Agent-based Distance Vector Routing: A Resource Efficient and Scalable approach to Routing in Large Communication Networks

Kaizar A. Amin and Armin R. Mikler

Department of Computer Science
University of North Texas
Denton, Texas 76203, USA
{*amin, mikler@cs.unt.edu*}

Abstract

In spite of the ever-increasing availability of computation and communication resources in modern networks, the overhead associated with network management protocols, such as traffic control and routing, continues to be an important aspect in the design of new methodologies. Resource efficiency of such protocols has become even more prominent with the recent developments of wireless and ad-hoc networks, which are marked by much more severe resource constraints in terms of bandwidth, memory, and computational capabilities. This paper presents an Agent-Based approach to Distance Vector Routing that addresses these resource constraints. Agent-Based Distance Vector Routing (ADVR) is a resource efficient implementation of Distance Vector Routing that is fault tolerant and scales well for large networks. ADVR draws upon some basic biologically inspired principles to facilitate coordination among the mobile agents that implement the routing task. Specifically, simulated pheromones are used to control the movement of agents within the network and to dynamically adjust the number of agents in the population. The behavior of ADVR is analyzed and compared to that of traditional Distance Vector Routing.

Key words: Distance-Vector Routing, resource efficient, intelligent mobile agents, ADVR.

1 Introduction

The unprecedented pervasivness of network access and the associated growth of communication networks represents a challenge to traditional distributed routing algorithms. The amount of network resources in the form of bandwidth, memory, and processing power that are consumed by these algorithms

is directly related to the size of the network (or autonomous systems). That is, traditional routing algorithms do not scale well with increasing network dimensions. In fact, it is the lack of scalability of these mechanisms that forces hierarchical structuring of a large network into autonomous domains. It has been observed, that the message overhead due to routing table updates increases drastically as the size of the autonomous system increases (Malkin and Streenstrup 1995). This increase in message activity is intrinsic to the implementation of most routing algorithms that are in use today, and is necessary to ensure that changes in routing cost are propagated throughout the network. Both, Distance Vector and Link State routing algorithms manifest a distributed version of shortest path algorithms (Bertsekas and Gallager 1987) designed for graphs (i.e., Bellman-Ford and Dijkstra).

In existing networks, the importance of fast route discovery and low routing delays surpasses the requirement of low resource overhead. Hence, aggressive messaging is deemed essential to quickly propagate local information from individual routers, thereby enabling other routers to utilize this information in routing decisions. It is the resource overhead incurred by this massively concurrent messaging that limits the scalability of these routing algorithms. Even though we witness an ever-increasing availability of network resources in conventional networks, the tremendous increase in network traffic makes it necessary to re-visit the fundamental design of current routing methodologies to find ways to limit the need for excessive messaging. Particularly in view of recent developments in mobile ad-hoc networks, which are characterized by limited bandwidth, memory, and computing power, it is imperative to find new ways of reducing resource overhead associated with routing algorithms. The goal is to devise a simple, resource efficient, scalable routing algorithm that discovers optimal routes expediently yet does so with *bounded* message activity. Towards this goal, this paper proposes the formulation of a new routing strategy that exploits the intelligent mobile agent paradigm. In comparison to ongoing research efforts that pursue the design of new routing paradigms, which exploit concepts such as reinforcement learning, this paper addresses the issue of propagating routing information in the network. Specifically, this paper focusses on the design and evaluation of an agent-based Distance-Vector Routing algorithm that facilitates scalability, resource awareness, and fault tolerance. The effort is motivated by two conjectures, which have been validated through a number of carefully crafted experiments.

Conjecture 1 *It is possible to bound the degree of message concurrency of distance-vector routing without significantly affecting the convergence behavior of the algorithm.*

Conjecture 2 *It is possible to dynamically control and effectively regulate the degree of message concurrency without centralized control or global knowledge of the state of the network.*

Although DVR-class algorithms like the distributed Bellman-Ford are simple to implement, they can suffer from the *routing loops* and the *counting to infinity* problem (Rajagopalan and Faiman 1989). However, there are a wide range of Distance Vector-based algorithms that eliminate temporary and permanent routing loops and avoid the counting to infinity problem altogether (Rajagopalan and Faiman 1989; Cheng et al. 1989). This paper aims at reducing the message complexity of conventional DVR-class algorithms. It does not aim at solving the looping problem and counting to infinity associated with them. Ongoing research focusses on implementing the agent based approach to certain *Loop-Free* routing algorithms, thereby making ADVR *loop-free*, *resource efficient*, and *scalable*.

The following section summarizes some of the research effort in agent-based routing during recent years and highlights principle approaches. The design of Agent-Based Distance Vector Routing (ADVR) is discussed in Section 3. We will revisit Conjectures 1 and 2 in Sections 4 where we present the experimental analysis of ADVR. Section 5 concludes the paper with a summary and direction for future work in the area of agent-based network-centric algorithms.

2 Mobile Agents in Routing

Intelligent Mobile Agent is a term that describes the concept of mobile computing or mobile code (Bradshaw; Fugetta et al. 1998). The appeal of the mobile agent paradigm is quite alluring - mobile agents roaming the network could search for or distribute information, meet and interact with other agents or remain bound to a single host or node. In general, an agent manifests four distinct characteristics, namely, *intelligence, communication, autonomy,* and *mobility*. Intelligence is the ability of agents to adapt their actions to circumstances brought upon by the dynamics of the system (or network). Communication is the property whereby the agents collaborate or coordinate their actions by the means of explicit or implicit exchange of information. Autonomy allows agents to make decisions and act upon them without the explicit control of a user. Last but not least, mobility is the property that makes agents conducive for distributed systems and network applications, as it allows the agent to migrate among the constituent nodes of the environment.

Most of the work in agent-based network routing is biologically inspired and based on insect colonies (Di Caro and Dorigo 1997; White 1997). It relies on the principles that individual insects exhibit a simple behavior while collective communities of these insects exhibit complex problem solving capabilities. Considerable research has been conducted in mapping the foraging activities of ants to routing and network management activities of mobile agents. Real ants are represented as artificial agents that traverse the network collect-

ing specific information from their environment and coordinate their actions through *Pheromones*. On the basis of this information the agents make several decisions to adapt their behavior (*Reactive Agents*) and/or change the existing environment affecting their future actions (*Proactive Agents*). Ant Based Control (ABC), is a recent network-centric algorithm, that utilizes an ant-based approach for routing and network management in circuit switched networks (Schoonderwoerd et al. 1997). Other approaches that exploit agents for routing and network management schemes in circuit switched networks exploit *Swarm Intelligence* (White 1997; White and Pagurek 1998). Such an approach exploits the concept of multiple colonies of agents coexisting and in some cases coordinating with each other working towards independent goals. AntNet applies the idea of deploying agents for routing in packet switched networks (Di Caro and Dorigo 1997). The algorithm generates mobile agents (artificial ants) at regular intervals at different nodes in the network. These agents select a random destination in the network; traverse the network to reach the destination and on their way back to the source node collect routing information. Although AntNet is an interesting approach for static networks with a good adaptive property, its application in dynamic networks is yet to be explored.

The performance of any agent-based system will depend on its agent population. Although significant research has been conducted on agent-based systems, little consideration has been given to the importance of agent population in dynamic networks. Most of the agent-based implementations assume a fixed number of agents in the network. Certain systems create agents at regular intervals and destroy them once the required task is accomplished (Di Caro and Dorigo 1997). Although the latter approach provides some degree of flexibility it does not adapt to sudden changes in the network topology. It is difficult to know, a *priori*, the optimal degree of concurrency or the number of agents required in the system since it depends on the system dynamics and availability of resources. Therefore, autonomous multi-agent systems should be capable of adapting to their environment and changing the agent population to an appropriate number with respect to resource availability.

3 Agent-based Distance Vector Routing (ADVR)

All Distance Vector Routing (DVR) algorithms exchange a *metric* that represents the distance from a node n_i to any destination n_j (Hedrick 1988). Distance is a generalized concept and may include transmission delay on a link, monetary cost of traversing a link, security level of links/nodes, or reliability measures. In most implementations of DVR, this information (metric) is exchanged among adjacent nodes in the form of triggered updates, which are initiated whenever a change in the routing table occurs in one of the

nodes in the network. After receiving the update information from a neighboring node, a node n_i updates its own routing table in the following manner (Hedrick 1988; Malkin and Streenstrup 1995):

$$D(i,j) = \begin{cases} 0 & \forall \;\; i = j \\ min[d(i,k) + D(k,j)] & \forall \;\; n_k \text{ adjacent to } n_i \end{cases} \tag{1}$$

where $D(i,j)$ represents the metric of the best route from node n_i to node n_j currently known to n_i. $d(i,k)$ represents the cost of traversing the link from node n_i to node n_k. Any node n_i that receives $D(k,j)$ from a neighbor n_k, computes $D(i,j)$ based on equation(1) and integrates this value in its routing table. When the routing table of n_i is updated, the changes are propagated to all neighbors, which in turn perform the same algorithm. Therefore, an update in one routing table can cause a sequence of update messages in nodes throughout the entire network.

In ADVR, the exchange of the metrics and the process of route discovery moves from the nodes to the agents. Hence in this approach, route discovery and updates are manifested in the movement of agents carrying routing information from one node to another rather than the propagation of individual update messages. Agents in ADVR can be formally described as: $\Lambda(i,x,y,R_x,\gamma)$, where Λ is an Agent with ID i migrating from node n_x to node n_y, carrying the routing table R_x and using the migration strategy γ to move among adjacent nodes. R_x is a subset of r_x, the routing table of n_x (See Figure 1).

In ADVR, agents start at arbitrary nodes and migrate to adjacent nodes using γ. Upon arriving at a node n_y, an agent $\Lambda(i,x,y,R_x,\gamma)$ updates the routing table R_y based on the following equation:

$$D(y,j) = \; min(D(y,j), [d(y,x) + D(x,j)]) \;\; \forall \; n_j \text{ in } R_x \tag{2}$$

where $D(x,j)$ is an entry in R_x. After performing the update, the agent selects R_y and migrates to an adjacent node using migration strategy γ.

At every node the agent has to make a decision regarding the routing data it would carry to the next node. This decision plays an important role in providing a resource efficient solution with ADVR. If the agent carries the entire routing table available at each node, it would incur excessive overhead in transferring redundant data. On the other hand, if the agent selected a subset of total routing data available at the node, it would unnecessarily delay the propagation of important routing information. The flexibility adopted by the agents in selecting the routing data reflects the inherent degree of intelligence acquired by it. It is important for the agents to execute certain book keep-

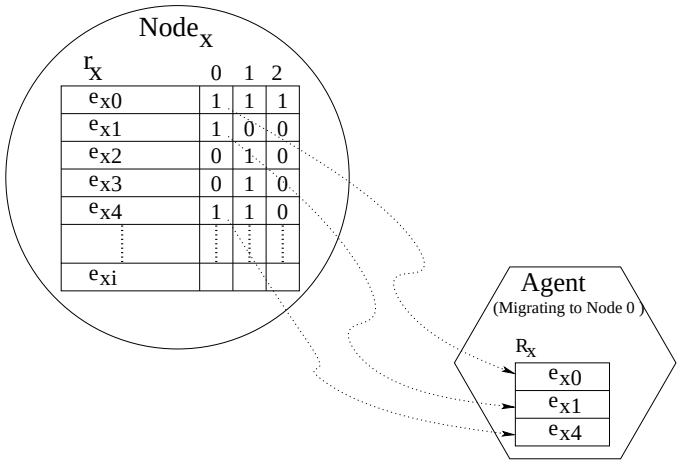


Fig. 1. Selection of Routing Table Entries by the Agent

ing functions at every node whereby it would limit the routing data carried by it to a minimum without affecting the propagation of important routing information.

To reduce the amount of information propagated in ADVR, agents refrain from transferring complete routing tables whenever possible. Agents identify routing table entries that have been modified, yet have not been transferred to a particular neighbor. Associated with every entry e_{xi} in routing table r_x is a vector V_{xi} of boolean flags for each of the neighbors nodes of n_x. $\mid V_{xi} \mid = \mid H_x \mid \ \forall e_{xi}$, i.e., the size of each of the V_{xi} is equivalent to the size of the neighborhood of a node n_x, H_x. Upon selecting a neighbor n_y of the current node n_x, an agent $\Lambda(i, x, y, R_x, \gamma)$ will carry only those entries e_{xi} in R_x for which $V_{xi}[y] == 1$. The agent copies each entry e_{xi} that is to be transferred to neighbor node n_y to its data segment, and sets the corresponding boolean flag $V_{xi}[y] == 0$. At startup, all the flags are set, i.e., $V_{xi} := 1 \ \forall e_{xi}$. Further, any routing table entry e_{xi} that is modified by an agent will have all its flags V_{xi} reset. To facilitate robustness and fault tolerance, all flags $V_{xi} \ \forall e_{xi}$ will expire after some time ΔT and reset (i.e., $V_{xi} := 1$). Resetting flags after ΔT enables nodes to re-transmit routing updates that may have been lost during previous transmission attempts.

3.1 Agent Migration Strategy

The mere replacement of messages with agents and the design of mechanisms that facilitate an optimized selection of information to be transferred between network nodes are insufficient to guarantee adequate performance of route discovery and maintenance. Agents must collaborate and coordinate their actions in order to strike a balance between resource efficiency and performance of route discovery and maintenance. In ADVR, a population of individual

agents migrates among the nodes in the network to obtain and distribute routing costs. The size of this agent population is much smaller than the number of messages that concurrently traverse the network in conventional distance vector routing. Even though each agent can be viewed as an individual, the movement of all agents must be coordinated in order to avoid agents to form clusters in some parts of the network while neglecting to migrate to other parts. This coordination manifests itself in what we refer to as migration strategy (γ).

An agent follows the migration strategy to determine the next node to visit (i.e., a neighbor of the current node). It is imperative that an agent-based system should carefully choose its migration strategy as there is no consensus on a single global optimal strategy. A method suitable for one system can produce unwanted side effects for other systems. The simplest migration strategy is a random selection among all neighbors with uniform probability (Amin et al. 2001; Minar et al. 1998; Minar et al. 1999; Kramer et al. 1999). Although simple, the random nature of this strategy could severely degrade the performance of ADVR, as certain areas of the network may remain unvisited for long periods.

Another candidate for agent migration strategy is the depth-first search of the network based on network information carried by the agents (Minar et al. 1998). This scheme requires that agents maintain a migration history carrying records of their previous node visitations. Systems implementing such a scheme could benefit from population of agents exchanging their migration history, thereby informing other agents of recently visited nodes. Multiple agents on the same node exchange their migration history and make migration decisions based on the combined migration history to visit an unvisited node. However, it was observed that by exchanging their migration history, all the agents on a given node contain the same global history thereby making similar decisions resulting in clustering of agents in specific parts of the network while leaving other parts unvisited (Minar et al. 1998). Further, carrying the migration history as a part of agent payload increases the agent size imposing an overhead on the system resources.

A biologically inspired migration strategy uses a population of naive, autonomous agents (simulating biological insects) performing complex tasks using *Stigmergy* (Schoonderwoerd et al. 1997; Di Caro and Dorigo 1997). Stigmergy is the mechanism for naive individuals to communicate with each other via local changes in the environment. Most of the migration strategies based on this scheme simulate foraging activities of ants. Although this strategy has shown impressive results in certain applications, it tends to favor migration patterns, preventing uniform distribution of agents throughout the network. Hence, it may not be a feasible solution for systems such as ADVR that require agents to explore the entire network with equal probabilities.

Fig. 2. Migration Strategy using Edge Pheromones

The migration strategy employed in ADVR combines the strengths of both of the above mentioned schemes. Our approach exploits the stigmergetic feature of the insect colonies and the exploratory feature of the depth-first-search. That is, with very little knowledge of the network, the agents communicate with each other via the environment and perform the depth-first-search on the network as a community. The agents do not carry any network information as a part of their payload. They simply indicate their presence leaving pheromone trails. Pheromone is a volatile chemical, decaying exponentially, released by insects in the environment indicating their presence. Ants use pheromone trails to follow the path of the successor ant. While the ant pheromones are used to attract other members of the community (Schoonderwoerd et al. 1997; Di Caro and Dorigo 1997; White 1997), in our approach, pheromones repel other agents. An agent traversing a link xy from node n_x to n_y deposits a pheromone on xy. Another agent migrating from n_x will chose a link with the weakest pheromone value thereby migrating to a least recently visited region of the network. For example, Figure 2 shows that the agent arriving at node A (time = t_0) selects the edge with least pheromone value. It also shows that the while traversing the edge, the agent deposits pheromone trails on it preventing other agents to immediately follow itself. This paper refers to this class of pheromones, that assist in agent migration strategy as *Edge Pheromones*. It can be observed that such an approach exploits the stigmergetic behavior of insect colonies and avoids the clustering of agents in specific regions of the network. By changing and retrieving information from the environment as opposed to carrying the network information (Minar et al. 1998; Minar et al. 1999), agents in ADVR impose minimal resource requirements.

3.2 Controlling the Agent Population

While an appropriate migration strategy may facilitate the performance of route discovery and maintenance, it contributes little towards solving the problem of resource efficiency, which is the central theme of this paper. There is no strict definition for resource efficiency. In fact, resource efficiency is rather relative to the amount of resources that are available, the complexity of the task to be performed, and the level of performance (i.e., in terms of conver-

# *Agents*	*Convergence Time*		*Average Routing Overhead*	
	Measured (ms)	Normalized	Measured (KB/ms)	Normalized
10	150	1.0	4.73	0.30
15	85	0.57	7.91	0.51
20	78	0.52	9.70	0.63
25	59	0.39	12.27	0.80
30	47	0.31	15.44	1.0

Table 1
Convergence Time and Routing Overhead for Different Agent Population

gence, quality of routes, routing cost etc) expected from the algorithm. For agent-based routing, all routing traffic for route discovery and maintenance, is carried by the constituent agents in the system. Hence, it is the size of the agent population, which manifests the resource overhead. In fact, if the size of the population is static, it represents an upper bound on the degree of message concurrency, and hence the resource overhead. The message activity in conventional routing algorithms (DVR) is in principle unbounded, however, in ADVR it is limited by the number of agents that constitute the current agent population.

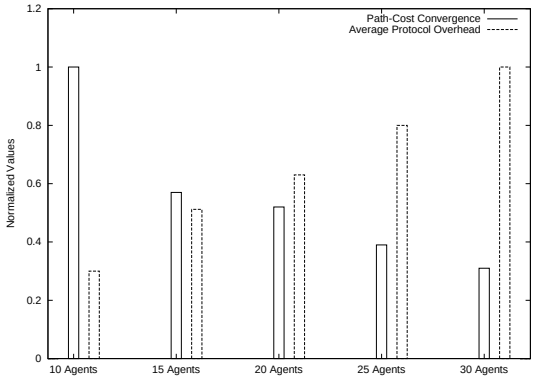


Fig. 3. Comparison of Routing Overhead with Path-Cost Convergence

A large population of agents would increase the parallelism of ADVR resulting in an improved convergence (Amin et al. 2001). However, it is extremely important to analyze the agent overhead in terms of bandwidth consumption and computational cycles. Increasing the agent population will improve the path-cost convergence of the algorithm at the expense of increased resource demands. Table 1 displays the convergence time and average routing overhead for different agent population. Figure 3 plots the normalized convergence time and average routing overhead for multiple agent population. The average routing overhead was calculated by dividing the cumulative routing overhead encountered in ADVR till convergence by the convergence time. It can be seen from Figure 3 that the convergence time and routing overhead are inversely

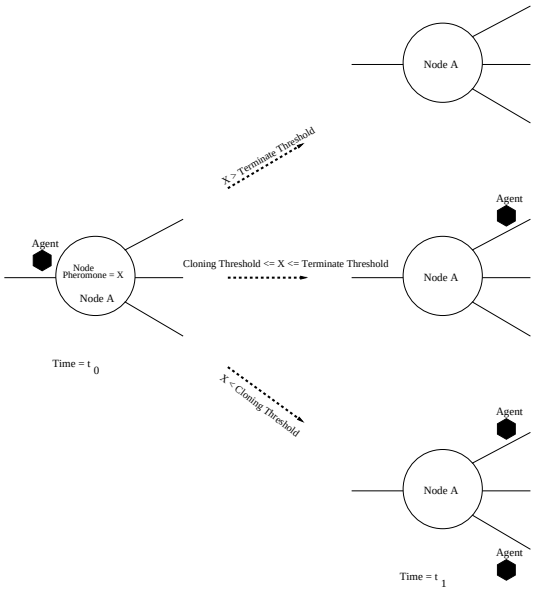


Fig. 4. Population Control using Node Pheromones

related to each other. It was observed that a large agent population has a significantly lower convergence time owing to its parallelism. Although low convergence time is desirable, it has other side effects. A larger agent population has a significantly larger average message overhead because a substantial number of agents traverse the network concurrently imposing resource requirements on the network. For scalable systems, the average overhead should be as low as possible. Therefore it is apparent that significantly large agent populations, resulting in high average overhead hamper the scalability of ADVR. On the other hand, a very small agent population will hinder the performance of ADVR, in terms of convergence times and reactivity to the dynamic behavior of networks. Hence, we shall strive for an *optimal* agents population for a given network that results in acceptable path-cost convergence without producing excessive average overhead. It can be observed from Figure 3 that for such an optimal agent population the difference in the normalized convergence time and normalized average overhead would be minimum. Hence in the given example 15 agents would result in an optimal trade-off between convergence time and resource overhead. However, the unpredictable behavior of dynamic networks makes it very difficult to estimate a *priori*, the value of this *optimal* population. Thus, it is necessary that an adaptive multi-agent system dynamically alters the agent population in response to its resource availability.

Changing the agent population dynamically in response to its environment (resources) is a non-trivial issue in the absence of a central controller. Individual agents lacking a bird's eye view of the system are unable to make global assessments regarding the environment in terms of resource availability. Therefore, it requires a high degree of coordination among agents to analyze the global

environment from local information available at nodes. To facilitate such a coordination, our approach exploits the stigmergetic properties of agents. Mobile agents with minimum cognitive capabilities communicate with each other using pheromones, establishing an infrastructure that assists them in assessing their environment. Pheromones that aid the agents in population control are referred to as *Node Pheromones* to distinguish them from Edge Pheromones (see Section 3).

Whenever an agent visits a node it deposits a pheromone which is simulated by timed tokens. The potency of the Node Pheromones is represented as decay functions expressed by the equation $e^{-\lambda(\Delta t)}$, where λ represents the degree of volatility of the pheromone and Δt is the time since the deposition of the pheromone. Using this equation the agents can extract the value of the Node Pheromone at a given time and calculate the inter-agent arrival time at that node. An agent visiting a node n_x at time t_2 calculates the value of the Node Pheromone that was deposited at time t_1 using the equation $e^{-\lambda(t_2-t_1)}$ (see Figure 4). If this value is above a certain *Termination Threshold* (Ψ) and the agent did not produce any routing update on n_x, the agent *terminates* itself. However if the Node Pheromone value has decayed below a *Cloning Threshold* (Ω), the agent *clones* itself. Before leaving n_x, the agent deposits additional Node Pheromone at time t_2. This approach controls the agent population based on the inter-agent arrival time expressed as a function of the Node Pheromone. If the inter-agent arrival time is small ($e^{-\lambda(\Delta t)} > \Psi$) and the agent produced no updates in the existing routing table entries, it implies an excessive number of agents in the system leading to the self termination of the agent. On the other hand, if the inter-agent arrival time is large ($e^{-\lambda(\Delta t)} < \Omega$), it implies there are a sub-optimal number of agents in the system resulting in agent cloning. However if $\Omega \leq e^{-\lambda(\Delta t)} \leq \Psi$, the agent neither clones nor terminates. Terminating requires the agent to destroy its instance along with its code and data segments. Cloning requires the agent to create another instance of itself with same attributes and privileges. The volatility of Edge Pheromones can be controlled by changing the *Degree of Volatility*, λ in $e^{-\lambda(\Delta t)}$. Pheromones with higher values of λ (Degree of Volatility) have a higher rate of decay.

ADVR implementing a dynamic agent population may start with a single agent or an arbitrary number of agents. Nevertheless, the agents coordinate themselves and converge to a particular range of population. This range represents an optimal population that results in an optimal performance of the network based on the availability of resources. This range however depends on the values of Ψ, Ω, and λ. An adaptive system should adjust these values dynamically based on its resource availability.

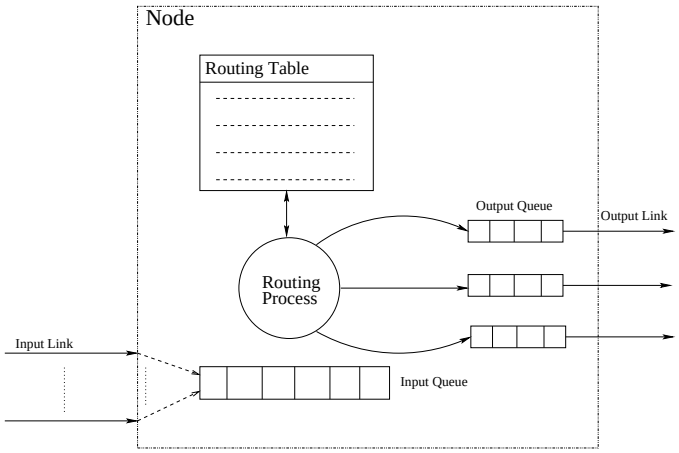


Fig. 5. Simulation Model for DVR

4 Experimental Analysis of ADVR

This section discusses our simulation environment and experimental results. A series of experiments have been conducted to support our conjectures mentioned in Section 1.

4.1 Simulation Environment

To analyze the properties of agents in DVR, an event driven simulator has been constructed. The simulator is based on an object-oriented paradigm and includes methods for DVR, single agent ADVR and multi-agent ADVR. A network is represented as a graph $G(V, E)$ that is generated by a *graph generator*. Every node in the graph represents a store-and-forward router, which is further characterized by a limited buffer space and processing speed. A link connecting two nodes is characterized with certain link capacity. Following the example of a particular implementation of DVR, namely the Routing Information Protocol (RIP) (Hedrick 1988), we assume a variable sized packet with a maximum of 512 *bytes*. Each packet consists of a 4 *byte* header and variable payload. Each entry in the routing table occupies 20 *bytes* in the payload. For fairness, both, DVR and ADVR, use the same packet characteristics.

Figure 5 shows the simulation model for DVR. Every node has an input queue whereby all incoming packets are queued. The average service rate for the input queue depends on the processing rate of the router which can be in the range of $300000 - 500000$ packets per second (pps) (Cisco; Powerrail). Every node has a routing process which inspects the input queue. The routing process is responsible for routing data packets to the appropriate output interface as well as maintaining the routing table. Each outgoing link (interface) is associated with an output queue whose service rate is controlled by the transmission

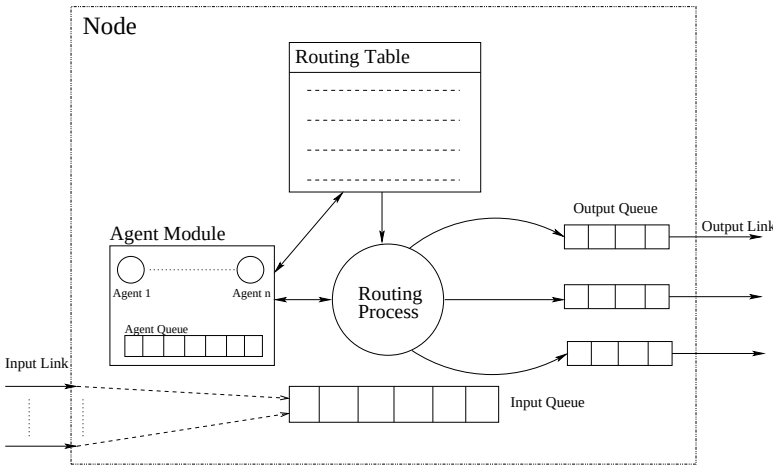


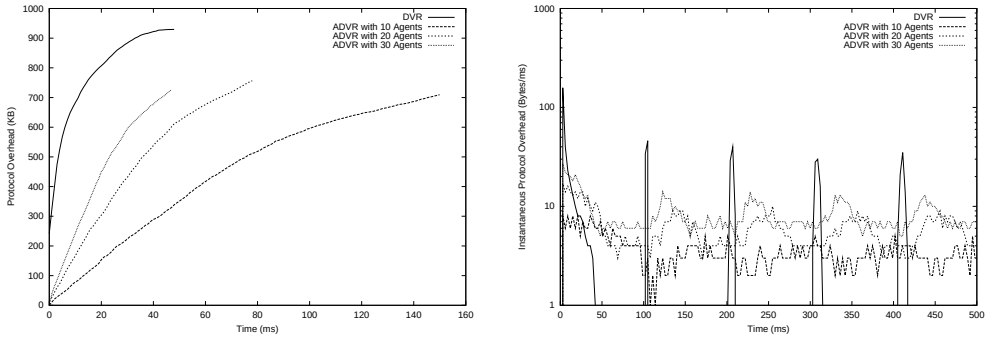
Fig. 6. Simulation Model for ADVR

rate of the link. The transmission rate of the link is given by $1/T_t$, where T_t is the transmission time for one packet. For our experiments we have assumed the link capacity to be 10 Mbps, thereby giving us an average transmission rate of approximately 2500 packets per second (pps). Hence it is clear that a majority of the queuing would occur at the output queues due to its slow service rate.

Figure 6 shows the simulation model for ADVR. It has an additional module for agent management which provides a runtime environment for agents. The agent management module provides the framework for agent transmission, reception, population control, and route maintenance functions. All agent related packets (agent code and agent data) are forwarded to the agent management module where they are queued in the agent queue. Agents (agent code) are activated by the agent management module from the agent queue and receive their respective data (agent data). Depending on the data received by the agents, they update the routing table. On completion of its task, the agent is transmitted by this module to the next node using the migration strategy discussed in Section 3.1. The routing process is responsible for routing incoming regular data packets to the appropriate interfaces using the routing table maintained by the agents.

4.2 Results

Experiments were conducted on a medium sized 40 node network with an average degree of 7. The results in this section represent the mean over multiple random experiments and different random graphs of the same type. The analysis does not cover the performance of the network after convergence of the routing tables, unless otherwise mentioned. The analysis of results in this section have been conducted with reference to certain definitions.



(a) Protocol Overhead until Convergence

(b) Instantaneous Protocol Overhead

Fig. 7. Comparison of Overhead in DVR and ADVR

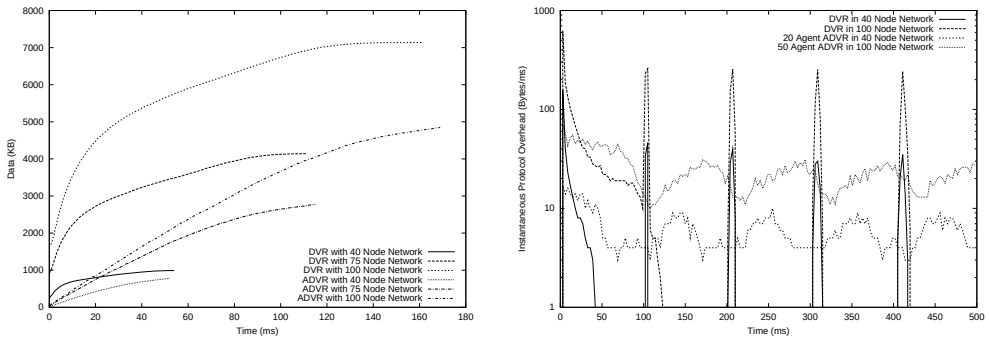
Definition 1 Instantaneous Routing Load *(IRL) of the routing algorithm at a given time is defined as the routing load or routing messages traversing the network at that instant.*

Definition 2 Path-Cost convergence *of the network is defined as the condition when every node has an optimal (shortest path) route to every other node in the network.*

Definition 3 Route discovery *is the process, whereby every node in the network obtains a route for every other node in the network.*

4.2.1 Analysis of Message/Agent Overhead

As mentioned earlier DVR attributes its sensitivity to the large number of routing messages exchanged by the nodes. The number of concurrent routing messages in a network implementing DVR is a function of time and network size. However, the number of concurrent routing messages in ADVR is constant and manifested in the number of constituent agents. Since the number of agents in the network can be adjusted as per resource availability, ADVR can provide a highly scalable solution to the routing problem. To validate Conjecture 1, Figure 7 compares the routing overhead incurred in DVR and ADVR. Figure 7(a) displays the cumulative routing data exchanged by the nodes in the network until convergence of the routing algorithm. It is apparent that DVR incurs a significant message overhead due to its overly-reactive nature. As explained in the previous section, although the total routing data exchanged among nodes until convergence is approximately the same for different agent populations, the average routing load is high for larger agent populations. In order to evaluate the scalability of any routing algorithm, it is essential to analyze the *Instantaneous Routing Load (IRL)* incurred in the algorithm. For an algorithm to be scalable, the *IRL* should be as low as possi-



(a) Routing Load over Different Network Sizes

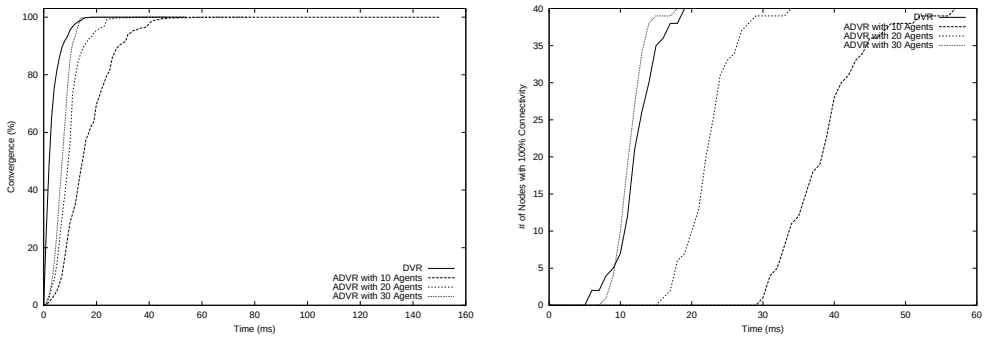
(b) Instantaneous Protocol Overhead

Fig. 8. Comparison of Overhead in DVR and ADVR for Different Network Sizes

ble and without large variation. Figure 7(b) shows the average *IRL* for DVR and ADVR in a time window of 3 milliseconds for a simulated time of 200 milliseconds. In order to depict the behavior of DVR, we have simulated a timed update every 100 milliseconds. A timed update in ADVR is manifested by the resetting of the routing flags. It can be observed that *IRL* in DVR is certainly higher than in ADVR. DVR is characterized with periods alternating activity and inactivity. Although periods of inactivity produces an *IRL* of 0 KB/ms, it is the periods of activity in DVR that produce an excessive *IRL*. A timed update or any change in network topology suddenly increases the *IRL* in DVR due to the broadcast storming problem. With ADVR, agents continuously traverse the network (with or without data segments), hence, there are no periods of inactivity. Therefore, unlike DVR, the *IRL* never reduces to 0 KB/ms. Nevertheless, *IRL* in ADVR is low, fairly stable, and proportional to the number of agents in the system.

Figure 8 compares the overhead involved in the two routing approaches over multiple network sizes and analyzes their scalability. Figure 8(a) shows the cumulative protocol overhead incurred in the network until convergence of the routing algorithm. It is apparent that even though ADVR converges comparable to DVR over multiple networks by varying the agent population, its cumulative overhead is always lower than that of DVR. Further, the non-scalability of DVR is evident from Figure 8(b). An increase in network size produces a excessive increase in the *IRL*. Such a sharp increase in routing traffic can overflow transmission queues, thereby can contribute to jitter, packet loss, or congestion in large networks implementing DVR. Among other things, the non-scalable characteristics of DVR restricts its use in large networks. Conversely, ADVR exhibits its scalability by incurring a marginal increase in the *IRL*, proportional to the increase in number of agents.

Our simulation model assumes that the agent code segment consumes 100 bytes of the IP packet. In view of the main objective of this paper, to reduce the



(a) Path-Cost Convergence (b) Route Discovery

Fig. 9. Comparison of Convergence and Route Discovery

resource overhead, it is imperative to consider the structure of the agents. If the agent code segment is excessive, the agent will consume significant amounts of resources in terms of bandwidth, memory, and computing. Conversely, if the code segment is severely restricted, it may be impossible to supply some of the agents with sufficient intelligence to optimize their task performance. In order to reduce the size of the code segment, it is possible to supply the agents code as pre-loadable software modules at each node. The behavior of these modules is controlled by a set of parameters that are provided by the agent upon arrival at that node. These parameters will replace the code segment that is otherwise carried by the agents, resulting in smaller *light-weight* agents that may consume less bandwidth. Nevertheless, this approach does not eliminate the problem, it does only shift the resource overhead from the link (i.e., bandwidth) to the node (i.e., computation).

4.2.2 Analysis of Path-Cost Convergence and Route Discovery

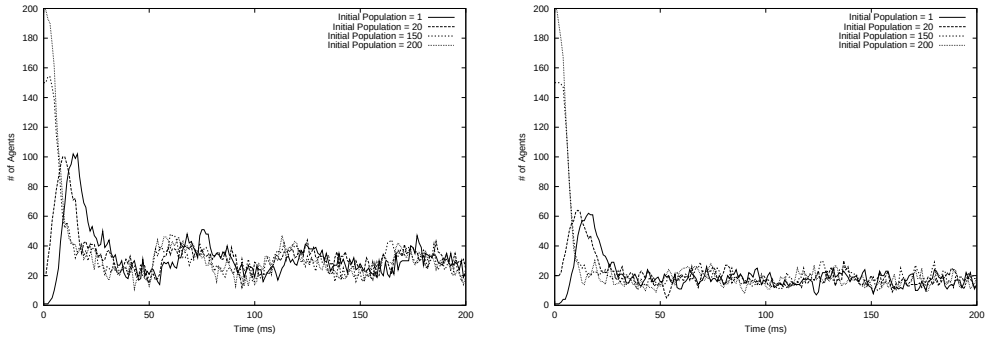
It is the characteristics of DVR that every change in the routing table of an individual node is broadcasted to its immediate neighbors. Additionally, the entire routing table of every node is broadcasted periodically to each of its neighbors. These events occur asynchronously making use of message concurrency, which in turn causes DVR to be highly reactive to small changes. Hence, any change in a single routing table has a cascading effect initiating a sequence of broadcasts throughout the network. Such an aggressive parallelism in DVR results in bursts of update messages within the network. Conversely, ADVR implements controlled parallelism characterized by the number of agents in the network. Although ADVR can replicate the behavior of DVR, routing information, encapsulated in the agent payload, is generally propagated to only one neighbor. Such an approach restricts the outburst of routing packets due to small changes. Nevertheless, controlled parallelism reduces the sensitivity of the algorithm, thereby exhibiting a relatively slow convergence. Figure 9(a)

shows the aggressive nature of DVR inherent in its rapid path-cost convergence when compared to the moderate yet comparable convergence behavior of ADVR. It can be shown, that in a static network, a single agent can achieve the correct convergence of routing tables at all nodes in the network, provided that it uses an appropriate migration strategy, which allows for complete traversal of the network. Nevertheless, a single agent is insufficient to complete this task in a time that is comparable to that of concurrent messaging i.e. DVR. Hence, a population of agents will have to be deployed. These agents implicitly cooperate, thereby accelerating the process of route discovery and path cost convergence.

Route discovery plays an important role in the performance of communication networks. It is crucial to evaluate any routing algorithm with respect to the speed at which every node in the network obtains a route for every other node in the network. Even if these routes are sub-optimal, they provide a benchmark to measure the availability of the network to be used by other applications. Figure 9(b) depicts the number of nodes that acquire complete connectivity to all other nodes in the network over time. It is observed that the aggressive parallelism in DVR facilitates quick assimilation of network connectivity for DVR. On the other hand, a small population of constituent agents, restrained in their concurrency are insufficient to discover routes as rapidly as DVR. Route discovery in ADVR can be improved to outperform DVR by escalating the agent population, thereby increasing the degree of concurrency. Even though increasing the number of agents in the network increases the resource consumption by agents, it is extremely low when compared to DVR. It is imperative to note that the performance of ADVR in terms of route discovery is greatly affected by the migration strategy adopted by the agents. A detailed comparison of the migration strategy is presented in (Amin et al. 2001).

4.2.3 Analysis of Agent Population in ADVR

As mentioned earlier, agents are the carriers of information in ADVR. Hence, the agent population in the network determines the resource overhead. A static agent population represents an upper bound on the degree of message concurrency, and the hence resource overhead. All the above experiments assume a fixed agent population, however Figure 10(a) shows the the effects of dynamic agent population control mechanisms using *Node Pheromones*. As explained in Section 3, values of Ψ, Ω, and λ have to be manipulated manually in order to exercise effective control on agent population. It was observed that irrespective of the initial population, the system converges to a stable number of agents in the system. Networks initialized with a small number of agents escalate the agent population to a certain value thereby improving the path-cost convergence of the network. On the other hand, networks initialized with a large number of agents realize the per-agent overhead and continuously re-



(a) Variance in Agent Population with $\Psi = 0.8$, $\Omega = 0.3$, and $\lambda = 1$

(b) Degree of Volatility $\lambda = 0.5$

Fig. 10. Dynamic Control of the Agent Population

duce the population until it reaches a stable number. Figure 10(b) displays the variance in agent population with Node Pheromones having reduced degree of volatility (λ). Low values of λ significantly stabilizes the variation in agent population. Although less volatile pheromones reduces the variance in population, it also reduces the sensitivity of the system to react and rapidly adapt to sudden changes in the environment.

The agent population control exhibits a flexible, fault tolerant mechanism whereby loss of agents in the network do not impose any performance penalties on ADVR. This is extremely important in an adaptive, self-controlling agent based system in the absence of centralized controller. Such a control mechanism gives agents the autonomy to escalate their population on detection of link/node failures, thereby rapidly propagating the new information throughout the network without suffering from the broadcast dilemma. *Regular* agents, on detecting a link/node failure clone themselves to produce multiple *Auxiliary* agents that are entrusted with the task of propagating the information regarding link/node failure throughout the network. Although the concept of *Auxiliary* agents is very interesting, its validity needs to be verified. Ongoing research is investigating issues such as routing loops created/terminated by these *Auxiliary* agents.

5 Conclusion and Future Work

This paper describes a distance vector routing scheme based on the mobile agent paradigm – Agent-based Distance Vector Routing. One of the major disadvantages of conventional implementations of distance vector routing algorithms is that their corresponding resource overhead is generally unbounded. In the proposed ADVR, the messages are replaced by a population of agents.

The corresponding message activity is thus bounded by the number of constituent agents. However, by limiting the number of agents in order to control resource overhead, the degree of concurrency which the algorithm can employ is restricted as well. We have conducted a number of experiments to analyze the performance of an agent-based distance vector routing scheme. In particular, we have focused on the *Instantaneous Routing Load (IRL)*, scalability, path-cost convergence, and route discovery of ADVR and have compared the results with that of DVR. We have also looked at the distributed manipulation of the agent population in the network.

It was observed that as per our conjecture, the *IRL* in ADVR is considerably low and scalable when compared to DVR. It was also verified that the *IRL* for ADVR has a very small variation as opposed to DVR which results in sharp spikes of routing loads for periods of activity. Further, it was verified that although DVR is aggressively reactive in path-cost convergence and route discovery, ADVR with a substantial number of agents can compete with the performance of DVR. To validate our conjecture, a dynamic and distributed mechanism was set up using pheromones to manipulate the number of agents in the network in order to reduce the overall protocol overhead.

The results of this paper are expected to provide alternative ways to design and implement resource efficient routing algorithms. Particularly in view of the recent developments in ad-hoc and mobile networks, agent-based solutions to routing may be alluring as the such system are inherently fault tolerant. While the main objective of this paper is on routing, agent-based solutions are deemed suitable for many other network centric applications. Network monitoring, for instance, could take advantage of the mechanisms developed as part of this approach. The dynamic population control mechanisms facilitate the design of adaptive solutions for monitoring processes or sensors that undergo complex dynamics and cannot rely on statically designed schedules and itineraries. The distributed control mechanisms described above may help to coordinate the actions of otherwise autonomous agents to find a global monitoring strategy. The management of large networks and distributed computing environments can take advantage of the mobile agent paradigm and the tools designed for this research. By exploiting mobility and intelligence, agents facilitate system fault tolerance through the expedient discovery of redundant communication paths and/or alternative computing platforms. Resource management and distributed cluster scheduling in support of scientific applications in Grid computing may take advantage of such properties. In general, we expect that this paper and its corresponding results will motivate the design of agent-based solutions for large scale system-level applications.

References

[Amin et al. 2001] K. Amin, J. Mayes and A. Mikler, *Agent-based Distance Vector Routing*. Proceedings of the Third International Workshop, MATA 2001, Montreal, Canada, August 2001.

[Bertsekas and Gallager 1987] D. P. Bertsekas and R. G. Gallager, *Data Networks*. Prentice-Hall, 1987.

[Bieszczad et al. 1998] A. Bieszczad, B. Pagurek, and T. White, *Mobile Agents for Network Management*. IEE Communications Surveys, 1998.

[Bradshaw] J. M. Bradshaw, *Software Agents*. AAAI Press, Menlo Park, California/The MIT Press.

[Cheng et al. 1989] C. Cheng, R. Riley, S. Kumar, and J.J. Garcia-Luna-Aceves, *A Loop-Free Extended Bellman-Ford Routing Protocol without Bouncing Effect*. Computer Communications Review, Vol. 19(4):224–236, September 1989.

[Cisco] Cisco, Product Specification of *Cisco 7300 Series Internet Routers*.

[Di Caro and Dorigo 1997] G. Di Caro and M. Dorigo, *AntNet: a mobile agents approach to adaptive routing*. Tech. Report, IRIDIA/97-12, Universit Libre de Bruxelles, Belgium.

[Di Caro and Dorigo 1998a] G. Di Caro and M. Dorigo. *AntNet: Distributed Stigmergetic Control for Communications Networks*. Journal of Artificial Intelligence Research, 1998.

[Di Caro and Dorigo 1998b] G. Di Caro and M. Dorigo, *An Adaptive Multi-Agent Routing Algorithm inspired by Ants Behavior*. Proceedings of PART98 - 5th Annual Australasian Conference on Parallel and Real-Time Systems, pages 261–272. Springer-Verlag, 1998.

[Fugetta et al. 1998] A. Fugetta , G. P. Picco, G. Vigna, *Understanding Code Mobility*. IEEE Transaction, Software Engineering 24(5), 342-361, 1998

[Garcia-Luna-Aceves and Murthy 1997] J. J. Garcia-Luna-Aceves and Shree Murthy, *A Path-Finding Algorithm for Loop-Free Routing*. IEEE/ACM Transactions on Networking, Vol. 5, No. 1, February 1997.

[Hedrick 1988] C. Hedrick, *Routing Information Protocol*. RFC 1058, June 1988.

[Jaffe and Moss 1982] J.M. Jaffe and F.M. Moss, *A Responsive Routing Algorithm for Computer Networks*. IEEE Trans. Commun., Vol. COM-30, No. 7, July 1982.

[Kramer et al. 1999] K. H. Kramer, N. Minar, and P. Maes. *Tutorial: Mobile Software Agents for Dynamic Routing*. Mobile Computing and Communications Review, 1999.

[Kurose and Ross 2001] J. F. Kurose and K. W. Ross, *Computer Networking, A Top Down Approach Featuring the Internet*. Addison-Wesley, 2001.

[Magendanz et al. 1996] T. Magendanz, K. Rothermel, and S. Krause, *Intelligent Agents: An Emerging Technology for Next Genereation Telecommunications*. The Proceedings of INFOCOM 96, San Fransisco CA, March 1996.

[Malkin and Streenstrup 1995] G. S. Malkin and M. E. Steenstrup, *Distance-*

Vector Routing. In M. E. Steenstrup, editor, *Routing in Communications Networks*, pages 83–98, Prentice Hall, 1995.

[Mikler and Chokhani 2001] A. R. Mikler and V. Chokhani, *Agent Based Wave Computation: Towards Controlling the Resource Demand*. Proceedings of the International Workshop IICS (Innovative Internet Computing Systems) 2001, Ilmenau, Germany, June 2001.

[Minar et al. 1998] N. Minar, K. H. Kramer and P. Maes, *Cooperating Mobile Agents for Mapping Networks*. Proceedings of the First Hungarian National Conference on Agent Based Computing, 1998.

[Minar et al. 1999] N. Minar, K. H. Kramer and P. Maes, *Cooperating Mobile Agents for Dynamic Network Routing*. Proceedings of the Software Agents for Future Communications Systems, Springer-Verlag, 1999, ISBN 3-540-65578-6.

[Motwani and Raghavan 1995] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[Powerrail] Packet Engines, Product Specification of *PowerRail 1000 Routing Switch*.

[Rajagopalan and Faiman 1989] B. Rajagopalan and M Faiman, *A New Responsive Distributed Shortest-Path Routing Algorithm*. ACM SIGCOMM, 1989.

[Schoonderwoerd et al. 1997] R. Schoonderwoerd, O. Holland, and J. Bruten, *Ant-like Agents for Load Balancing in Telecommunications Networks*. Proceedings of the First International Conference on Autonomous Agents, pages 209–216. ACM Press, 1997.

[Tennenhouse and Wetherall 1996] D. L. Tennenhouse and D. Wetherall, *Towards an Active Network Architecture*. Proceedings of Multimedia Computing and Networking, San Jose CA, 1996.

[White 1997] T. White, *Routing with Swarm Intelligence*. Technical Report SCE97 -15, Systems and Computer Engineering Department, Carleton University, September, 1997.

[White and Pagurek 1998] T. White and B. Pagurek, *Towards Multi-Swarm Problem Solving in networks*. Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS '98), July 1998.