success of transmission over a given link is then obtained by:

$$E\left(\frac{c_S}{c_A}\right) = \frac{c_S + \alpha\beta(c_R + c_B + c_P)}{c_A + \beta(c_R + c_B + c_P)}$$

with *a* representing the strength of the belief that successful transmission will occur given transmission attempt has not yet occurred, but rather only successful receipt of a packet. The parameter β determines the weighting of received packets with respect to transmitted packets.

Link cost, denoted by $D_i(s'|s,a)$, is statically assigned, -1 for success and -7 for failure. These reward values are chosen with regards to the nature of link layer retransmission in the 802.11 protocol, namely the fact that up to seven retransmissions are allowed before a packet is truly lost. Success or failure of transmission is then reinforced at each node via the distributed learning equation

$$Q_i(s,a) = R(s,a) + \sum_i E_i\left(\frac{c_S}{c_A}\right)\cdot\left(D_i(s'|s,a) + Decay(r_j)\right)$$

with $R(s,a)$ signifying the reward value of the reinforcement function. Dowling et al define the forwarding selection as a Boltzmann-action selection process, where node *i* chooses the next hop from among its *M* neighbors based upon the following probability function:

$$P(s,a) = \frac{e^{-(Q_i(s,a))/T}}{\sum_a e^{-(Q_i(s,a))/T}}$$

where *T* dictates how likely suboptimal routing actions are taken, and is referred to as the temperature. Increasing *T* increases the degree to which route exploration in the network takes place, decreasing *T* restricts it. The authors note that a more dynamic network with more frequent topology changes should employ a higher temperature value than required for relatively static mesh networks.

Q-Routing vs. Cognitive Management

In the above described routing approaches, reinforcement learning is used to determine the actual routing paths in the multi-hop wireless network. While leveraging machine learning techniques to perform explicit route determination is can provide immediate network-layer improvement, this approach makes little sense from a resource efficiency perspective. Specifically, dynamic network management must concern itself with resource constraints and cross-layer knowledge to effectively manage future wireless mesh networks. The generation of explicit routing/management traffic, as in the case of CPNs and Swarm-based Routing, can adversely effect network performance, which is itself the primary concern the management scheme is striving to improve. Rather, the chosen management system should be as light-weight as possible, in terms of bandwidth consumption and computational complexity, to better facilitate increased performance in dynamic mesh deployments, where power, compute, and network resources are potentially scarce commodities.

Furthermore, it is quite clear from the literature that no single routing protocol will facilitate robust performance in all architectural scenarios [37]. Mobile and static wireless mesh networks have fundamentally different needs in terms of route management and routing traffic generation, as well as the support of application-layer QoS needs. When concerned with the creation of a general mesh management framework, a far better approach is to have access to a dynamic proactive and reactive routing protocol for relatively static and mobile mesh networks respectively. Depending on network characteristics, the appropriate protocol can be implemented and augmented in real-time by the intelligent network management agent(s) responsible for self-management and reconfiguration. The

creation of intelligent management mechanisms to dynamically reconfigure existing flat routing protocols is the first step to realize this goal, and we discuss our implementation and initial results in the subsequent sections.

AODV-Q

Rather than devising an entirely new routing protocol based solely upon link-layer measurement, as others have done, our work employs reinforcement learning to make an existing flat mesh routing protocol more adaptive to network dynamics. We chose the AODV routing protocol for augmentation based upon the body of work available regarding the protocol's performance, as well as its suitability for dynamic mesh environments. The overall research aim of our work is oriented towards cognitively intelligent network management. As such, the issue of creating an intelligent network-layer framework is the most compelling area in this nascent field. The first step in realizing this overall goal is the dynamic tuning of a reactive routing protocol, AODV.

Using AODV, network observation can be performed by monitoring the route_request/route_reply mechanism, which is built into the protocol itself. When a particular node attempts to establish a path to the desired destination, a route_reply message is received, indicating the success or failure of this request. Our management scheme leverages this information to infer network stability, rather than generating additional overhead in terms of network probing.

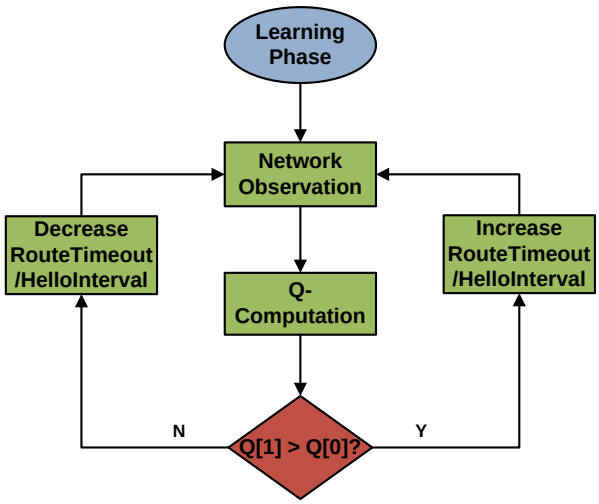


Figure 11 - AODV-Q Iterative Tuning Workflow

Specifically, we employ the Q-Learning technique to reinforce the success or failure of this process, and modify our management policy accordingly. This modification entails each node retaining two Q-values:

- Q[0]: *penalty Q-value for non-successful route requests*
- Q[1]: *reward Q-value for successful route requests*

The iterative updating of Q-values is determined by the following equation:

$$Q[i] \leftarrow (1-\alpha)Q[i] + \alpha \cdot r$$

For the series of simulations, α is set to a value of 0.2, to reflect the fluid nature of network stability in the hybrid network. For our experimentation, we chose the end-to-end delay of the route reply packet as the penalty of failure, and the inverse of the delay as the reward for success, corresponding to the reward/penalty function value '*r*.' The workflow of the Q-

Learning management mechanism is conveyed in Figure 11, and the algorithm is executed as follows:

1.) For a time period specified as the learning_phase_duration, increase of decrease the active_route_timeout and hello_interval with equal probability.
2.) Receive route_reply message, compute Q-value for success or failure
3.) If Q[1] >Q[0], GoTo step 4, else GoTo step 5.
4.) Stable Network: Increase active_route_timeout and hello_interval by a value of 2. GoTo step 6.
5.) Network Instability: Decrease requesting node's active_route_timeout and hello_interval by 2.
6.) Repeat (GoTo step 2)

We have employed this auto-configuration mechanism in a distributed fashion, by modifying the AODV protocol process model in Opnet. Our preliminary results reflect the fact that this management modification occurs in a distributed fashion, on a per-node basis. Each node implementing the enhanced AODV protocol model is also responsible for its own dynamic reconfiguration. Future work will entail creating a hierarchical network management framework in which the intelligent mechanism will be responsible for varying levels of knowledge aggregation and policy management, for increased scalability and contextual representation, rather than concentrating all policy decisions at the level of individual nodes. The ideas we propose, as well as the simulation results described in this paper, are meant to serve as a proof of concept for distributed self-configuring network management using Q-Learning for policy updating. As such, the following simulations and corresponding results

are meant to illustrate the performance enhancement achievable by employing this mechanism in wireless mesh networks.

Simulation and Observed Results

The work presented in this paper is a specific instance of our ongoing efforts towards adaptive intelligent network management. The simulations carried out and results observed pertain to the effects of Q-Learning based parameter optimization of the AODV routing protocol with regards to network mobility. The experiments have been carried out using Opnet Modeler network simulator using the wireless and mobile ad hoc network modules. The network environment is comprised of 30 nodes, each equipped with standard 802.11b radios with range of ~300m, deployed in a geographic area 1500m x 400m. For all scenarios, 4 nodes are static and the other 26 are mobile. The mobility profile used for the mobile nodes is random waypoint with a uniform distribution of 0-15 m/s. The pause times are varied between scenarios to study the effects of varying levels of network stability upon AODV and the dynamically tuned AODV.

For a measure of consistent performance, we used a single- source single-destination traffic demand with mobile source and static sink. A constant bit rate of 300kbps was used to model video traffic in each scenario. To observe the effectiveness of our dynamically managed AODV, video traffic was introduced into the network after 60 seconds and continued through the duration of the simulation, for all scenarios.

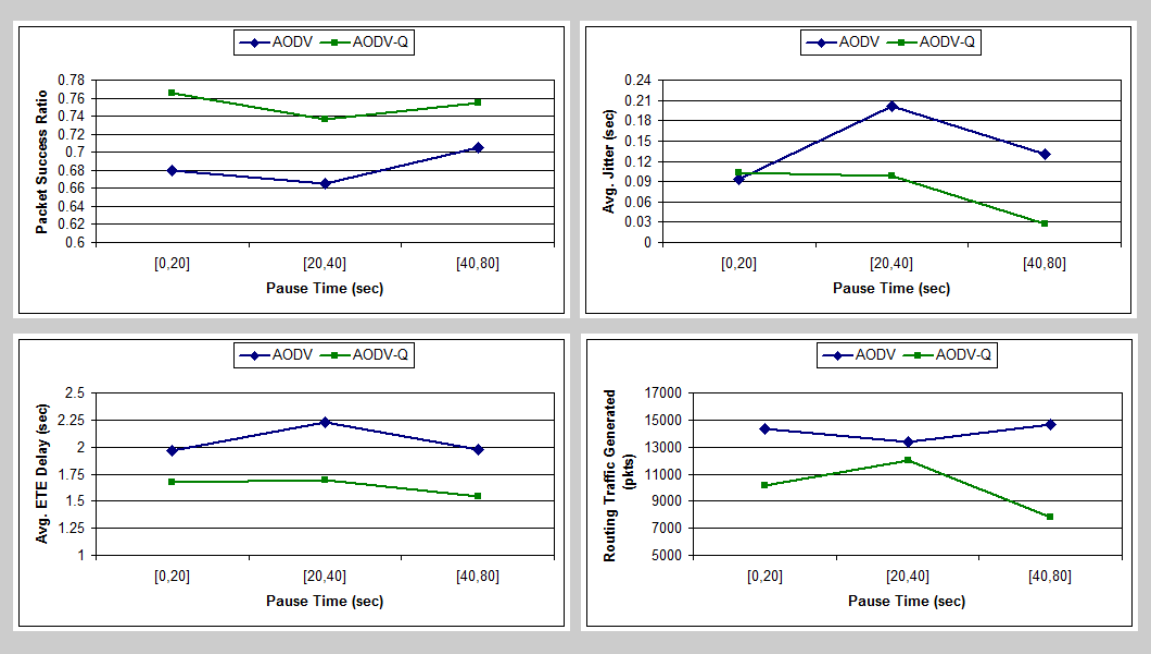


Figure 12 - AODV-Q Simulation Results

The main variance used between simulation scenarios was pause time distribution. The degree of mobility, and in particular the pause time, allowed for each node was used as a key indicator for the stability of each network scenario. Generally, the longer the pause time, the more stable the network becomes. Routes tend to be valid for longer periods of time, and nodes may therefore take advantage of the increased temporal locality by holding on to routing information longer.

Experimental Results

For analyzing the performance of AODV-Q in the above described scenarios, we chose four metrics of interest:

- *Packet Success Ratio* – The ratio of received video traffic packets to those transmitted at a constant bit rate for the traffic demand.
- *Average End-to-End Delay* – The average delay (in seconds) for data packets from source to destination for the video traffic demand.
- *Average Jitter* - The average jitter (in seconds) for packets transmitted for the video traffic demand.
- *Total Routing Traffic* – The total amount of routing traffic generated (in packets) for the duration of the simulation.

As displayed in Figure 3, our intuition of better overall performance with dynamic tuning over standard AODV was generally supported by the data. With respect to the packet success ratio, the simulations of our network environment typically yielded a packet success ratio improvement of about 6-8% over standard AODV. While the improvement in packet success ratio is not groundbreaking by itself, these results are promising when coupled with the fact that jitter was typically halved (except in the instance of a pause time between 0 and 20 seconds). We attribute this to the fact that our observed network characteristic was the delay of the route reply packet. Since the route reply nearly traverses the entire new path from source to destination (if not the entire path), the delay of the route reply is the most recent measurement of network stability available to the network layer when routing with the AODV protocol.

Moreover, the end-to-end delay of the video packets themselves was also decreased by over 20% when compared to standard AODV. The results did hint at a generally more accelerated decreasing trend for average per path hop count (as pause times increased) over that of standard AODV, but the differences, on the order of 0.2 hops per path, were too small

to be conclusive. Though average hop count is not necessarily the best metric to measure path fitness for a routing protocol, the ability of the routing protocol to find routes with both shorter hops and shorter delays does provide more evidence of improved performance.

There are limitations to this work however, such as the lack of explicit link-layer knowledge in terms of link congestion. Our approach does assume that decreased network performance can be rectified by tuning the network path discovery to be more proactive. However, this modification may be deleterious to network performance in the case of severe congestion, further exacerbating poor link performance by way of generating control packets more frequently. Though our management approach has been shown to decrease the aggregate network layer overhead (insofar as the number of control packets generated), future work will need to obtain cross-layer metrics, to allow the management agents a more empirical view of network performance in the correct context. This increase in knowledge acquisition may further increase the level of improvement realized by our solution.

VI. CONCLUSION

Though there are many facets to the reseach and ideas proposed in this proposal, there is a single clear unifying theme, that each comprises a part of a future Autonomic Network Management Framework. The logical decomposition of this framework has been discussed in preceeding sections, along with preliminary design and results in each. This final section will serve to highlight the proposed deliverables for a future Ph.D. dissertation in this area of research.

Ph.D. Dissertation Research Proposal

The proposed dissertation work falls into two main categories: simulative and implementation oreinted. Preliminary simulation work has already begun in both MATLAB[1] and OPNET[2] environments. Presently, the MATLAB simulation framework (see Appendix A for simulation interface) models nodal mobility (random waypoint model), dynamic link formation (as a function of distance and signal strength), source-destination routing (utilizing dijkstra's algorithm), virtual hierarchy (for state aggregation), and random link utilizations (based-upon defined source-destination pairs and routing information). Further work is to include:

1. Implementation of packetized traffic based upon standard statistical models for specific application profiles.

[1] The Mathworks, Inc.: http://www.mathworks.com

[2] OPNET Technologies, Inc.: http://www.opnet.com

2. Modeling of TCP/IP stack for cross-layer optimization study.

3. Multi-threaded execution to more accurately model network-distributed behavior.

4. Numerical analysis of the benefits of virtual hiearachy and the overhead cost inherent.

5. Implementation of additional Machine Learning techniques to optimize management decisions.

By utilizing MATLAB's built-in analysis tools, we can better analyze this complex environment to effectively see which techniques work and which ones do not. Further, by implementing additional Machine Learning techniques, such as Bayesian Learning or Temporal Difference Learning, we can analytically determine the costs and benefits of each technique.

While MATLAB is suitable for numerical studies of complex systems, OPNET provides a truly unique environment in which to simulate fine-grained network behavior, from the physical characterstics of signal propogation to packets sizes and header formats. Whereas MATLAB will be utilized for initial proof of concept experimentation, OPNET will allow for stage-1 implementation and emulation, allowing for a sanity check before implementation ina physical testbed. Further work in the OPNET domain is proposed as follows:

1. Creation of a cross-layer optimization framework which employs machine learning at varying levels of the protocol stack, and communicates through a standard API.

2. Creation of independent processes within a single node, where the number of processes corresponds to the number of virtual hierarchy layers in which the node

resides. Each process is independent but communicates with one another with respect to state observations and policy modifications.

3. Implementation of this Autonomic Management Framework with respect to wireless, wireline, and heterogeneous technology simulation scenarios.

Once sufficient results are collected from OPNET simulations, this will allow for testbed implementation of the proposed management framework.

The physical testbed implementation is key to successful adoption of our research ideas in "the real world." By collecting real-time results using the physical technology we are modeling, we can prove beyond any doubt the validity of the proposed framework. The physical testbed will be comprised of the following attributes:

1. Heterogeneous Interoperability: Ethernet, WiFi, and Optical network technologies co-existing and autonomously managed by the same framework model (where policies are formulated with the underlying technologies in mind).

2. Implementation of cross-layer optimization framework shown earlier, with a standard API devised for cross-platform, cross-protocol interoperabilitity.

3. Creation of a distributed knowledge management base where state information is collected and analyzed overtime. It is proposed that a Distributed Hash Table [DHT] be employed to track the location of varying state information to provide effective fault-tolerance in the event of network failures.

4. Employment of Team-based Collaborative Reinforment Learning techniques to further optimize policy formulation.

Once Autonomic Network Management techniques are shown to function optimally in a physical testbed environment, disussion can take place about forming a standard architecture with industry partners and networking consortiums alike.

Publications

Key to the success of any research is a history of successful publication, which contributes, if not forms the area of research and provokes meaningful input from other researchers. The following is a list of related publications, accepted and under review, by the author of this proposal:

Accepted:

1. Minsoo Lee, Dan Marconett, Xiaohui Ye, and S. J. Ben Yoo, "Cognitive Network Management with Reinforcement Learning for Wireless Mesh Networks," accepted for publication in Lecture Notes in Computer Science, Proc. IPOM07, 7th IEEE International Workshop on IP Operations and Management, October 31-November 2, 2007 in San Jose CA, USA.

2. Minsoo Lee, Dan Marconett, Xiaohui Ye, and S. J. Ben Yoo, "Autonomic Reconfiguration Management for Heterogeneous Wireless Networks using Reinforcement Learning," OPNETWORK 2007, 1538 Case Studies: Specialized Wireless Network Modeling, Washington D.C., USA, August 27-31, 2007.

3. Dan Marconett, Minsoo Lee, Xiaohui Ye, and S. J. Ben Yoo, "Intelligent Network-Layer Management for Wireless Mesh Networks," OPNETWORK 2007, 1544 Case

Studies: 802.11 WLAN and Wireless Mesh Networks, Washington D.C., USA, August 27-31, 2007.

Under Review:

1. Minsoo Lee, Dan Marconett, Xiaohui Ye, Rao Vemuri, and S. J. Ben Yoo, "Cross-Layer Self-Configuration in Dynamic Wireless Mesh Network using Q-Learning," IEEE Journal of Systems, Man, and Cybernetics, 2008.

2. Dan Marconett, Minsoo Lee, Rao Vemuri, and S. J. Ben Yoo, "Stability-based Distributed Mesh Network Clustering Using Reinforcement Learning," SECON '08: Fifth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad-Hoc Communications and Networks, 2008.

As per the prosposed work in the previous section, it is also proposed that for each subcomponent of the Autonomic Management System that is simulated and later implemented, that the requisite conference and journal publications will be completed in support of said work, maximizing the recognition and exposure of the ideas proposed and results achieved.

REFERENCES

[1] M. Allman, D. Glover, L. Sanchez, "*Enhancing TCP Over Satellite Channels using Standard Mechanisms*", RFC 2488, IETF. January 1999.

[2] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, R. Katz,"*A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links*", Proceedings of ACM SIGCOMM. Stanford, CA, USA. August 1996.

[3] S. Corson and J. Macker, "Mobile Ad hoc Networking: Routing Protocol Performance Issues and Evaluation Considerations", IETF RFC 2501, Jan 1999.

[4] I. F. Akyildiz, X. Wang and W. Wang, "Wireless mesh networks: a survey," *Computer Networks*, Volume 47, Issue 4, 15 March 2005, Pages 445-487.

[5] D. Clark, C. Partridge, J. C. Ramming, and J. Wroclawski, "A Knowledge Plane for the Internet," *Proc. of ACM SIGCOMM '03,* August 2003.

[6] P. Demestichas, G. Dimitrakopoulos, J. Strassner, and D. Bourse, "Introducing Reconfigurability and Cognitive Networks Concepts in the Wireless World," *IEEE Vehicular Technology Magazine,* June 2006, pp. 32-39.

[7] R. W. Thomas, L. A. DaSilva, and A. B. MacKenzie, "Cognitive Networks," *IEEE DySPAN,* November 2005, pp. 352-260.

[8] H. Duran-Limon, G. Blair, A. Friday, T. Sivaharan and G. Samartzidis, " Resource and QoS Management Framework for a Real-Time Event System in Mobile Ad Hoc Environments," *in Proc. of IEEE WORDS '03,* pp. 217-224, Oct. 2003.

[9] L. Kant, S. Demers, P. Gopalakrishnan, R. Chadha, L. LaVergne and S. Newman, "Performance Modeling and Analysis of a Mobile Ad Hoc Network Management System," *in Proc. of MILCOMM '05,* Oct. 2005.

[10] S. M. Lake Sr., "Cognitive Networking with Software Programmable Intelligent Networks for Wireless and Wireline Critical Communications," *in Proc. of MILCOMM '05*, vol. 3, pp. 1693-1699, Oct. 2005.

[11] T. Forde, L. Doyle and D. O'Mahony, "Ad Hoc Innovation: Distributed Decision Making in Ad Hoc Networks," *IEEE Communications Magazine,* April 2006, pp. 131-137.

[12] T. Forde, L. Doyle and D. O'Mahony, "Self-Stabilizing Network-Layer Auto-Configuration for Mobile Ad Hoc Network Nodes," *in Proc. of IEEE WiMob '05,* vol. 3, pp. 178-185, Aug. 2005.

[13] H. Kawano, S. Nishio, and T. Hasegawa. "Knowledge Acquisition in Communication Networks: Towards Flexible Network Management." *In Proceedings IEEE Tencon '92,* pp. 881-885, November 1992.

[14] N. Sanderson, V. Goebel and E. Munthe-Kaas. "Knowledge Management in Mobile Ad-Hoc Networks for Rescue Scenarios."*in Proceedings International Semantic Web Conference,* 2004.

[15] R. Mortier and E. Kiciman. "Autonomic Network Management: Some Practicle Concerns." *In Proceedings SIGCOMM '06,* September 2006.

[16] P. Groth, S. Jiang, S. Miles, S. Munroe, V. Tan, S. Tsasakou and L. Moreau. "An Architecture for Provenance Systems." *The Provenance Consortium,* 2005.

[17] P. Buneman, S. Khanna, and W. Tan. "Data Provenance: Some Basic Issues." *In Lecture Notes in Computer Science (LNCS): FST TCS 2000 – Foundations of Software Technology and Theoretical Computer Science,* pp. 87-92, 2000.

[18] J. Y. Yu and P. H. J. Chong. "A Survey of Clustering Schemes for Mobile Ad Hoc Networks," IEEE Communications Surveys and Tutorials, First Quarter 2005.

[19] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," Proceedings of DIAL 1999.

[20] T. J. Kwon, M. Gerla, V. K. Varma, M. Barton, and T. R. Hsing, "Efficient Flodding with Passive Clustering – An Overhead-Free Selective Forward Mechanism for Ad Hoc/Sensor Networks," Proceedings of the IEEE, Vol. 91, NO. 8, August 2003.

[21] S. Sivavakeesar, G. Pavlou, and A. Liotta, "Stable Clustering Through Mobility Prediction for Large-Scale Multihop Intelligent Ad Hoc Networks," WCNC 2004, IEEE Communications Society.

[22] P. Y. Chen, "A Mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks," International Journal of Network Management, ACM, November 2001.

[23] O. Younis and S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach," Proc. of IEEE Infocom 2004.

[24] S. Basagni, "Distributed Clustering for Ad-hoc Networks," Proc. of International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN) 1999.

[25] V. Bricard-Vieu and B. Mikou. "Distributed Mobility Prediction-Based Weighted Clustering Algorithm for MANETs," ICOIN 2005, LNCS 3391, pp. 717-724, 2005.

[26] T. Mitchell. *Machine Learning*. Boston, MA: WCB/McGraw-Hill, 1997.

[27] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[28] A. Venkateswaran, V, Sarangan, N. Gautam, and R. Acharya, "Impact of Mobility Prediction on the Temporal Stability of MANET Clustering Algorithms," Proc. of PE-WASUN 2005.

[29] S. Davy, K. Barrett, et. al, "Policy-Based Architecture to Enable Autonomic Communications – A Position Paper." Proceedings of IEEE CCNC 2006, pp. 590-594.

[30] C. E. Koksal and H. Balakrishnan, "Quality-Aware Routing Metrics for Time-Varying Wireless Mesh Networks," *IEEE Journal on Select Areas in Communications*, vol. 24, no. 11, pp. 1984-1994, Nov. 2006.

[31] J. Boyan and M. Littman, "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach," *in Advances in Neural Information Processing Systems,* vol. 7, pp. 671-678, 1994.

[32] P. Wang and T. Wang, "Adaptive Routing for Sensor Networks using Reinforcement Learning," *Proc. of IEEE CIT '06*, 2006.

[33] E. Gelenbe, R. Lent, and A. Nunez, "Self-Aware Networks and QoS," *in Proc. of the IEEE,* vol. 92, no. 9, Sept. 2004.

[34] G. Di Caro and M. Dorigo, "AntNet: Distributed Stigmergic Control for Communication Networks," *Jounal of Artificial Intelligence Research,* vol. 9, pp. 317-365, 1998.

[35] S. Ziane and A. Mellouk, "A Swarm Intelligent Scheme for Routing in Mobile Ad Hoc Networks," *in Proc. of IEEE ICW '05,* 2005.

[36] J. Dowling, E. Curran, R. Cunningham and V. Cahill, "Using Feedback in Collaborative Reinforcement Learning to Adaptively Optimize MANET Routing," *IEEE Trans. On Systems, Man, and Cybernetics,* vol. 35, pp.360-372, 2005.

[37] T. Forde, L. Doyle and D. O'Mahony, "Ad Hoc Innovation: Distributed Decision Making in Ad Hoc Networks," *IEEE Communications Magazine,* April 2006, pp. 131-137.

Appendix A: Matlab Simulation Environment