Using Feedback in Collaborative Reinforcement Learning to Adaptively Optimize MANET Routing

Jim Dowling, Eoin Curran, Raymond Cunningham, and Vinny Cahill

***Abstract*—Designers face many system optimization problems when building distributed systems. Traditionally, designers have relied on optimization techniques that require either prior knowledge or centrally managed runtime knowledge of the system's environment, but such techniques are not viable in dynamic networks where topology, resource, and node availability are subject to frequent and unpredictable change. To address this problem, we propose collaborative reinforcement learning (CRL) as a technique that enables groups of reinforcement learning agents to solve system optimization problems online in dynamic, decentralized networks. We evaluate an implementation of CRL in a routing protocol for mobile *ad hoc* networks, called SAMPLE. Simulation results show how feedback in the selection of links by routing agents enables SAMPLE to adapt and optimize its routing behavior to varying network conditions and properties, resulting in optimization of network throughput. In the experiments, SAMPLE displays emergent properties such as traffic flows that exploit stable routes and reroute around areas of wireless interference or congestion. SAMPLE is an example of a complex adaptive distributed system.**

***Index Terms*—Feedback, learning systems, mobile ad hoc network routing.**

I. Introduction

THE USE of self-organizing techniques to engineer distributed systems has recently been driven by the need to build systems with improved performance and scalability characteristics in dynamic, decentralized environments, such as mobile *ad hoc* networks (MANETs) [1], [2]. In decentralized environments, designers cannot make prior assumptions about the system's environment, such as an expected network topology, and hierarchical control techniques are not viable due to network dynamism [1], [2]. In MANETs, distributed systems designers overcome the lack of prior or centrally managed system knowledge by making few assumptions about the system's environment. This leads to the design of inefficient mechanisms to support services, such as network flooding for route discovery, that cannot adapt to a changing environment to improve system performance.

Inspired by complex adaptive systems from nature [3]–[5], there have been recent advances in using self-organizing techniques to build complex adaptive distributed systems (CADS) that operate in dynamic, decentralized environments [6]–[8]. In CADS, system behavior or structure emerges from the local interactions between the different components or agents without any explicit representation of system behavior or structure on the level of the individual component or agent. A common feature of CADS is their ability to perform complex collective tasks and to collaboratively adapt their system behavior or structure to changes in their environment using relatively simple agent-level behaviors. Positive and negative feedback between agents have been identified as key mechanisms in realizing complex adaptive system behavior [3], [4].

This paper introduces and evaluates collaborative reinforcement learning (CRL) as a self-organizing technique for building a MANET routing protocol, called SAMPLE, as a CADS that can adapt and optimize system routing behavior to a changing environment using positive and negative feedback. CRL models the desired system routing properties as system optimization problems that are solved by decentralized, collaborating routing agents that learn routing policies using reinforcement learning (RL) [9], [10]. RL is an unsupervised learning technique that allows an autonomous agent to monitor the state of its environment and take actions that affect its environment in order to learn an optimal policy. RL can be used to solve optimization problems for an individual agent [9], but the application of RL to adaptively solve system optimization problems in dynamic, decentralized environments is an open research problem.

CRL extends RL with different feedback models, including a negative feedback model that decays an agent's local view of its neighborhood and a collaborative feedback model that allows agents to exchange the effectiveness of actions they have learned with one another. In a system of homogeneous agents that have common system optimization goals and where agents concurrently search for more optimal actions in different states using RL, collaborative feedback enables agents to share more optimal policies, increasing the probability of neighboring agents taking the same or related actions. When CRL is applied to SAMPLE, this process can produce positive feedback in route selection for a group of routing agents in a network. The positive feedback mechanism in SAMPLE reinforces changes in agent routing behavior to favor the use of stable routes in a MANET. In SAMPLE, the positive feedback in route selection process causes convergence between the routing policies of agents and continues until negative feedback, produced either by routing congestion or our decay model, causes agents to adapt their routing policies.

Typically, the goal of system optimization for routing protocols is to have the routing agents' policies converge to produce collective routing behavior that meets the system routing

Manuscript received May 29, 2004; revised October 10, 2004 and November 16, 2004. This work was supported in part by the TRIP project funded under the Programme for Research in Third Level Institutions (PRTLI) administered by the Higher Education Authority (HEA) of Ireland and in part by the European Union funded "Digital Business Ecosystem" Project IST-507953. This paper was recommended by the Guest Editors.

The authors are with the Department of Computer Science, Trinity College, Dublin, Ireland (e-mail: jpdowling@cs.tcd.ie).

Digital Object Identifier 10.1109/TSMCA.2005.846390

optimization criteria. However, in open dynamic distributed systems, such as MANETs, the environment is nonstationary and we require agents than can collectively adapt their routing behavior to a changing environment to continue to meet the system routing optimization criteria. In general, we believe that the adaptability of system behavior to changes in its environment is as important an evaluation criterion for CADS as the more traditional criterion for static environments of convergence and stabilization on optimal system behavior.

The remainder of this paper is structured as follows. Section II introduces the CRL model, Section III presents SAMPLE as both a CRL system and an on-demand routing protocol for *ad hoc* networks. Section IV compares simulation results for SAMPLE with two widely used on-demand MANET routing protocols in different scenarios and explains the differing abilities of the protocols to adapt and optimize to a changing MANET environment. Finally the paper concludes with related work and a summary of how positive and negative feedback in CRL can be used to build CADS.

II. Collaborative Reinforcement Learning

In reinforcement learning, an autonomous agent associates actions with system states, in a trial-and-error manner, and the outcome of an action is observed as a reinforcement that, in turn, causes an update to the agent's optimal policy using a reinforcement learning strategy [9], [10]. The goal of reinforcement learning is to maximize the total reward (reinforcements) an agent receives over a time horizon by selecting optimal actions. Agents may take actions that give a poor payoff in the short-term in the anticipation of higher payoff in the medium/longer term. In general, actions may be any decisions that an agent wants to learn how to make, while states can be anything that may be useful in making those decisions.

Reinforcement-learning problems are usually modeled as Markov decision processes (MDPs) [9], [10]. An MDP consists of a set of states, $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$, a set of actions, $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$, a reinforcement function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and a state transition distribution function: $P : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$, where $\Pi(\mathcal{S})$ is the set of probability distributions over the set $\mathcal{S}$.

A. Decomposing System Optimization Problems

In CRL, system optimization problems are decomposed into a set of discrete optimization problems (DOPs) [6] that are solved by collaborating RL agents. There are many system optimization problems in distributed systems that can be naturally discretized into DOPs that can be distributed amongst agents in a distributed system, such as load balancing of computation over a group of servers and locating replicated resources in a network.

The solution to each DOP is initiated at some starting agent in the network and terminated at some (potentially remote) agent in the network. Each agent uses its own policy to decide probabilistically on which action to take to attempt to solve a DOP. In CRL the set of available actions that an agent can execute include *DOP action*s, $\mathcal{A}_{p_i}$, that try to solve the DOP locally, *delegation action*s, $\mathcal{A}_{d_i}$, that delegate the solution of the DOP to a neighbor and a *discovery action* that any agent can execute in any state to attempt to find new neighbors. An agent is more likely to delegate a DOP to a neighbor when it either cannot solve the problem locally or when the *estimated cost* of solving it locally is higher than the estimated cost of a neighbor solving it.

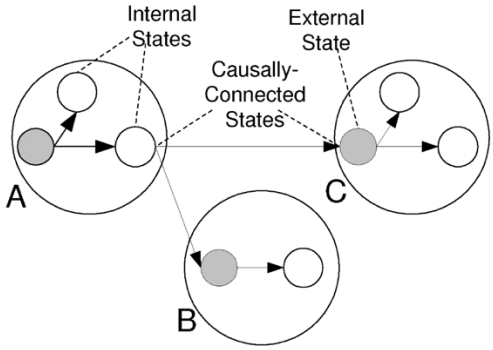

Fig. 1. Causally connected states between the MDPs in agents A, B, and C.

B. Heterogeneous Environments

In heterogeneous distributed systems, agents typically possess different capabilities for solving a given DOP. To model the differing capabilities of agents, CRL allows newly discovered agents to negotiate the establishment of causally connected states with their neighbors by exchanging device capability information. Causally connected states represent the contractual agreement between neighboring agents to support the delegation of DOPs from one to the other. Causally connected states map an internal state on one agent to an external state on at least one neighboring agent. An internal state on one agent can be causally connected to external states on many different neighboring agents, see Fig. 1. An agent's set of causally connected neighbors represents its partial view of the system.

In CRL, for every neighbor, n_j, with whom agent n_i shares a causally connected state, there exists a delegation action $a_j \in \mathcal{A}_d$ that represents an attempt by n_i to delegate a DOP to n_j. If the delegation action is successful, n_i makes a state transition to its causally connected state s, terminating the MDP at n_i, and n_j initiates a new MDP to handle the DOP. Apart from an agent's capabilities, runtime factors, such as the agent's available resources and the quality of its network connections, also affect the ability of agents to solve a given DOP. These are modeled in the agent's reward model.

C. Model-Based Reinforcement Learning

RL strategies can be either model-free, such as Q-learning [11], or model-based [10], [12], [13]. Model-based learning methods build an internal model of the environment and calculate the optimal policy based on this model, whereas model-free methods do not use an explicit model and learn directly from experience. Model-based methods are known to learn in many settings much faster than model-free methods, since they can reuse information stored in their internal models [14]. In general, model-based methods have been less popular in RL because of their slower execution times and greater storage costs, especially as the state size grows. However, in distributed systems where acquiring real-world experience is

expensive, the model based approach has a distinct advantage over model-free methods as much more use can be made of each experience. Model-based learning requires that the state transition model and possibly the reward model are updated throughout the execution of the CRL algorithm.

D. Distributed Model-Based Reinforcement Learning

In distributed systems, when estimating the cost of the state transition to a remote state on a neighboring agent we also have to take into consideration the network connection cost to the neighboring agent. For this reason, we use a distributed model-based reinforcement learning algorithm that includes both the estimated optimal value function for the next state at agent $n_i, V_j(s')$, and the connection cost, $D_i(s' \mid s, a) \in \mathbb{R}$ where $a \in \mathcal{A}_{d_i}$, to the next state when computing the estimated optimal state-action policy as $Q_i(s, a)$ at agent n_i, see (5).

In the distributed model-based *RL* algorithm, our reward model consists of two parts. Firstly, a *MDP termination cost*, $R(s, a) \in \mathbb{R}$, provides agents with evaluative feedback on either the performance of a local solution to the DOP or the performance of a neighbor solving the delegated DOP. Secondly, a connection cost model, $D_i(s' \mid s, a)$, provides the estimated network cost of the attempted delegation of the DOP from a local agent to a neighboring agent. The connection cost for a transition to a state on a neighboring agent should reflect the underlying network cost of delegating the DOP to a neighboring agent, while the connection cost for a transition to a local state after a delegation action should reflect the cost of the failed delegation of the DOP. The connection cost model requires that the environment supplies agents with information about the cost of distributed systems connections as rewards. Ideally a middleware will provide support for monitoring connection quality, such as in [15].

E. Advertisement

In CRL, neighbors are informed of changes to $V_j(s)$ of a causally connected external state, s, at agent n_j using an *advertisement* (Fig. 2). Each agent maintains a local view of its neighbors by storing V values for causally connected states to neighbors in a local cache, Cache_i. The cache consists of a table of Q-values, for all delegation actions, a_d, at agent n_i, and the last advertised $V_j(s)$ for sucessful transition to the causally connected state. A Cache_i entry is a pair $(Q_i(s, a_j), r_j)$, where r_j is the cached $V_j(s)$ value. When the agent n_i receives a $V_j(s)$ advertisement from neighboring agent n_j for a causally connected state s, it updates r_j in $(Q_i(s, a_j), r_j)$.

F. Dynamic Environments and Decay

Similar to RL, CRL models are based on MDP learning methods that require complete observability [10], however at any given agent in a decentralised system the set of system-wide states are only partially observable. To overcome problems related to partially observable environments, state transition and connection cost models can be built to favor more recent observations using a finite-history-window [10], and cached V_j values become stale using a *decay* model [6]. In CRL, we decay cached V_j information in the absence of new advertisements

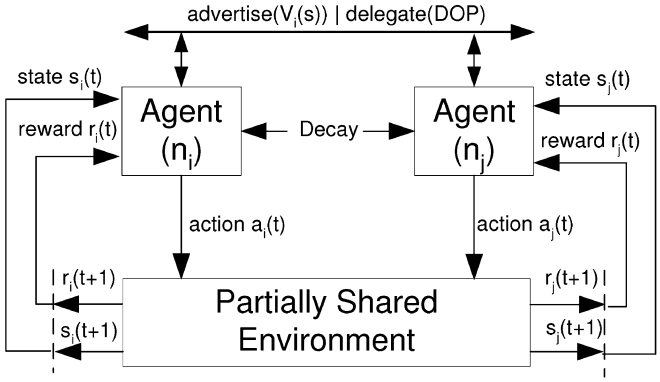

Fig. 2. In CRL, an agent, n_i, takes an action at time t, $a_i(t)$, and makes in a transition to state $s_i(t+1)$, that produces a reward, $r_i(t+1)$. Agents can advertise updated V values to neighbors, and delegate a DOP to a neighboring agent. Cached V values are decayed over time.

of V_j values by a neighbor as well as after every recalculation of Q_i values. The absence of V_j advertisements amounts to negative feedback and allows us to use a cleanup updater to remove cache entries, actions, and agents with stale values in the system. The rate of decay is configurable, with higher rates more appropriate for more dynamic network topologies.

G. CRL Algorithm

The CRL algorithm can be used to solve system optimization problems in a multiagent system, where the system optimization problem can be discretized into a set of DOPs and modeled as absorbing MDPs [9] in the following schema.

1) A dynamic set of *agents* $\mathcal{N} = \{n_1, n_2, \ldots, n_M\}$, often corresponding to nodes in a distributed system.
2) Each agent n_i has a dynamic set, $\mathcal{V}_i$, of *neighbors* where $\mathcal{V}_i \subset \mathcal{N}$ and $n_i \notin \mathcal{V}_i$.
3) Each agent n_i has a fixed set of states $\mathcal{S}_i$, where $\mathcal{S}_i \subseteq \mathcal{S}$ and $\mathcal{S}$ is the system-wide set of states.
4) Agents have both *internal* and *external states.*
   $\mathrm{Int} : \mathcal{N} \to \mathcal{P}(\mathcal{S})$ is the function that maps from the set of agents to a nonempty set of internal states that are not visible to neighboring agents.
   $\mathrm{Ext} : \mathcal{N} \to \mathcal{P}(\mathcal{S})$ is the function that maps from the set of agents to a set of externally visible states. The relationship between internal and external states is the following:
   $$\begin{matrix} \mathrm{Int}(n_i) \subset \mathcal{S}_i \\ \mathrm{Ext}(n_i) \subset \mathcal{S}_i \end{matrix} \wedge \quad \begin{matrix} \mathrm{Int}(n_i) \cup \mathrm{Ext}(n_i) = \mathcal{S}_i \\ \mathrm{Int}(n_i) \cap \mathrm{Ext}(n_i) = \{\,\} \end{matrix}. \tag{1}$$
5) We define a set of causally connected states between agents n_i and n_j as
   $$\mathcal{C}_{n_i n_j} = \mathrm{Int}(n_i) \cap \mathrm{Ext}(n_j), \qquad \text{where} \quad n_j \in \mathcal{V}_i \tag{2}$$
   $s \in \mathcal{C}_{n_i n_j}$ is a causally connected state where an internal state s at n_i corresponds to an external state s at n_j.
6) Each agent n_i has a dynamic set of actions
   $$\mathcal{A}_i = \mathcal{A}_{d_i} \cup \mathcal{A}_{p_i} \cup \{\text{discovery}\} \tag{3}$$
   where $\mathcal{A}_i \subseteq \mathcal{A}, \mathcal{A}_{d_i}$ are the set of delegation actions, $\mathcal{A}_{p_i}$ are the set of DOP actions. The *discovery* action

updates the set of neighbors $\mathcal{V}_i$ for agent n_i and queries if discovered neighboring agent n_j provides the capabilities to accept a delegated MDP from n_i. If it does, $\mathcal{A}_{d_i}$ is updated to include a new delegation action that can result in a state transition to $s \in \mathcal{C}_{n_i n_j}$ and the delegation of a MDP from n_i to n_j.

7) There are a fixed set of state transition models, $P_i(s' \mid s, a)$, that model the probabilities of making a state transition from state s to state s' under action a.
8) $D_i : \mathcal{S}_i \times \mathcal{A}_{d_i} \times \mathcal{S}_i \rightarrow \mathbb{R}$ is the connection cost function that observes the cost for the attempted use of a connection in a distributed system. $D_i(s' \mid s, a)$ is the connection cost model at agent n_i that describes the estimated cost of making a transition from state s to state s' under delegation action a.
9) We define a cache at n_i as $\text{Cache}_i = \{(Q_i(s, a_j), r_j) : r_j \in \mathbb{R} \wedge a_j \in \mathcal{A}_{d_i}\}$. The value r_j in the pair $(Q_i(s, a_j), r_j)$ corresponds to the last advertised $V_j(s)$ received by agent n_i from agent n_j. For each $n_j \in \mathcal{V}_i$, Cache_j is updated by a V_j *advertisement* for a causally connected state. The update replaces the r_j element of the pair $(Q_i(s, a_j), r_j)$ in Cache_i with the newly advertised V_j value.
10) $\text{Decay}(r_j) \rightarrow \mathbb{R}$ is the decay model that updates the r_j element in Cache_i,

$$\text{Decay}(r_j) = r_j \cdot \rho^{td} \tag{4}$$

where td is the amount of time elapsed since the last received advertisement for r_j from agent n_j and ρ is a scaling factor that sets the rate of decay.

11) A *cleanup updater* is available at each agent n_i to remove stale elements from its set of neighbors $\mathcal{V}_i$ delegation actions $\mathcal{A}_{d_i}$, connected states $\mathcal{C}_{n_i n_j}$, and its Cache_i. When a $(Q_i(s, a_j), r_j)$ entry in the cache drops below a specified threshold, the cleanup updater removes the delegation action a_j from $\mathcal{A}_{d_i}$, the stale connected state s from $\mathcal{C}_{n_i n_j}$, and the pair $(Q_i(s, a_j), r_j)$ from Cache_i. If after removing $s, c_{s_i s_j} = \{\}$ for some neighbor n_j of n_i, then n_j is removed from $\mathcal{V}_i$.
12) The distributed model-based reinforcement learning algorithm is

$$Q_i(s, a) = R(s, a) + \sum_{s' \in S_i} P_i(s' \mid s, a) \cdot \Big(D_i(s' \mid s, a) + \text{Decay}\big(V_j(s')\big) \Big) \tag{5}$$

where $a \in \mathcal{A}_d$. If $a \notin \mathcal{A}_d$, this defaults to the standard model-based reinforcement learning algorithm [10] with no connection costs or decay function. $R(s, a)$ is the MDP termination cost, $P(s' \mid s, a)$ is the state transition model that computes the probability of the action a resulting in a state transition to state s', $D_i(s' \mid s, a)$ is the estimated connection cost and $V_j(s')$ is $r_j \in \text{Cache}_i$ if $a \in \mathcal{A}_d$, and $V_i(s')$ otherwise. Note that rewards that are received in the future are not discounted since agents do not learn about state transitions after successful delegation to a neighboring agent.

13) The value function at agent n_i, V_i, can be calculated using the Bellman optimality equation [16]

$$V_i(s) = \max_a [Q_i(s, a)].$$

H. Adaptation and Feedback in Distributed Model-Based Reinforcement Learning

Adaptation of system behavior in CRL is a feedback process in which a change in the optimal policy of any RL agent, or a change in the system's environment as well as the passing of time causes an update to the optimal policy of one or more RL agents, see (5). This is in contrast to model-free Q-learning, where agents only adapt their behavior after executing actions [11]. In CRL, changes in an agent's environment provide feedback into the agent's state transition model and connection cost model, while changes in an agent's optimal policy provides collaborative feedback to the cached V values of its neighboring agents using advertisement. Time also provides (negative) feedback to an agent's cached V values. As a result of the different feedback models in CRL, agents can utilize more information when learning an optimal policy in a distributed system. Collaborative feedback also enables agents to learn from their neighbors to solve collective system problems.

III. SAMPLE: MANET Routing Using CRL

A. Routing and System Optimization

The CRL model was evaluated by using it to build a MANET routing protocol called SAMPLE. Routing algorithms are designed to meet multiple, often conflicting system optimization criteria [17]. In SAMPLE, we attempt to maximize overall network throughput, maximize the ratio of delivered packets to undelivered packets and minimize the number of transmissions required per packet sent. *Ad hoc* routing is a challenging problem as it exhibits properties such as the lack of global network state and frequently changing network topology due to node mobility. This ensures that system properties of the protocol only emerge from local routing decisions and that routing agent's behavior has to frequently adapt to a changing environment.

In Section IV, we compare experimentally SAMPLE with the two most widely deployed MANET routing protocols, *ad hoc* on-demand distance vector routing (AODV) [18] and dynamic source routing (DSR) [19]. Both protocols make several static assumptions about the MANET environment to avoid problems typically encountered by proactive routing protocols in MANETs.

The main assumptions made by AODV and DSR [20] are, first, that network links are either working or not working. This discrete model for links is essentially based on an assumption of perfect radio links, ignores the effects of interference and network contention, and is generally based on the last measurement of a link's status. The second assumption is that MANETs have a random network topology. AODV and DSR use on-demand routing where routes to destinations are only discovered when needed using flooding [18], [19]. On-demand routing is essentially based on the assumption of a random network topology,

although both DSR and AODV attempt to improve routing performance using route maintenance [19] and link maintenance techniques, such as hello messages in AODV.

The SAMPLE routing protocol seeks to experimentally test the following hypotheses about AODV and DSR.

1) Due to their discrete model of network links, AODV and DSR will not be able to adapt to use different quality links, which is important when link quality deteriorates.
2) AODV and DSR will not be able to adapt to favorable changes in network topology, as they make the prior assumption of a random network topology.

Our aim is to compare how effective AODV, DSR, and SAMPLE are in adapting their routing behavior to a changing network environment in order to meet the system optimization criteria idenitifed above.[1] In order to meet the system optimization criteria for SAMPLE, we identify link quality as an important metric in maximizing network throughput and minimizing the number of transmissions in the network. In Section III-C, we define a more optimal network link in a MANET as a stable link that has a higher probability of successful packet transmission. A stable route is a path that contains a set of network links that each has high probability of successful packet transmission, generally due to nodes having low mobility and high network up-time characteristics. In MANETs, stable routes are preferable to less reliable routes with fewer hops [22].

B. SAMPLE Routing Protocol

SAMPLE is an on-demand *ad hoc* routing protocol based on CRL [20]. Routing information is captured as a route cost (V value) and is advertised in the network by attaching it to data packets. Each agent has routing tables to store route cost information to its neighboring nodes and the last advertised V value from those neighbors for each destination or source node in use. Each packet sent by the routing protocol will contain the information shown in Table I. Any node receiving a packet will store the V values for its neighboring node, and also update the local state transition model, see Section III-C, for that node. SAMPLE uses the distributed model-based RL algorithm, defined by (5), to calculate a routing agent's V values that are attached to any packet that it transmits.

1) Routing Action Selection in SAMPLE: In SAMPLE, for each of the possible routing (delegation) actions by an agent to one of its M neighbors, the decision of which action to take for a packet from node n_i is chosen probabilistically between the set of possible next hops on the route using Boltzmann-action selection [9]:

$$P(B,a) = \frac{e^{-\left(Q_i(B,a_d)\right)/T}}{\sum_{a_d} e^{-\left(Q_i(B,a_d)\right)/T}}, \quad a \in \mathcal{A}_{d_i}. \tag{6}$$

The parameter T is the temperature and it determines the likelihood of choosing suboptimal routing actions. The higher the temperature, the more likely a suboptimal action is likely to be chosen. Varying the temperature controls the amount of exploration that will be taken. An increase in T increases the possibility of a packet finding a more optimal route but also increases the possibility of a packet arriving later and out of order. In general, networks with more dynamic topologies and few stable routes should have a higher temperature than more stable network topologies, such as the metropolitan area MANET introduced in Section IV-B. We carried out a series of experiments, tuning the value T, to optimize the performance of SAMPLE for different network scenarios [23]. In the experiments in Section IV, we chose a low temperature $T = 3$ to reduce the amount of exploratory packets and increase the exploitation of the stable links in the network. SAMPLE also uses a simple *greedy heuristic* in order to restrict exploration to useful areas of the network: we only allow node n_i to forward to those neighboring nodes with a V value that is less than that of n_i.

In the case that no routing information is available, such as before the network has been bootstrapped with traffic or the routing information has gone stale, the routing protocol must still be able to function correctly. A *broadcast* action, that represents the discovery action in CRL, is provided to bootstrap the network with routing information. We also allow this action to be chosen during normal operation, with a certain (albeit low) probability in order to discover new routes.

TABLE I
SAMPLE PACKET FORMAT. PACKETS HAVE ORIGIN AND DESTINATION ADDRESSES, AS WELL AS SEQUENCE NUMBERS TO IDENTIFY DUPLICATES. $V_i(B_{\mathrm{src}})$ IS THE ESTIMATED ROUTE COST FOR SENDING A PACKET FROM THE CURRENT NODE i TO NODE src, WHILE $V_i(B_{\mathrm{dest}})$ IS ESTIMATED ROUTE COST FOR SENDING A PACKET FROM THE CURRENT NODE i TO NODE dest

Field Name	Field Type	Field Contents
ORIGIN	IP Address	The original source of the packet
DESTINATION	IP Address	The final destination of the packet
SEQUENCENUMBER	Integer	Generate at source node
SOURCEDIST	Floating-Point	$V_{\mathrm{i}}(\mathrm{B}_{\mathrm{src}})$
DESTINATIONDIST	Floating-Point	$V_i(\mathrm{B}_{\mathrm{dest}})$
IPPACKET	Data Packet	IP Packet, or empty

C. State Transition Model for Network Links

In contrast to the discrete model of network links favored by DSR and AODV, SAMPLE uses a state transition model, $P_i(s' \mid s, a)$ where $a \in \mathcal{A}_{d_i}$, to represent the probability of successful transmission over a given network link in our *ad hoc* network. The state transition model is a statistical model that acquires information about the estimated number of packets required for a successful unicast. In order to build the state transition model, we sample the number of different system events within a small time window τ into the past.[2] The events that we monitor are:

1) attempted unicast transmissions c_A;
2) successful unicast transmissions c_S;
3) received unicast transmissions c_R;
4) received broadcast transmissions, c_B;
5) promiscuously received (overheard) unicast transmissions c_P.

[1] We do not provide a formal treatment of routing as a multiobjective optimization problem as in [21].

[2] For the experiments in this paper $\tau = 10$ s

The rate of these events is used to estimate the probability of an attempted unicast transmission being successful, i.e., we attempt to estimate the future value of (c_S/c_A). Since a successfully received packet is indicative of a functioning network link, we allow receive events to influence our estimation to a configurable extent

$$E\left(\frac{c_S}{c_A}\right) = \frac{c_S + \alpha\beta(c_R + c_B + c_P)}{c_A + \beta(c_R + c_B + c_P)}. \tag{7}$$

The parameter α represents our *belief* about the probability of successfully transmitting a packet in the case that we have received but not attempted to transmit. The parameter β controls how much received packets are weighted compared to transmitted packets. For the experiments carried out in this paper, we used values of 0.5 and 0.2 for α and β, respectively. Sent packets are weighted higher than received packets due to the lower information content in received broadcast and promiscuously received packets. Note that this is quite a simple estimate of delivery ratio. More complicated measures of link quality could be devised and other variables considered, as shown in [24].

For a given network link between agent n_i and agent n_j in the network we have

$$P_i(s' \mid s, a_j) = E\left(\frac{c_S}{c_A}\right) \tag{8}$$

where $P_i(s' \mid s, a_j)$ represents the probability of a successful unicast, given a state transition from state s at node n_i under attempted delegation action $a_j \in \mathcal{A}_{d_i}$ to a causally connected state $s' \in \mathcal{C}_{n_i n_j}$. If $s' \notin \mathcal{C}_{n_i n_j}$, $P_i(s' \mid s, a_j)$ represents the probability of an unsuccessful unicast.

The state transition model ensures that stable links are favored in routes, as the total estimated route cost is a product of the neighbor's estimated route plus the connection cost and the estimated probability of successful transmission over the link to the neighbor. Route costs increase rapidly with degradation of link quality, thus favoring routes that use links with more successful transmission ratios.

D. Connection Costs for Network Links

SAMPLE uses a reward model for the estimated connection cost, $D_i(s' \mid s, a) \in \mathbb{R}$, that is designed to approximate the number of transmissions[3] needed to transmit a packet along that link. The reward model is a simple static model of connection cost based on the relative cost differential between a successful unicast over a connection versus an unsuccessful unicast over the connection.[4] In an 802.11, network unicasts are retried up to seven times [25] before a packet transmission failure event. We use static costs $r_S \in \mathbb{R}$ and $r_F \in \mathbb{R}$ to model the reward when transmission succeeds under a delegation action and fails, respectively. The rewards are set at values -1 and -7, respectively, to reflect the cost of unicast failures in an 802.11 network. Depending on the distributed system, more complex connection cost models could be devised based on parameters such as link latency, throughput, or some quality of service metric.

[3]transmissions made by the 802.11 protocol, i.e., including the number of retransmissions that are made until the packet is acknowledged successfully

[4]Note we do not need to model broadcast as it is not included in the calculation of the V values, see Section II-G.

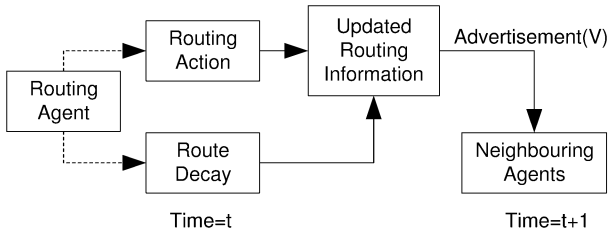

Fig. 3. Routing information in SAMPLE is updated by routing actions, decay, and advertisement.

E. Advertisement and Decay in SAMPLE

In SAMPLE, each agent stores the last advertised route cost to a given destination from each of its neighbors in a routing table, but considers this value to decay (i.e., grow steadily larger) from the time it is advertised (see Fig. 3). In this way, routes that are not advertised are gradually degraded and eliminated from consideration for routing decisions. The rate of route decay can be configured to match an estimated mobility model for a MANET.

When a lower cost route to a destination is advertised to neighbors, it increases the probability of that route being used in the near future by neighbors. The more successful a route is, the more it is advertised throughout the network, resulting in a positive feedback mechanism whereby stable routes with high traffic flows emerge in the network. However, as these stable routes become more popular they can become congested resulting in degradation of the state transition model for the stable links. In this case negative feedback appears in the form of the capacity constraints of the network links causing routing agents to favor alternative routes to the destination. Negative feedback enables routing agents to adapt their behavior to send traffic flows around the congested links. An interesting emergent property of SAMPLE is that the amount of effort used for routing to a given destination is relative to the popularity of that destination. Hence the quality of routing information is higher for more popular traffic destinations, such as nodes providing internet gateway services in metropolitan area MANETs.

F. SAMPLE as a CRL System

1) Each node n_i has a set of states $\mathcal{S}_i = \{B, P, D\}$, where B indicates that a packet is in a buffer waiting to be forwarded, P indicates that a packet has been successfully unicast to a neighbor, and D indicates that a packet has been delivered at node n_i (see Fig. 4).
2) The external and internal states of n_i are $\mathrm{Ext}(n_i) = \{B\}$ and $\mathrm{Int}(n_i) = \{D, P\}$.
3) $\{B\}$ is the start state of the MDP at n_i and the terminal states are $\{D, P\}$.
4) The total set of actions available at n_i is $\mathcal{A} = \mathcal{A}_{d_i} \cup \{\text{deliver}\} \cup \{\text{broadcast}\}$. This includes a dynamic set of delegation actions $\mathcal{A}_{d_i} = \{U_i(v_0), \ldots, U_i(v_{M-1})\}$ where n_i has M reachable neighbors, v_0 to v_{M-1}, and the action $U_i(v_j)$ represents an attempted unicast from node n_i to its neighbor v_j. It also includes a set of DOP actions $\mathcal{A}_{p_i} = \{\text{deliver}\}$, where deliver represents an attempt to deliver the packet to the current node n_i. The broadcast action is an 802.11b broadcast used by routing

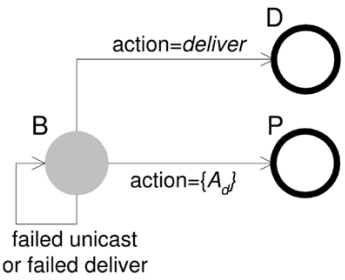

Fig. 4. SAMPLE MDP. Each agent's MDP has three states: B (packet in buffer), P (packet successfully unicast), and D (packet delivered at this agent). The actions available at different states in the MDP are a packet delivery action, a broadcast action and for each neighboring node, a delegation action (unicast).

agents to discover new neighbors, links, and routes in the network.

5) For $P \in \text{Int}(\mathcal{S}_i)$, there is a causally connected state $B \in \text{Ext}(\mathcal{S}_j)$, where $v_j \in \mathcal{V}_i$.
6) The family of advertisement updates in SAMPLE are opportunistic functions that piggyback V values in all unicast and broadcast packets. A node n_i promiscuously listens for V_j advertisements, updates r_j in its Cache$_i$ and recalculates its $Q_i(s, a_j)$ value for the appropriate destination.
7) $r_S \in \mathbb{R}$ and $r_F \in \mathbb{R}$ are the static connection costs when transmission succeeds ($B \rightarrow P$) under a delegation action, $U_i(v_j)$, and fails ($B \rightarrow B$), respectively.
8) The state transition models for the MDP are as follows.
   a) The probability of a packet transmission over a link succeeding is $p_{ijS} = P_i(P \mid B, a_j)$.
   b) The probability of a packet transmission over a link failing is $p_{ijF} = P_i(B \mid B, a_j) = 1 - P_i(P \mid B, a_j)$.
   c) The probability of the packet being delivered if the current node is the destination is $P_i(D \mid B, \text{deliver}) = 1$, if $n_i = \text{dest}$.
   d) For all other state transitions under all actions $P_i(S' \mid S, a) = 0$..
9) The following are the connection cost models when both delegation actions and other actions are attempted.
   a) $D_i(B \mid P, a) = r_S, \quad \text{where } a \in \mathcal{A}_{d_i}$.
   b) $D_i(B \mid B, a) = r_F, \quad \text{where } a \in \mathcal{A}_{d_i}$.
   c) $D_i(s' \mid s, a) = 0, \quad \text{where } a \notin \mathcal{A}_{d_i}$.
10) The MDP termination cost model is $R(s, a) = r_D$. We set its reward values to -7 for the execution of a *broadcast* action in any state, 0 for a delegation action and to 0 for completion of the DOP. We would normally expect that the reward r_D received for the local solution to the routing DOP to be a high value relative to a delegation action to increase the probability that a deliver action is executed when a packet arrives at its destination node. However, in the implementation of SAMPLE, we optimized the solution to the problem by simply always choosing the deliver action when a packet reaches its destination.
11) In the decay model, $\text{Decay}(r_j) = r_j \cdot \rho^{td}$, we set ρ to 1.1 [20]. This value was obtained through experimentation, with the goal being the optimization of the protocol for the metropolitan area network scenario in Section IV-B. For wireless networks with different mobility models and link stability characteristics, different values for ρ could be obtained through tuning and experimentation.
12) Given state B at node n_i and an attempted delegation action $a_d \in \mathcal{A}_{d_i}$, the Q_i values for that next state can be calculated using distributed model-based reinforcement learning

$$\begin{aligned} Q_i(B, a_j) &= R(B, a_j) + \sum_{s' \in \{B,P,D\}} P_i(s' \mid B, a_j) \\ &\quad \cdot \Big(D_i(s' \mid B, a_j) + \text{Decay}\big(V_i(s')\big) \Big), \quad a_j \in \mathcal{A}_{d_i} \\ &= r_D + p_{ijS}\big(\text{Decay}(V_i(P)) + r_S\big) \\ &\quad + p_{ijF}\Big(\text{Decay}\big(V_i(B)\big) + r_F\Big). \end{aligned}$$

Since

$$V_i(B) = \max_a Q_i(B, a_j)$$

we are seeking the solution of

$$\begin{aligned} V_i(B) &= r_D + \max_a \left[\frac{p_{ijS}\Big(r_S + \text{Decay}\big(V_i(P)\big)\Big) + p_{ijF} r_F}{1 - p_{ijF}} \right] \\ &= r_D + \max_a \left[\text{Decay}\big(V_i(P)\big) + r_S + \frac{p_{ijF}}{p_{ijS}} r_F \right]. \end{aligned} \tag{9}$$

G. Adaptation and Feedback in SAMPLE

As we can see from (9), a routing agent's behavior adapts to feedback from the execution of routing actions, feedback from the models of network link quality p_{ijF} and p_{ijS}, feedback from changes in the advertised route cost by a neighboring agent $V_i(P)$, and negative feedback from the decay of advertised route costs. The rate at which SAMPLE can adapt routing behavior to a changing environment is a function of

1) the parameters of the state transtition model α, β and the network events $(c_A, c_B, c_P, c_R, c_S)$.
2) the parameters of the decay model, ρ and td.
3) and the ratio of route exploration to exploitation T.

In the following section, we show how for particular values of the above parameters SAMPLE can adapt its routing behavior to a changing environment. Our goal is to show how feedback allows the behavior of routing agents to adapt to meet system optimization goals, rather than to show the effect of tuning these parameters. For further information on the effect of tuning these parameters in the network scenarios presented in the next section, see [23].

IV. Experimental Results and Discussion

We have implemented the SAMPLE routing protocol described in Section III in the NS-2 network simulator [26]. The simulation accuracy of NS-2 is discussed in [27]. We compare the performance of the SAMPLE routing protocol to that of two well-known on-demand routing protocols for MANETs, AODV, and DSR, in two different scenarios. The first scenario is a random network scenario that is designed to test how SAMPLE compares with AODV and DSR when the MANET environment reflects their design assumptions,

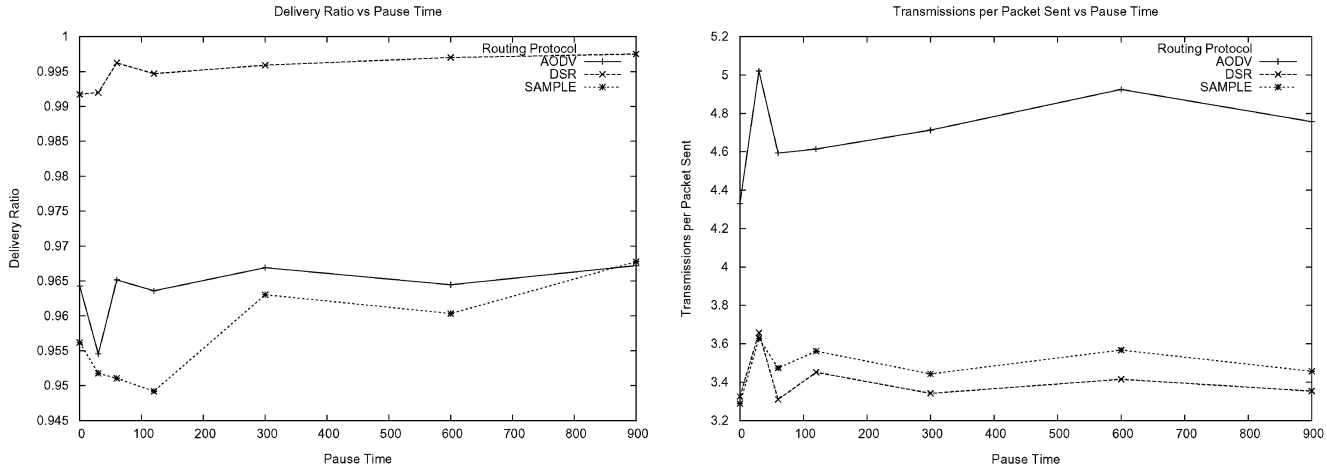

Fig. 5. Comparison of SAMPLE with AODV and DSR with no packet loss. This random network scenario represents idealized, unrealistic wireless network conditions that favors the design assumptions of both AODV and DSR. Packet delivery ratio near-optimal, at over 95% for all three protocols, with DSR showing the highest delivery ratio. The average number of transmissions per packet sent is lower is also very efficient, at lower than five transmissions/packet, for all three protocols.

see Section III-A. The second scenario is a metropolitan area MANET, where a subset of the links in the network are stable, that is designed to test the ability of the protocols to adapt their routing behavior to use the stable routes in the network. We also introduce congestion into both scenarios to investigate the effectiveness of the protocols in adapting their routing behavior to a changing MANET environment. Our goal here is to compare the performance of the SAMPLE routing protocol in different, changing environments with on-demand protocols that make static assumptions about the MANET environment using the system optimization criteria identified in Section III-A.

Since the SAMPLE routing protocol combines routing information with data packets, the metric of number of routing packets is not a valid one for comparison with AODV and DSR. For this reason, we use the number of transmissions (unicast or broadcast) that each protocol makes per application packet sent during the simulation run as a metric to compare the protocols. This metric represents the cost to the network of routing each data packet.

A. Adapting to Packet Loss in a Random Network

This scenario mimics that used in [28]. A simulation arena of 1500 m $\times$ 300 m is used, with the transmission power of the radio interfaces set to 250 m. The random way-point mobility model is used, with a maximum speed of 20 m/s and varying pause times. Constant bit rate traffic of 64 byte packets, four packets per second, with ten flows between random pairs of nodes is used.

First, we compare the SAMPLE routing protocol to AODV and DSR with no packet loss added to the simulation. Fig. 5 shows the packet delivery ratio and transmissions-per-packet metrics as they vary with the level of mobility in the network. In this scenario, both AODV and DSR have near-optimal packet delivery ratios, while SAMPLE has between 1% and 4% worse packet delivery ratios, and all protocols have a near-minimal cost (in terms of network transmissions).

We introduce packet loss to the simulation in order to measure how well the different protocols adapt their operation to a network with lossy links. Radio interference is simulated by introducing random packet loss into the simulation. An NS-2 ErrorModel is used to drop packets both at the transmitter and receiver (each with half the rate shown in the results). This is a simplistic simulation of interference in the wireless network as packet loss is not introduced as a function of signal strength. However, it is indicative of the effect of lossy network links on the routing protocols.

Fig. 6 shows the performance of the routing protocols as the level of packet loss in the network increases. Data points shown are the average of at least 30 simulation runs with varying traffic and mobility scenarios. The SAMPLE routing protocol manages to maintain more optimal performance for packet loss levels at which AODV and DSR show significantly reduced performance. For packet loss rates of up to 20%, SAMPLE has packet delivery ratio above 85%, with only slight increase in the number of transmissions made per packet. At a 20% packet loss rate, however, AODV and DSR have packet delivery ratios of 60% and 10%, respectively, and produce more than double the number of radio transmissions for each packet sent compared to SAMPLE.

B. Adapting to Stable Links in a Metropolitan Area Manet

We have also evaluated the performance of SAMPLE against that of AODV and DSR in a network scenario based on a metropolitan *ad hoc* network. In this scenario, there is a subset of nodes in the network that are not mobile. The network scenario is motivated by the recent appearance of *ad hoc* networks designed to supply internet access to mobile nodes.

In this scenario, we anticipate that certain nodes in the network will be immobile for extended periods of time, resulting in stable links between them, and that the traffic patterns in the network will be concentrated on this subset of immobile nodes. In the experiments presented here we use three server nodes. Each client sends constant-bit-rate traffic to one of the servers at a rate of four packets per second. The number of client nodes in the network is varied in order to create congestion in the network. Fig. 7 shows the layout of the simulation arena. There are 33

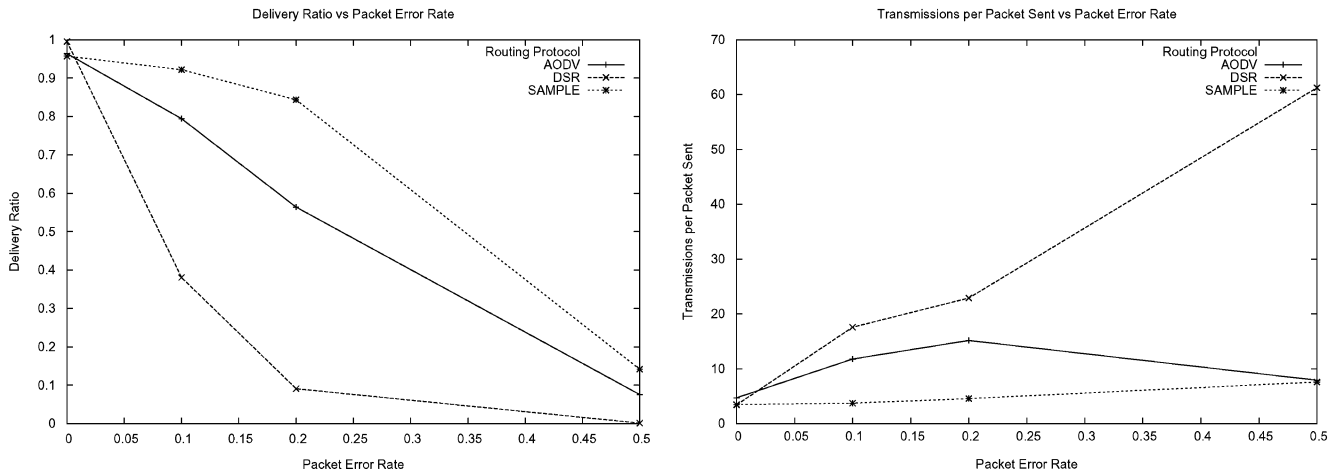

Fig. 6. In this experiment, packet loss is introduced into the random network scenario and the ratio of delivered packets to dropped packets degrades quickly for AODV and even more so for DSR. However, SAMPLE's ability to adapt routing decisions to changing network link quality means that at a packet error rate 0.2, it can still deliver 80% of the packets, while AODV and DSR can only deliver 55% and 9% of the packets, respectively.

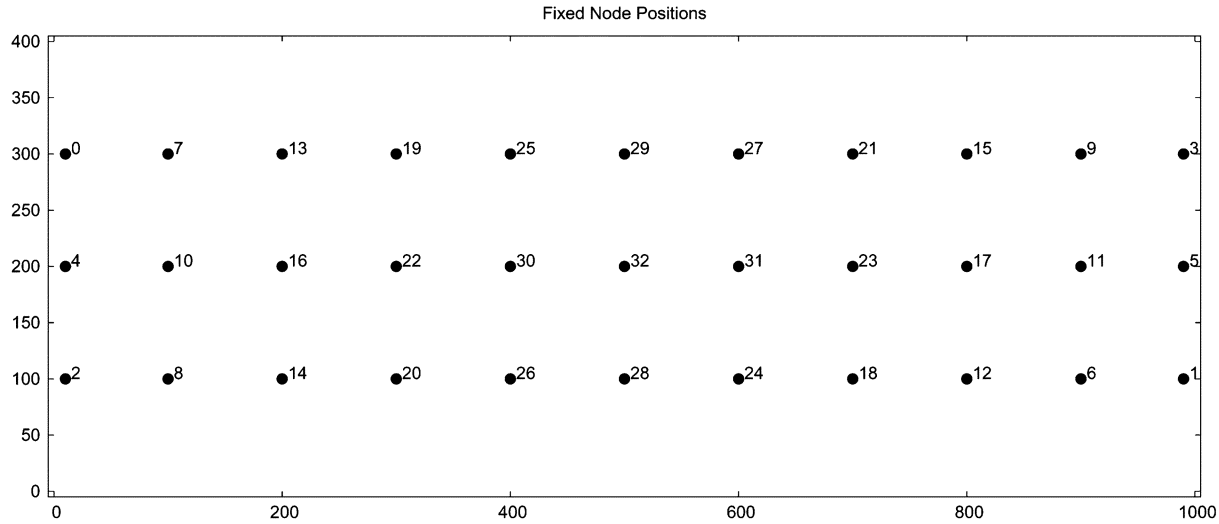

Fig. 7. Simulation arena, showing fixed node positions (transmission range is 100 m). The presence of stable nodes in the environmental property to which SAMPLE routing agents can adapt to improve routing performance.

fixed nodes in our simulations, and 50 mobile nodes. The three server nodes, nodes 30–32, are the fixed nodes at the centre of the simulation arena. The fixed nodes in the simulation provide stable links in the network which the routing protocols could exploit. SAMPLE's configurable parameters, see Section III-G, have been tuned to provide improved performance for this scenario, see [23] for more details. Fig. 10 shows the variation in performance of the three routing protocols as the number of clients in the network is increased. For these figures the packet size sent by clients was kept fixed at 64 bytes, sent three times per second. Fig. 9 shows the same experiment, this time with 512 byte packets. *Offered Throughput* is used in both figures to enable comparison of the results.

As the number of clients in the network is increased, the offered throughput to the routing protocols is increased. This in turn increases the level of packet loss and the amount of contention that the media access control (MAC) protocol must deal with. This increased congestion increases the number of failed MAC unicasts in the network. Figs. 8 and 9 show that this increased packet loss results in lower throughput and packet delivery ratios in AODV and DSR, but that SAMPLE is able to maintain high throughput and packet delivery ratios with high levels of packet loss.

In [29], it was demonstrated that for multihop 802.11 networks, the achievable throughput is significantly less than the transmission rate of the radio interfaces. The maximum achievable data throughput in an 802.11 *ad hoc* network is approximately 0.25 Mb/s (which [29] achieved using 1500 byte packets). Fig. 10 shows that SAMPLE manages to approach this limit in this scenario. This shows experimentally that using CRL, SAMPLE can meet the system optimization criteria of maximizing network throughput in the metropolitan area MANET scenario.

C. Discussion of Results

The results presented in the previous section show that AODV and DSR can only meet our routing optimization criteria when their assumptions of perfect radio links and a random network topology hold, see Fig. 5. SAMPLE, however, can meet or approach many of its system optimization goals in a changing

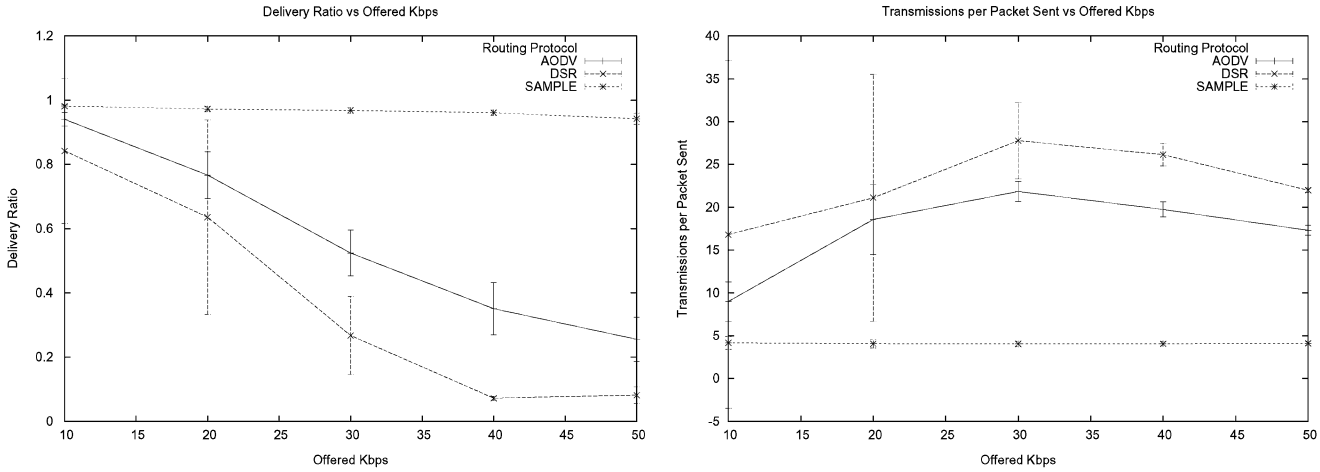

Fig. 8. This experiment shows the performance of the protocols in the metropolitan area MANET scenario with varying load of 64 byte packets. AODV and DSR cannot adapt their routing behavior to favor routes over stable links, instead preferring the shortest path from source to destination. In SAMPLE, positive feedback in link selection by routing agents means that the routing behavior of agents converges on the stable network links in the environment. SAMPLE maintains a near-optimal delivery ratio and using a minimal number of transmissions even at high offered throughput for 802.11 MANETs. (a) Delivery ratio, (b) Throughput.

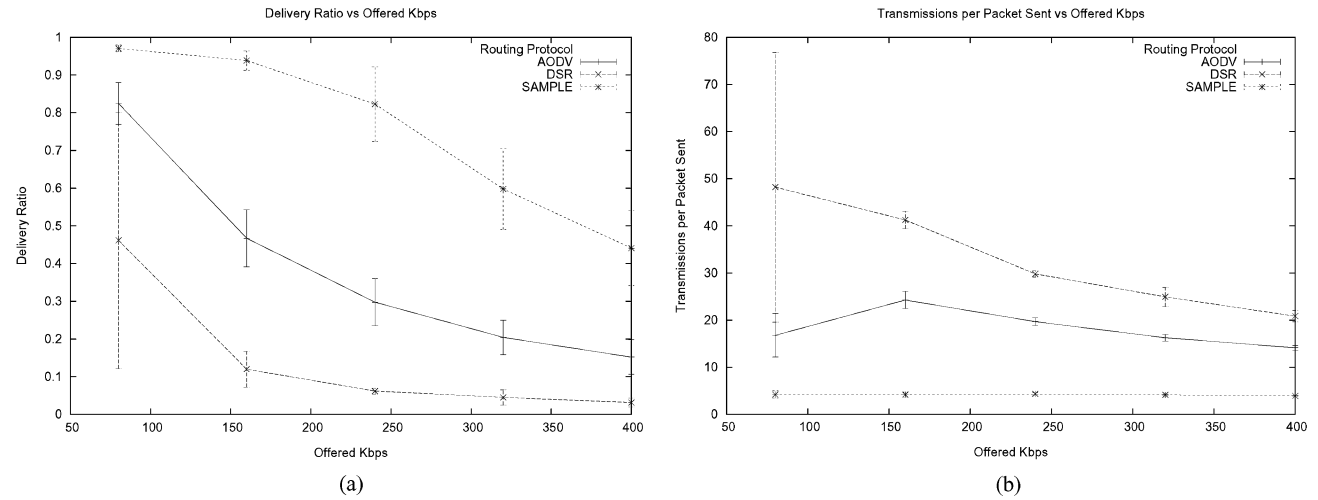

Fig. 9. This experiment shows the performance of the protocols in the metropolitan area MANET scenario with varying load of 512 byte packets. As in Fig. 8, routing agents in SAMPLE adapt their routing behavior to use stable routes, resulting in superior performance to AODV and DSR as the offered throughput is increased. (a) Delivery ratio. (b) Throughput.

MANET environment, as can be seen in maximizing network throughput in Fig. 10, minimizing the number of transmissions per packet sent in Figs. 5, 6, 8, and 9 and maximizing the ratio of delivered packets to dropped packets in Figs. 5 and 8.

We believe that the SAMPLE protocol performs better than AODV and DSR under changing and adverse network conditions for a number of reasons. First, in congested networks, or when suffering from interference, all links will have less than 100% reliability. In this situation, AODV and DSR generate increased routing traffic in response to dropped packets (Fig. 8, for example, shows a clear increase in traffic as congestion increases). By learning that not every dropped packet is not necessarily a broken link, SAMPLE avoids generating a large number of routing packets. Second, the retransmission of failed unicast packets in 802.11 does not change route costs for AODV and DSR, since their route costs are based on a hop-count metric [20], but in SAMPLE a failed unicast updates our state transition model for the network link. Nodes advertise changes in route costs (V values) to neighboring nodes, making it less likely that the link will be chosen in the future. This collaborative feedback enables routing agents to adapt their routing behavior to use more optimal routes to the destination. Third, in the metropolitan area MANET environment, there are a subset of the links in the network that are stable. In SAMPLE, the state transition models help identify the stable links in the network. Collaborative feedback adapts routing agent behavior to favor paths with stable links, producing the emergent stable routes in the network. A discrete model of network links, however, does not allow differentiation between multiple available links in this manner. Finally, in our experiments, the on-demand and opportunistic transfer of routing information (V advertisement) is an effective way to reduce the amount of control traffic generated in the network. SAMPLE produces much fewer route discovery and route maintenance packets than AODV and DSR.

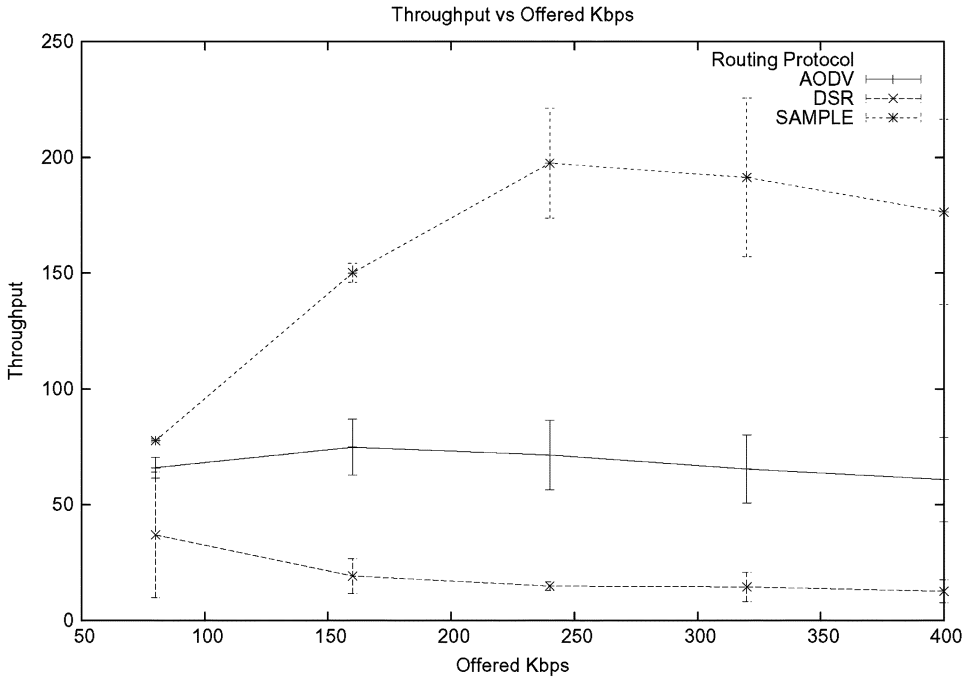

Fig. 10. At a varying load, 512 byte packets, SAMPLE delivers a data throughput of up to 200 Kb/s. This throughput approaches the theoretical limit of the throughput achievable in a multihop 802.11 mobile *ad hoc* network scenario.

V. Related Work

CRL was inspired by ideas from RL and swarm intelligence algorithms. Ant colony optimization (ACO) [6] defines a class of heuristic algorithms for solving DOPs and has a feedback mechanism similar to CRL, where agents learn from the successes of their neighbors. Positive feedback is provided by stigmergic pheromone trails, where one ant choosing an option increases the chance of other ants choosing that option in the future. However, ACO algorithms can easily converge to suboptimal solutions if the positive feedback is not carefully controlled. Similar to the decay model in CRL, an ACO algorithm can also include a pheromone decay procedure, in which the intensity of the pheromone trails decreases automatically over time.

There has also been some previous research in the use of reinforcement learning for multiagent systems and network routing, Barto's Elevator Group Control example [30] shows how multiple reinforcement learning agents can collaborate on a task with shared objectives using a global reinforcement signal, Q-Routing [31] for fixed networks showed how RL can provide superior routing performance to shortest-hop count protocols in congested fixed networks, Mariano presents a Distributed Q-Learning Algorithm that has been extended for multiobjective optimization problems [21] and team-partitioned opaque-transition RL (TPOT-RL) [32] describes a decentralized routing protocol. TPOT-RL's aim to provide network routing in decentralized environments most closely resembles SAMPLE's goals. It partitions value functions and introduces the notion of action-dependent feature spaces to guide behavior in uncertain environments, but similar to the other protocols, the routing experiments performed were based on a simple network topology.

VI. Conclusion and Future Work

We have shown experimentally in this paper that CRL has been used to build a complex adaptive distributed system, SAMPLE, that can adapt agent behavior in a changing environment to meet system optimization goals with agents using only local state information. CRL provides feedback models that map changes in an agent's environment and neighbors onto internal changes in the agent's policy using distributed model-based reinforcement learning. Positive and negative feedback are the key mechanisms that adapt an agent's behavior to both its neighbors and a changing environment. The problem of convergence to suboptimal solutions occurs in CRL and is an area that requires further research, however, we believe that in a constantly changing environment the ability of a distributed system to adapt its structure or behavior to more optimal configuration in a timely manner is as important a metric for evaluating the performance of the system as eventual convergence on some transient, optimal behavior. One of the limitations of CADS in general, compared with traditional distributed systems design techniques, is that strong system properties cannot be deterministically guaranteed using self-organizing techniques such as CRL. Also, given a top-down system-level optimization requirement or a complex adaptive behavior specification, there is no well defined process for the discretization of system optimization problems and the configuration of the models that will produce the required system behavior.

Future work on the SAMPLE protocol will involve tuning its configuration parameters in order to determine a set of reasonable default values for common MANET environments. We will also investigate whether improvements measured in tuning the configuration parameters can be learned online by the routing

protocol. This is a difficult problem as the system optimization criteria, such as offered throughput, that are used to evaluate changes in parameter tunings are not available to individual nodes during system operation. Finally, an interesting property of SAMPLE is that, as a result of the decay of routing information, there is a requirement for a continual critical mass of routing traffic to maintain knowledge of system structure, similar to dissipative systems [5].

Future work on CRL will involve applying the model to other optimization problems in distributed systems, such as load balancing in peer-to-peer networks. One challenging problem in applying CRL to fixed network systems is the implementation strategy for V advertisement. Further research is required to understand how the increased cost of advertisement will negatively impact the rate of collaborative learning and consequently the rate at which positive feedback accelerates adaptive changes in system structure or behavior.

Finally, we believe that engineering complex adaptive distributed systems represents a challenging new field of research and that CRL provides useful feedback models that can be used to build systems that adapt and optimize system behavior in dynamic, decentralized environments.

REFERENCES

[1] J. Dowling, E. Curran, R. Cunnhingham, and V. Cahill, "Collaborative reinforcement learning of automatic behaviour," in *Proc. 2nd Int. Workshup Self-Adaptive Automatic Comput. Syst.*, 2004, pp. 700–703.

[2] A. Montresor, H. Meling, and O. Babaoglu, "Toward self-organizing, self-repairing, and resilient distributed systems," *Future Directions Distributed Comput.*, vol. LNCS 2584, pp. 119–126, 2003.

[3] S. Camazine, J. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-Organization in Biological Systems*. Princeton, NJ: Princeton Univ. Press, 2003.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford Univ. Press, 1999.

[5] I. Prigogine and I. Stengers, *Order Out of Chaos*. New York: Bantam, 1984.

[6] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic," *New Ideas Optim.*, pp. 11–32, 1999.

[7] A. Montresor, H. Meling, and O. Babaoglu, "Load-balancing through a swarm of autonomous agents," in *Proc. 1st Workshop Agent Peer-to-Peer Syst.*, 2002, pp. 125–137.

[8] G. W. Flake, *The Computational Beauty of Nature*. Cambridge, MA: MIT Press, 2000.

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

[10] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[11] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, U.K., 1989.

[12] A. Moore and C. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Mach. Learning*, vol. 13, pp. 103–130, 1993.

[13] K. Doya, K. Samejima, K. Katagiri, and K. Kawato, "Multiple model-based reinforcement learning," *Neural Comput.*, vol. 14, no. 6, pp. 1347–1369, 2002.

[14] M. Appl and W. Brauer, "Fuzzy model-based reinforcement learning," in *Proc. 3rd Eur. Symp. Intell. Tech.*, 2000, pp. 211–223.

[15] M. Atighetchi, P. Partha, C. Jones, P. Rubel, R. Schantz, J. Loyall, and J. Zinky, "Building auto-adaptive distributed applications: The quo-apod experience," in *Proc. 23rd Int. Conf. Distributed Comput. Syst. Workshops*, 2003, pp. 104–110.

[16] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.

[17] A. Mikler, V. Honavar, and J. Wong, "Autonomous agents for coordinated distributed parameterized heuristic routing in large dynamic communication networks," *J. Syst. Software*, vol. 56, no. 3, pp. 231–246, 2001.

[18] Ad Hoc on Demand Distance Vector (AODV) Routing, C. Perkins. (1997, Nov.). [Online]. Available: http://citeseer.nj.nec.com/article/perkins99ad.html

[19] D. Johnson, D. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multihop wireless ad hoc networks," in *Ad Hoc Networking*. Reading, MA: Addison-Wesley, 2001, pp. 139–172.

[20] E. Curran and J. Dowling, "SAMPLE: An on-demand probabilistic routing protocol for ad hoc networks," Tech. Rep., Dept. Comput. Sci., Trinity College, Dublin, Ireland, 2004.

[21] C. Mariano and E. Morales, "A new distributed reinforcement learning algorithm for multiple objective optimization problems," in *Proc. Adv. Artif. Intell., Int. Joint Conf., 7th Ibero-Amer. Conf. Artif. Intell., 15th Brazilian Symp. AI*, 2000, pp. 290–299.

[22] H. Lundgren, E. Nordstrom, and C. Tschudin, "The gray zone problem in ieee 802.11b based ad hoc networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 6, no. 3, pp. 104–105, 2002.

[23] E. Curran, "Swarm: Cooperative reinforcement learning for routing in ad hoc networks," M.Sc. thesis, Dept. Comput. Sci., Trinity College, Dublin, Ireland, 2003.

[24] G. Gaertner and V. Cahill, "Understanding link quality in 802.11 mobile ad hoc networks," *IEEE Internet Comput.*, vol. 8, no. 1, pp. 55–60, Jan./Feb. 2004.

[25] IEEE Std. 802.11: Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications (1999). [Online]. Available: http://standards.ieee.org/catalog/olis/lanman.html

[26] *NS-2 Network Simulator*, 2003. Information Sciences Institute, Software Package.

[27] G. Flores-Lucio, M. Paredes-Ferrare, E. Jammeh, M. Fleury, and M. Reed, "Opnet-modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed," in *Proc. Int. Conf. Simul., Model., Optim.*, vol. 2, 2003, pp. 700–707.

[28] J. Broch, D. Maltz, D. Johnson, J. Hu, and J. Jetcheva, "A performance comparison of multihop wireless ad hoc network routing protocols," *Mobile Comput. Netw.*, pp. 85–97, 1998.

[29] J. Li, C. Blake, D. De Couto, H. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Proc. 7th ACM Int. Conf. Mobile Comput. Netw.*, 2001, pp. 61–69.

[30] R. Crites and A. Barto, "Elevator group control using multiple reinforcement learning agents," *Mach. Learning*, vol. 33, no. 2/3, pp. 235–262, 1998.

[31] M. Littman and J. Boyan, "A distributed reinforcement learning scheme for network routing," in *Proc. Int. Workshop Applicat. Neural Netw. Telecommun.*, 1993, pp. 45–51.

[32] P. Stone, "TPOT-RL applied to network routing," in *Proc. 17th Int. Conf. Mach. Learning*, 2000, pp. 935–942.

Jim Dowling received the B.A. and Ph.D. degrees in computer science from Trinity College, Dublin, Ireland.

He is a Lecturer in the Department of Computer Science, Trinity College. He manages national and international research projects, including the Digital Business Ecosystem IST project and a national project on software licensing for pervasive computing applications. His research interests are primarily in the areas of self-organizing distributed systems, automatic computing, and pervasive computing.

Eoin Curran received the B.A. degree in mathematics and the M.Sc. degree in networks and distributed systems from Trinity College, Dublin, Ireland, in 2002 and 2003, respectively.

He is currently a freelance Software Developer and Consultant, based out of Trinity College.

Raymond Cunningham received the B.A. degree in mathematics and the M.Sc. and Ph.D. degrees in computer science fdrom Trinity College, Dublin, Ireland.

He is a Research Fellow in the Department of Computer Science, Trinity College. He has published a number of peer-reviewed papers in the distributed systems area related to research carried out during both his M.Sc. and Ph.D. degrees. Generally, his research interests cover the area of mobile distributed systems, distributed systems optimization techniques, and adaptive middleware.

Vinny Cahill received the B.A., M.Sc., and Ph.D. degrees from Trinity College, Dublin, Ireland.

He is an Associate Professor in the Department of Computer Science, Trinity College, where his research addresses middleware and programming language support for distributed computing. He has lectured at Trinity College since 1988 and was elected a Fellow of the College in recognition of research achievement in 1999. He is currently investigating dependable sentient computing for applications, ranging from intelligent vehicles to outdoor smart spaces.