RESOURCE EFFICIENT AND SCALABLE ROUTING USING INTELLIGENT MOBILE AGENTS

Kaizar Abdul Husain Amin, B.E.

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

May 2003

APPROVED:

Amin, Kaizar Abdul Husain, *Resource Efficient and Scalable Routing using Intelligent Mobile Agents.* Master of Science (Computer Science), May 2003, 74 pp., 1 table, 16 figures, 51 titles.

Many of the contemporary routing algorithms use simple mechanisms such as flooding or broadcasting to disseminate the routing information available to them. Such routing algorithms cause significant network resource overhead due to the large number of messages generated at each host/router throughout the route update process. Many of these messages are wasteful since they do not contribute to the route discovery process. Reducing the resource overhead may allow for several algorithms to be deployed in a wide range of networks (wireless and ad-hoc) which require a simple routing protocol due to limited availability of resources (memory and bandwidth). Motivated by the need to reduce the resource overhead associated with routing algorithms a new implementation of distance vector routing algorithm using an agent-based paradigm known as *Agent-based Distance Vector Routing* (ADVR) has been proposed. In ADVR, the ability of route discovery and message passing shifts from the nodes to individual agents that traverse the network, co-ordinate with each other and successively update the routing tables of the nodes they visit.

ACKNOWLEDGMENTS

Several people have supported me during the course of my Masters program. I would like to take this opportunity to thank them all. Firstly, I would like to thank my parents who have supported me emotionally throughout my academic career. They have always encouraged me to realize my dreams. I would also like to thank Dr.Mikler who has been my adviser, friend, and guardian in the last three years. He is the epitome of a perfect *teacher*. I am grateful to him for all his valuable time spent to teach me the essence of honest research and the value of commitment. I feel very fortunate to have worked with him and look forward to working with him again in the coming years. Thanks to my wife Aneesa for making me realize that there is more to life than a Masters thesis. She has been my inspiration and has helped me maintain my focus. I would also like to thank Gaurang, Nikhil, Glyco, and Prasanna for being my family away from home and helping me through difficult times. Thanks to the members of my thesis committee for their important time and valuable comments.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The research conducted as a part of this thesis addresses two issues. Firstly, it presents a new routing algorithm using mobile agents that is inherently scalable and resource efficient to be applied to large communication networks. Secondly, it validates the application of mobile agents to intrinsically distributed and asynchronously parallel problems. Further, it introduces some novel concepts for multi-agent coordination and autonomous population control mechanism to enhance the performance of the agent-based solution. The inferences derived from this thesis are supported by the results from carefully designed experiments.

1.1 Intelligent Mobile Agents

Software agents, softbots, intelligent agents, and mobile agents are terms that originally evolved from the domain of artificial intelligence [18]. However, they are applied to solve problems in a wide range of domains including distributed object architectures, expert systems, distributed algorithms, adaptive self-learning systems, collaborative communities, peer-to-peer communication, and security infrastructures. For the rest of this thesis we use the terms agents, mobile agents, and software agents interchangeably. Software agents are distinctively different from software programs in that they can be highly specialized to meet the requirements of the end-user. Further, agents are proactive in nature whereby, they take the initiative in undertaking a task rather than awaiting a pre-defined condition. Finally, agents are autonomous and adaptive thereby targeting their resources towards a goal oriented approach.

There is no formal and succinct definition for mobile agents. However, in general an agent can be described as a long-lived goal-oriented computational entity. It operates autonomously under a certain user controlled policy migrating among host nodes continuing the same thread of execution and acting on behalf of a user [28]. In other words, an agent encapsulates a thread of execution, a code segment, and a data segment. The code segment is customized to achieve the end-task desired by the user. At any network node, the agent may suspend its thread of execution, migrate to another node of its choice, and resume the thread of execution from the point where it was previously suspended. Intuitively, one may argue that an agent mechanism is similar to a process migration system [17]. However, it must be noted that in a process migration system, the decision to migrate the process and the choice of the new host machine is invested in the migration system, whereas, in an agent system these decisions are taken by the agent itself. Further, one can compare agent execution with the execution of Java applets. In this regard it is essential to note that an applet is a small piece of code that is downloaded from the server and executed on the client machine in its entirety, whereas an agent is a segment of code that is executed on a remote machine, resuming its execution from the point where it was previously suspended.

Based on the above discussion, it is reasonable to mention that agents may contain several (or all) of the following characteristics:

- *Intelligence:* It enables the agent to interact with its environment and translate the knowledge gathered by it towards its individual goal. A *reactive* agent can sense the environment and adapt its behavior thereby leading to an optimal progress. On the other hand, a *proactive* agent can sense the environment and modify the environment itself thereby implicitly affecting its future actions.

- *Autonomy:* It allows the agent to make decisions and act upon them without explicit control of an external user. Autonomy is an important property that distinguishes agents from conventional programs by attributing agents with a decision making capability.

- *Communication:* It allows the agent to exchange information gathered by it to the end-user. Although the agent operates autonomously, it is essentially a piece of code that acts on behalf of a user, and hence must have a capability to effectively communicate with the user.

- *Collaboration:* It provides a framework where individual agents can communicate among themselves and take shared responsibilities in accomplishing a common tasks. This is the most important characteristics in multi-agent systems (MAS) [15].

- *Mobility:* It allows agents to migrate from one network terminal to another. It is the attribute that make agents conducive for distributed systems and network applications.

- *Goal-oriented:* It is the characteristics that helps the agent to modify its immediate local actions thereby always moving towards a global goal.

It is apparent that mobile agents can be used for a large number of intrinsically distributed applications. Agent systems are used in several cases to replace the conventional client-server paradigm. Client-server communications require a handshake protocol including request submission, security negotiations, parameter exchanges, and result delivery. A mobile agent based approach can considerably reduce network bandwidth by avoiding multiple connections and communications between the client and server by migrating an agent to the server that only returns back with the desired results. In high performance computing applications [16] that generate large sets of data, it is a more elegant solution to migrate the code that analyzes this data to the data itself thereby significantly reducing bandwidth latency. A mobile agent is also a considerable candidate to replace conventional remote procedure call (RPC) [49] in several cases. RPC is a protocol that allows a user or a program to request a service from another remote machine. In RPC communications, the client has to remain online until it receives a final reply from the remote terminal. If the user disconnects, the entire RPC communication must be restarted. An agent based approach will benefit a mobile user that is intermittently connected to the network. The client can create an agent with the desired task and disconnect from the network. The agent will then migrate itself to the remote host, accomplish the task, return to the host machine, and hold the results till the client connects to the network again. Mobile may agents also offer an efficient solution to complex problems involving extensive workflows and itineraries. Complicated workflow dependencies between distributed tasks can be expressed as an agent itinerary and dispatched with the customized agent [7]. The agent exploits it resources, migrating to individual services making important information exchanges, and finally returning to the end-user with the end-results.

For mobile agents to migrate across heterogeneous platforms, adapt to its environment, communicate with its users, and collaborate with other agents it must be supported by an underlying agent hosting infrastructure. In other words, an agent can perform only those functions on a host that are supported by the agent hosting environment. It is the function of the hosting environment to translate the agent instructions into local machine instructions, provide the agents with the appropriate resources, negotiate security policies with the agent, and protect the agent from a malicious user. For any host to be agent-aware, it must have the hosting environment deployed on it. There are certain characteristics that are required from any agent hosting system. It must provide a security model that is flexible enough to protect both the host and the agent. The system should be capable of hosting different types of agents written in various languages. Since agents perform autonomous actions, it is the the responsibility of the hosting system to provide with robust, fault tolerant services capable of recovering the agent and the associated data in the event of a fatal error. The most popular agent hosting environments in use are Tracy [8], IBM Aglets ® (IBM Corp., www.ibm.com) [23], TACOMA [25], and AgentTCL [21].

1.2 Network Routing

Routing is the process of discovering, selecting, and maintaining paths from one node to another and using these paths to deliver data packets. It is an important aspect of network communication since it effects many other characteristics of network performance. An efficient routing mechanism can get significantly complex due to the fact that it may involve all the nodes in the network. Routing algorithms can be broadly classified under the following categories:

- Central versus Distributed
- Static versus Dynamic
- Shortest-Path versus Heuristic

1.2.1 Central versus Distributed

In central routing, a central or master node is assigned the task of computing the routes between different nodes in the network. This node is also responsible of communicating these routes to individual nodes in the network. A central routing approach has several disadvantages and is rarely used in practice. It offers a single point of failure, where the entire routing algorithm fails if the master node stops functioning. Secondly, it results in a hot-spot in the network. A hot-spot is a small region in the network that experiences high traffic and congestion, therefore becoming the bottleneck of network performance. Since all the routing information is generated by the master node, the network region around this node experiences large volumes of data.

Distributed routing algorithms on the other hand divide the entire routing process into smaller sub-tasks that are executed in parallel at individual nodes. The network nodes communicate among themselves independent of any central controller, thereby resulting in a more robust system. Distributed routing algorithms are deployed in most of the networks and will be considered exclusively in this thesis.

1.2.2 Static versus Dynamic

Static routing, also referred to as oblivious routing depends on fixed routes between nodes. The routes selected between end-nodes are predetermined by the network managers at the time of network startup. Any change in the routing policy is reflected by the managers by explicitly changing the routes at every node in the network. This scheme requires constant human monitoring to update the routes reflecting an up-to-date network.

Dynamic routing, also known as adaptive routing automatically utilizes information about the network characteristics such as link costs, traffic congestion, and network failures to deduce routing paths at run-time. It requires minimum human control and is used in most networks due to its implicit flexibility.

1.2.3 Shortest-Path versus Heuristic

A shortest-path routing algorithm chooses the minimal cost paths between end-nodes. Minimum costs between nodes are computed from either the local knowledge or global knowledge about the network gathered by the individual node. Shortest-path routing algorithms such as distance vector and link state algorithms are most widely used in contemporary networks and will be the center of discussion in this thesis.

Heuristic routing extracts data such as packet delay and quality of service parameters from incoming packets over a period of time to optimize the routing process based on some heuristic function [36]. Such class of algorithms are application specific and have very limited use.

1.3 Dynamic Distributed Shortest-Path Routing

As mentioned earlier, this thesis concentrates on dynamic distributed shortest-path routing algorithms. All the routing algorithms in this class can be conveniently divided into: *Link State Algorithms* and *Distance Vector Algorithms*.

1.3.1 Link State Algorithms

The link-state approach is a brute force approach, where each node maintains its view of the entire network. The node locally executes some algorithm, generally based on the Dijkstra's shortest-path algorithm, to compute the shortest-path routes from itself to every other node in the network. In order to maintain a consistent and up-to-date view of the network, the nodes exchange topological information among themselves by means of flooding. Every node floods the information about its current view of the network to its peers. Upon receiving an update from other nodes, the algorithm for shortest-path routes is locally run to modify the routing table appropriately. Since, every node is required to maintain information about each of the $O(n^2)$ edges in the network, it requires a storage space of $O(n^2)$, where n is the number of nodes in the network. The computational complexity of a link-state algorithm is a function of the shortest-path algorithms executed locally at the nodes. The most common algorithm to compute shortest paths in a link-state routing algorithm is the Dijkstra's shortest-path algorithm which has an algorithmic complexity of $O(n^2)$. Some examples of the link-state protocol are the new Arpanet routing protocol [32], Open Shortest Path First (OSPF) [40], and IS-IS (Intermediate System - Intermediate System) [42].

The popularity of link-state algorithms can be attributed to the fact that they are free from long term routing loops. Since these algorithms are loop-free, they are marked by a considerable faster convergence when compare to the loop-prone counterparts, distance-vector algorithms. Their responsiveness is further improved by the flooding mechanism employed by them to exchange routing information. However, this improvement in convergence and responsiveness in the link-state algorithms is not without a price. As mentioned earlier, all link-state algorithms are characterized by large memory requirements at the node, $O(n^2)$, to store the network information. Further, the message overhead in link-state algorithms is aggravated by the fact that every topological change perceived by the node is flooded throughout the network. The high message and computational complexity incurred by these algorithms make them less scalable. These complexities can be reduced by dividing the network into a hierarchical structure. Nevertheless, a hierarchical approach makes the link-state algorithms highly complex, thereby making them difficult to configure and maintain.

1.3.2 Distance Vector Algorithms

Distance-vector routing algorithms, also known as Distributed Bellman-Ford algorithms, have the shortest-path computations based on the distributed version of the Bellman-Ford equation [4]. In distance-vector algorithms, every node maintains a distance table, where it stores the shortest-path (distance) information to all the destinations in the network from each of its neighbors. The node, locally applies the Bellman-Ford equation to the distance table to compute the routing table that provides the shortest distance to each of the destinations and the next hop used to reach that destination. The nodes maintain an up-to-date version of their distance and routing tables by exchanging the routing tables with their respective neighbors at regular intervals or when the distance to a destination has changed. Every node requires a storage space of $O(nE)$, where E is the average degree of the network. Some popular examples of the distance-vector routing algorithms are the old Arpanet routing protocol [32], Routing Information Protocol (RIP) [22], Exterior Gateway Protocol (EGP) [34], Border gateway Protocol (BGP) [45], Diffusion Update Algorithm (DUAL) [19], Extended Bellman-Ford Algorithm [9], and the Optimal Algorithm (OP) [44].

The distributed version of the Bellman-Ford algorithm described above is marked by its simplicity and ease of maintenance. Since a change in the routing table is broadcasted only to its immediate neighbors and not to all the nodes in the network, the distance-vector algorithms have a considerably lower message overhead than their respective link-state counterparts. Further, the computational complexity at each node is $O(n)$. However, the distributed Bellman-Ford class algorithms have several major drawbacks. The convergence of these algorithms after a link or node failure can be extremely slow due to the looping problem and/or the counting to infinity problem. The looping problem arises if a path from a node to any destination visits the same node twice. The counting to infinity problem arises when a node continuously increments the distance to a destination till the distance reaches some predefined infinity.

Certain distance-vector routing algorithms implement special mechanisms in addition to the basic Bellman-Ford equation in order to reduce or avoid the looping and counting to infinity problem. RIP implements mechanisms such as split horizon, hold down, and poison reverse to reduce the looping problem. However, none of these methods are able to solve the problem completely. BGP exchanges the entire routing path in its routing messages, thereby avoiding loops. Several other algorithms use synchronization mechanisms among the nodes to avoid the looping problem at all instances [19, 24, 33]. Some of them use synchronization among nodes over multiple hops, whereas some synchronize over a single hop. Nevertheless, synchronization mechanisms require additional protocol overhead and make the algorithm more complex. Certain algorithms such as the Extended Bellman-Ford and the Optimal Algorithm (OP) store additional information in their routing tables to avoid the looping problem. They store the predecessor information, which is the second-to-last node in the path to the destination [9, 44]. With this additional information, the nodes are able to break any long-term routing loops formed in the network.

It is apparent that there is a wide variety of different routing algorithms available in literature. The routing algorithm used for a particular network entirely depends on the requirements and characteristics of the network and has to be decided by the network managers, as there is no algorithm that satisfies all the requirements.

1.4 Mobile Agents in Routing

Most of the work in agent-based network routing is biologically inspired and based on insect colonies. It relies on the stigmergy principles of insect colonies, where individual insects such as ants 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 collecting specific information from their environment and coordinating their actions through pheromones. A Pheromone is a volatile chemical released by insects in the environment to indirectly communicate with the other members of the community. 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).

Schoonderwoerd et al. implemented a network-centric algorithm, that utilizes an ant-based approach for routing and network management in virtual circuit switched networks [46]. This approach was implemented in symmetric telephone networks where identical costs were associated with the links in both directions. Routing agents travel the network from a source (s) to a destination (d). On their way, the agents collect information about the path (in terms of routing hops). At every intermediate node through d, the agents update the routing table entry for the s with the information collected on the journey from s. It is based on a valid assumption that in a symmetric network, the characteristics of a route from a source to a destination is the same as its reverse path. Individual nodes maintain a probabilistic routing table rather than a fixed shortest-path routing table, whereby a node can select any neighbor to route the traffic with a non-zero probability. Routing agents traversing the network update the routing probabilities for a neighbor based on the data collected by them during their journey. Hence, as per the principles of reinforcement learning, a good, non-congested route will be more likely to be used by nodes to forward their data traffic.

The approach adopted by Schoonderwoerd et al. provided load-balancing in symmetric circuit switched networks. However, in asymmetric packet switched networks the estimate of the distance of a route in one direction is different than that in the reverse direction. Therefore, an agent moving from a source s to destination d cannot accurately update the routing table at d based on the information gathered during its journey. For this reason, AntNet introduces the concept of two types of agents, namely, the forward agent and the backward agents [11, 12, 13]. At regular intervals, every node generates a forward agent that selects a destination at random and moves towards that destination using the probabilistic routing tables of intermediate nodes. On their way agents collect the path information. At the destination, the forward agent transforms itself into a backward agent and backtracks towards the source. The backward agent uses the exact reverse path of the forward agent and updates the probabilistic routing tables of intermediate nodes for destination d based on the information collected by the forward agent.

AntNet, also referred as backward routing, offers a robust adaptive routing approach in asymmetric packet switched networks. However, it is intrinsically slow since it requires the agent to reach the destination before any updates to the routing tables can be made. Co-operative Asymmetric Forward routing (CAF routing) combines the approach used by Schoonderwoerd et al. and AntNet to eliminate backward routing in asymmetric networks. Every node n stores the traffic characteristics (average packet delays) from its neighbors j. This statistical information is utilized by forward routing agents moving from n to j in order to update the probabilistic routing table in j. Thus eliminating the need for backward agents.

Other approaches that exploit agents for routing and network management schemes in circuit switched networks benefit from the concepts of Swarm Intelligence [50, 51]. Such an approach draws from the concept of multiple colonies of agents coexisting and in some cases coordinating with each other working towards independent goals. A similar approach was introduced by Minar et al., where a population of agents continuously traverse the network maintaining a brief history of its journey [26, 37, 38]. At every node visited, the agent updates the routing table of the node. The agents co-ordinate among themselves by sharing the history information with other agents on the same node, thereby attaining information about parts of the network without actually visiting it. They analyze the performance of their mechanism with different migration strategies of agents and with different agent populations.

Every agent-based strategy described above has it own individual goal. The routing algorithms based on the ant-approach offers an efficient adaptive routing algorithm that provides load balancing in the communication networks. The agent-routing approach utilizing a population of agents that co-ordinate among themselves provide a robust and fault-tolerant routing mechanism in networks. This thesis emphasizes on the use on intelligent mobile agents to provide a resource efficient and scalable routing approach in large communications network.

1.5 Thesis Overview

Many of the contemporary routing algorithms use simple mechanisms such as flooding or broadcasting to disseminate the routing information available to them. Such routing algorithms cause significant network resource overhead due to the large number of messages generated at each host/router throughout the route update process. Many of these messages are wasteful since they do not contribute to the route discovery process. Reducing the resource overhead may allow for several algorithms to be deployed in a wide range of networks (wireless and ad-hoc) which require a simple routing protocol due to limited availability of resources (memory and bandwidth). Motivated by the need to reduce the resource overhead associated with routing algorithms a new implementation of distance vector routing algorithm using an agent-based paradigm known as agent-based distance vector routing (ADVR) has been proposed. In ADVR, the ability of route discovery and message passing shifts from the nodes to individual agents that traverse the network, co-ordinate with each other and successively update the routing tables of the nodes they visit. The approach presented in this thesis is similar to that adopted by Minar et al. [26, 37, 38]. However, rather than carrying the entire history of their journey, the agents in ADVR efficiently distribute the routing information by employing an elegant communication and coordination mechanism.

Agents carrying minimal information with them, co-ordinate with other agents on the same node to make sure that they visit the entire network rather than clustering in specific parts of the network. In order to further reduce the resource overhead in ADVR, this thesis introduces a distributed control mechanism, whereby, the agent population can be dynamically controlled based on the resource overhead incurred in the network.

Chapter 2 explains the generic principles of an agent-based distance vector routing scheme giving a detailed analysis of its migration strategies. It also discusses the rationale behind maintaining an agent population and formulates techniques in achieving it. Chapter 3 gives a detailed overview of the experimental environment and presents the model of the event-driven simulator and various utility tools used for performance analysis in this thesis.Chapter 4 provides an experimental analysis of ADVR with respect to its performance in different aspects of network routing. The results obtained in ADVR is compared with the conventional version of distributed Bellman-Ford. Chapter 5 summarizes this thesis and gives the direction and scope for future research.

CHAPTER 2

AGENT-BASED DISTANCE VECTOR ROUTING (ADVR)

An extensive overview of the agent-technology and the use of intelligent mobile agents in the network routing domain was provided in the previous chapter. A detailed introduction to the agent-based routing mechanism implemented as a part of this thesis is outlined in this section. It explains the different aspects of the agent-based routing mechanism, such as the agent migration strategy and the dynamic control of agent population. An analysis of the experiments conducted with agent-based routing is presented in the next section.

2.1 Principles

As explained in Chapter 1, distance vector routing (DVR) algorithms has an advantage over link state routing algorithms with respect to simplicity, memory, and computation. Therefore, the work done in this thesis exclusively concentrates on DVR-class of algorithms. Distance vector is a simple, parallel, asynchronous, and a completely distributed routing algorithm implemented in different flavors in a large number of networks [30]. Every node n_i in the network updates its routing table using the Bellman-Ford equation [4, 30]:

$$D(i,j) = \begin{cases} 0 & \forall \quad i = j \\ \\ min[d(i,k) + D(k,j)] & \forall \quad n_k \text{ adjacent to } n_i \end{cases} \tag{2.1}$$

where $D(i,j)$ represents the distance 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(2.1) and integrates this value in its routing table. If the routing table of n_i is updated, the changes are broadcasted to all the neighbors, which in turn perform the same algorithm. Hence, a change in the routing table of one node can potentially cause a burst of broadcast messages throughout the network. This is an undesired phenomenon referred to as the broadcast storm problem. These broadcasted messages consume several important network resources such as network bandwidth, routing queues, and the processing power of the nodes. Further, the generation of these routing messages increases non-linearly with the size of the network, making them highly unscalable.

The agent-based solution proposed in this thesis replaces the routing messages in the network with an active population of agents. Rather than individual nodes exchanging routing information, mobile agents continuously traverse the network. At each node, they monitor the routing table and improve the existing routes based on the information gathered by them. After updating the routes, the agents select the updated routing table, choose a neighboring node, and migrate to it repeating the entire activity. This agent-based routing mechanism is called as Agent-based Distance Vector Routing (ADVR).

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. Hence, 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 Bellman-Ford equation [4]:

$$D(y, j) = \quad min(D(y, j), [d(y, x) + D(x, j)]) \quad \forall \; n_j \text{ in } R_x \tag{2.2}$$

where $D(x, j)$ is an entry in R_x representing the shortest distance to n_j from n_x. After performing the update at n_y, the agent selects R_y and migrates to an adjacent node using migration strategy γ. It is apparent that ADVR relies on the effectiveness of the distributed Bellman-Ford equation. However, the message overhead is considerably reduced and is bounded by the number of agents in the network.

2.2 Routing Table Selection Algorithm

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 to nodes with sub-optimal routes. The flexibility adopted by the agents in selecting the routing data reflects the inherent degree of intelligence acquired by it. In order to limit the routing data carried by the agents to a minimum, the agents undertake certain book keeping functions at every node, also known as the routing table selection algorithm.

Agents identify routing table entries that have been modified, yet have not been transferred to a particular neighbor. Every entry e_i in the routing table r_x of node n_x has a neighborhood vector v_i associated with it. The neighborhood vector is essentially a list of boolean flags corresponding to the neighbors of n_x. An entry $v_i[y]$ is set to 1 if e_i has been transferred to the corresponding neighbor y. At every node n_x, the agent performs the routing table selection algorithm as described in Figure 2.2 to choose a subset R_x from the routing table r_x. At startup, all the flags

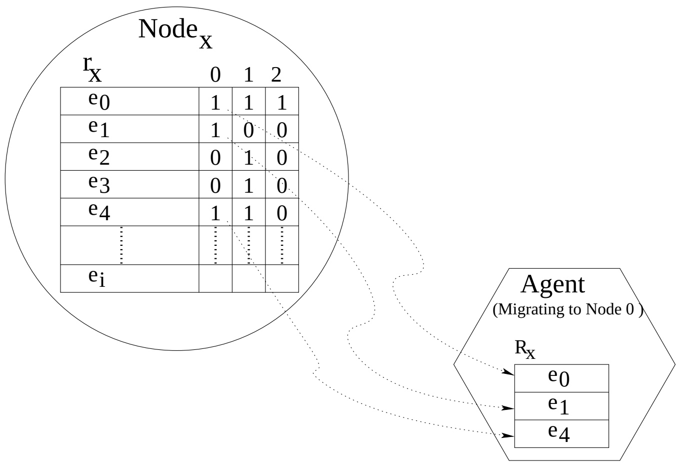


Figure 2.1: Selection of Routing Table Entries by the Agent

```
      INITIALIZATION OF NEIGHBORHOOD VECTOR
 1.   $\forall e_i$:
 2.       $v_i[y] := 1$, $\forall$ neighboring node $n_y$

      SELECTION OF ROUTING ENTRIES
 3.   $\forall e_i$:
 4.       for neighbor $n_y$, if $(v_i[y] == 1)$
 5.           Add $e_i$ to $R_x$
 6.           set $v_i[y] := 0$

      CHANGE IN ROUTING TABLE
 7.   $\forall e_i$ that is changed:
 8.       $\forall y, v_i[y] := 1$

      PERIODICALLY AFTER $\Delta t$
 9.   $\forall e_i$:
10.       $v_i[y] := 1$, $\forall$ neighbor $n_y$ whose entry has expired
```

Figure 2.2: Agent Routing Table Selection Algorithm

are set, i.e., $v_i := 1 \ \forall e_i$. 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_i in R_x for which $v_i[y] == 1$. The agent copies each entry e_i that is to be transferred to the neighboring node n_y to its data segment, and sets the corresponding boolean flag $v_i[y] == 0$. Further, any routing table entry e_i that is modified by an agent will have all its flags v_i reset to 1. To facilitate robustness and fault tolerance, all flags $v_i \ \forall e_i$ will expire after some time Δt and reset (i.e., $v_i := 1$). Resetting flags after Δt enables nodes to re-transmit routing updates that may have been lost during previous transmission attempts.

2.3 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. Even though each agent can be viewed as an individual, the movement of all agents must be coordinated in order to prevent agents from forming clusters in some parts of the network while neglecting to migrate to other parts. This coordination manifests itself in what is referred to as an agent migration strategy (γ).

An agent follows a 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 carefully chooses its migration strategy as there is no consensus on a single globally optimal strategy. A method suitable for one application can produce unwanted side effects for others. The simplest migration strategy is a random selection among all neighbors with uniform probability [1, 26, 37, 38]. 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 [37]. This scheme requires that agents maintain a migration history carrying records of their previous node visitations and refrain from migrating to recently visited nodes. 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 visitation history thereby making similar decisions resulting in clustering of agents in specific parts of the network while leaving other parts unvisited. Further, carrying the migration history as a part of agent payload increases the agent size imposing an overhead on the system resources.

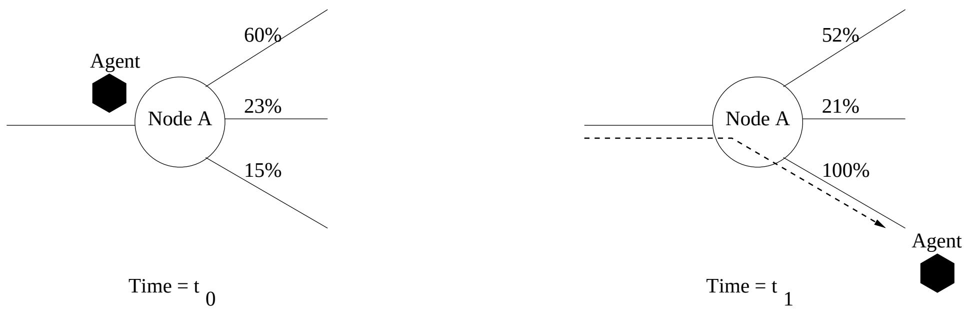


Figure 2.3: Migration Strategy using Edge Pheromones

A biologically inspired migration strategy derives from the concepts of stigmergy [11, 46]. Stigmergy is the mechanism used by naive insects to communicate with each other via changes in the local environment. Most of the migration strategies based on this scheme simulate foraging activities of ants. Ants in search of food leave a trail of pheromone, an exponentially decaying volatile chemical, to indicate their presence along a path. Other ants sense the strengths of these pheromone trails and follow the path of the strongest trail. Although this strategy has shown impressive results in certain adaptive routing applications, it tends to favor migration patterns, preventing uniform distribution of agents throughout the network. Hence, it may not be a feasible solution for a mechanism such as ADVR that require agents to explore the entire network with equal probabilities.

It is apparent that an ordered traversal of the network is advantageous for an agent population to uniformly distribute across the network. However the need to remember the migration itinerary must be eliminated. Further, the stigmergetic characteristics of insect colonies can be exploited to facilitate a communication mechanism between agents. Hence, the migration strategy employed in ADVR combines the strengths of exploratory feature of the depth-first-search and the biologically inspired communication mechanism. 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 by leaving pheromone trails. While the ant pheromones are used to attract other members of the community [46, 11, 50], in ADVR, 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 2.3 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 class of pheromones, that assist in agent migration strategy is referred to as edge pheromones.

It must be noted that agents migrating on the basis of edge pheromones will eventually form a migration pattern selecting the adjacent nodes in a fixed order.

Agents traversing the network using a fixed pattern may result in certain unwanted side-effects. Therefore, in order to break any such rigid patterns, the agents choose the next node at random with a small probability, thereby breaking the fixed order of node visitations. In other words, at every node, the agent may choose the migration strategy based on edge pheromones with a large probability (α) or randomly select the adjacent node with a small probability ($1 - \alpha$).

2.4 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. 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 convergence, 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)

# 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 2.1: Convergence Time and Routing Overhead for Different Agent Population

is in principle uncontrolled and depends on time and size of the network. However in ADVR it is limited by the number of agents that constitute the agent population.

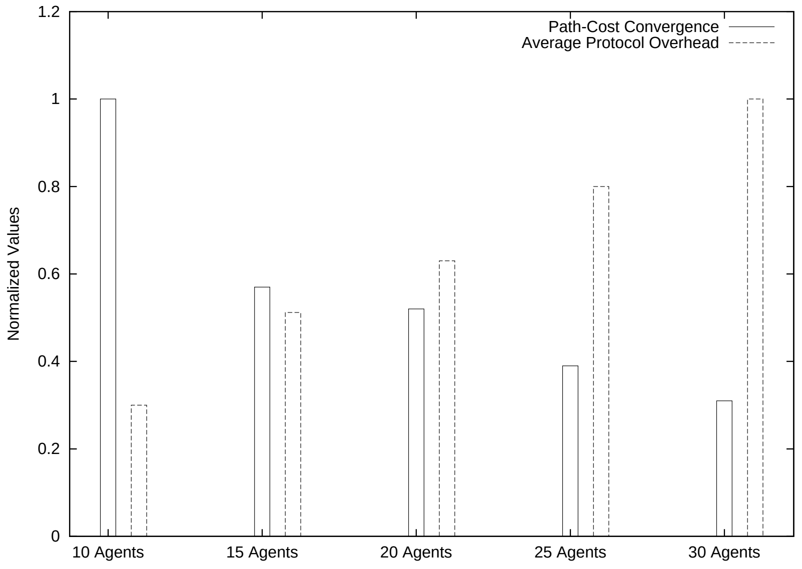


Figure 2.4: Comparison of Routing Overhead with Path-Cost Convergence

A large population of agents would increase the parallelism of ADVR resulting in an improved convergence [1]. However, it is extremely important to analyze the agent overhead in terms of consumption of bandwidth and computational cycles. Increasing the agent population will improve the path-cost convergence of the algorithm at the expense of increased resource demands. Table 2.1 displays the convergence time and average routing overhead for different agent population. Figure 2.4 shows 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 2.4 that the convergence time and routing overhead are inversely 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 population, 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, there is a need for a mechanism that is capable of adjusting the agent population as close to the optimal value as possible.

This will result in an acceptable path-cost convergence without producing excessive average overhead. It can be observed from Figure 2.4 that for such an optimal agent population the difference in the normalized convergence time and normalized average overhead must be minimum. In the given example 15 agents would result in an optimal trade-off between convergence time and resource overhead. It is easy to make such conclusions about the optimal population after the simulation results have been obtained. However, in the absence of such post-simulation analysis, it is difficult to obtain a balance between route convergence and resource overhead. Further, the unpredictable behavior of dynamic networks continuously vary the importance of the two conflicting requirements. Thus, it is necessary that an adaptive multi-agent system provides an efficient control mechanism that allows a dynamic control of the agent population to balance the convergence time against resource overhead in the network.

Changing the agent population dynamically in response to its environment is a complex issue in the absence of a central controller. Individual agents lacking a bird's eye view of the network are unable to make global assessments regarding the resource availability and the characteristics of the network. Therefore, it requires a high degree of coordination among agents to analyze the global environment from local information. To facilitate such a coordination, ADVR once again exploits the

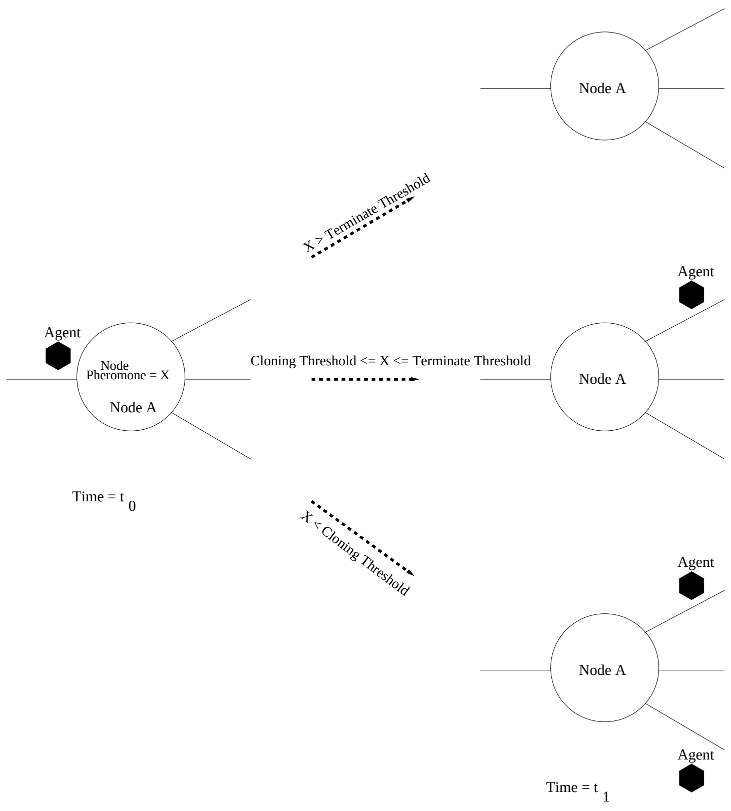


Figure 2.5: Population Control using Node Pheromones

principles of stigmergy. 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.

The strength or intensity of node pheromones can be expressed by the equation $e^{-(\Delta t)}$, where Δt is the time since the deposition of the pheromone. An agent visiting a node n_x at time t_2 extracts the value of the node pheromone that was deposited at time t_1 ($t_1 < t_2$) using the equation $e^{-(t_2-t_1)}$. If this value is above a certain termination threshold (Ψ), the agent kills itself. On the other hand, if the node pheromone value reduces below a cloning threshold (Ω), the agent clones itself. However if $\Omega \leq e^{-(\Delta t)} \leq \Psi$, the agent neither clones nor kills itself. Before leaving n_x, the agent deposits additional node pheromone at time t_2. The values of Ψ and Ω can be controlled dynamically to indicate the availability of local resources at the nodes. Hence, when an agent realizes that the resource availability at a node is low, it dynamically manipulates the values of Ψ and Ω, such that successive agents visiting the same node have a higher probability of killing themselves. Likewise, when an agent realizes that the local agent visitation on a node is low, signifying slow route convergence, it manipulates Ψ and Ω to increase the local agent population. A flexible mechanism can combine other factors to control the population of agents including the path-cost convergence time.

ADVR implementing a dynamic agent population may start with a single agent or an arbitrary number of agents. Nevertheless, the agents coordinate among themselves and dynamically adjust to a particular range of population based on the values of Ψ and Ω. Values of Ψ and Ω may vary asynchronously at individual nodes causing constant fluctuations in the agent population in the network. However, an appropriate mapping of Ψ and Ω to the resource availability must reflect the global state of the network based on local information, thereby reducing extreme fluctuations in agent population at individual nodes. Mapping these thresholds in a distributed fashion to the global state of the network based on local information is beyond the scope of this thesis.

It is important that the agents are highly responsive to changes in the network conditions. For example, any failure in the network must be reflected in the routing tables of individual nodes as soon as possible to avoid any long-term or short-term routing loops. To accomplish this task, a node that detects a failure in the network generates special type of agents called as auxiliary agents. Auxiliary agent propagate the negative information regarding the network failure within its neighborhood. They are special because they are not subject to the rules governing the agent population using node pheromones. In other words, when a node detects a failure in the network it generates a suitable number of auxiliary agents to traverse each of the adjacent edge and carries the new information to its immediate neighbors. Making these agents immune to the population control mechanism guarantees the propagation of important routing information without the agent terminating itself. On the other hand, allowing an unrestricted lifespan to an agent can monotonically increase the agent population over a period of time. Therefore, auxiliary agents are characterized by a hop count, which signifies the number of migrations they can make before being subject to the population control mechanism. This ensures that when a network failure occurs, the agent population is increased, sustained at this increased level for a sufficient time for the new information to propagate, and finally reduced to optimize the network resources.

CHAPTER 3

SIMULATION ENVIRONMENT

A detailed discussion including the basic principles and migration strategy in agent-based distance vector routing (ADVR) was presented in the previous section. An extensive description of the simulation and experimental environment is provided in this chapter. An analysis of different performance issues of ADVR in comparison with conventional distance vector routing is depicted in the next chapter.

3.1 Simulation Model

In order to analyze and compare the performance of ADVR with conventional DVR, an object-oriented discrete event-driven simulator was implemented. This simulator will be referred to as RSim (Routing SIMulator) throughout this thesis. RSim models the physical entities in the target system as objects. Interaction between these entities is mapped as an event (e_i, t_i), where e_i is some action to be performed at time t_i.

As shown in Figure 3.1, to the heart of RSim is the simulation engine that maintains the global time of the system and manipulates a set of events (e_i, t_i). The

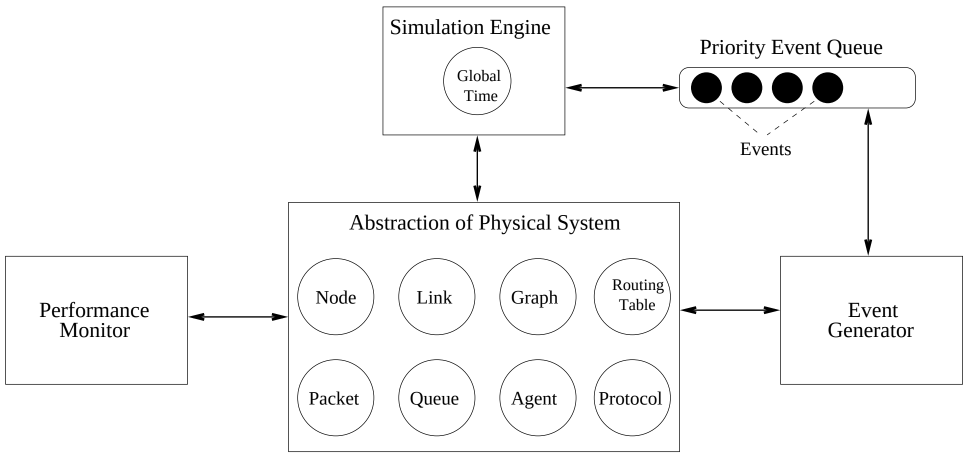


Figure 3.1: RSim (Routing SIMulator)

simulation engine maintains the set of events as a heap (priority queue) in an ascending order of time. The engine selects the event (e_i, t_i) such that $t_i < t_x, \forall(e_x, t_x)$, executes the routines associated with it, and advances the global system time to t_i. The execution routines associated with the event e_i abstract the characteristics of the physical event by calling routines on the objects that model the physical entities. Physical entities such as network node, communication link, routing queue, mobile agent, data packet, routing table, and network graph are represented by corresponding software components, each of them capable of generating events to communicate with other entities. Further, RSim uses a sophisticated performance monitor to record different statistical data generated during the course of simulation.

3.2 Network Model

A physical network is represented in RSim as an undirected connected graph $G(V, E)$ with $|V|$ vertices and $|E|$ edges. An edge xy in the graph connects nodes n_x and n_y in both directions. Asymmetric weights are associated with edges in both directions. The weight of the edge can change to any positive value during the course of the simulation to reflect network characteristics. RSim assumes a reliable network, thereby avoiding the complexities of network layer protocols that handle data loss and corruption.

Further, every node in the graph represents a store-and-forward router, which is characterized by a limited buffer space and processing speed. A link connecting two nodes is characterized with a specific link capacity. Following the example of a popular implementation of DVR, namely the Routing Information Protocol (RIP) [22], RSim assumes 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 3.2 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

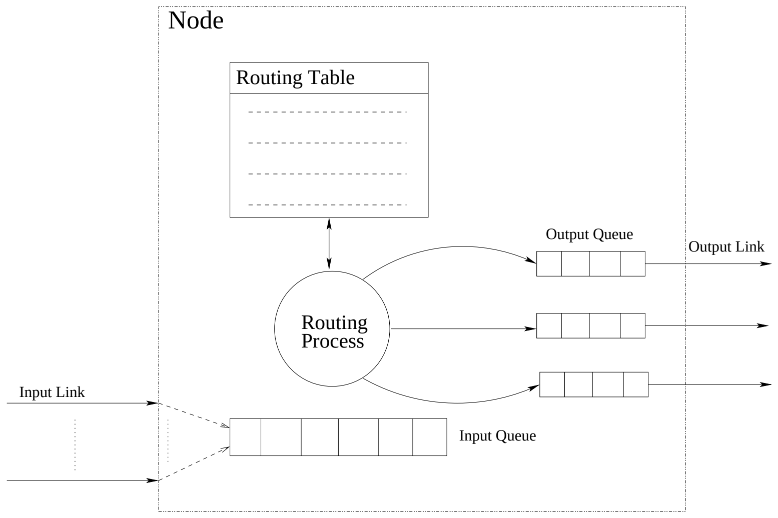


Figure 3.2: Simulation Model for DVR

queue depends on the processing rate of the router which can be in the range of $300000 - 500000$ packets per second (pps) [10, 43]. 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 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 experiments conducted with RSim, a link capacity of 10 Mbps is assumed, which gives 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

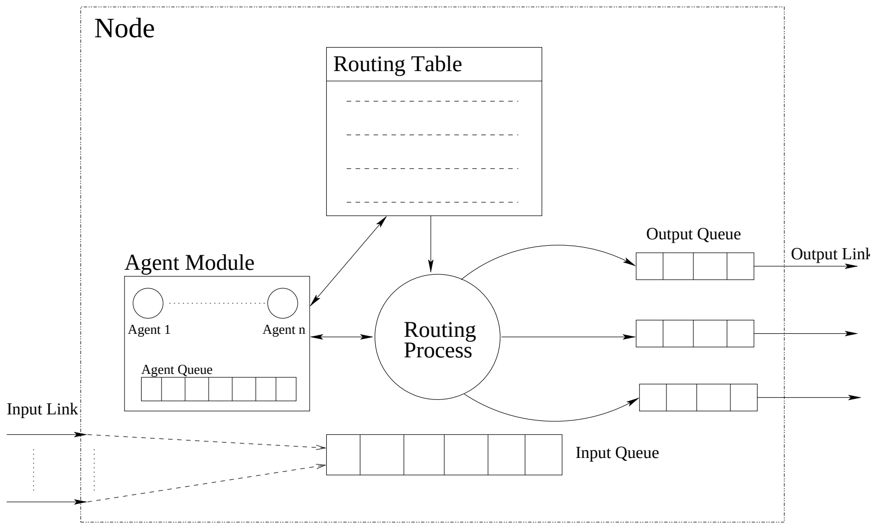


Figure 3.3: Simulation Model for ADVR

service rate.

Figure 3.3 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

```
    Initialization
1.  set $A \leftarrow V$
2.  set $B \leftarrow \{\}$
3.  set $C \leftarrow \{\}$
4.  set $D \leftarrow \{\}$
    Spanning Tree
5.  $v \leftarrow$ randomly chosen vertex from set $A$
6.  $B \leftarrow \{v\}$
7.  $A \leftarrow A - \{v\}$
8.  for each $v \in A$ chosen randomly, do
9.          $u \leftarrow$ randomly chosen vertex from $B$
10.         $C \leftarrow C + \{\{u,v\},\{v,u\}\}$
11.         $A \leftarrow A - \{v\}$
12.         $B \leftarrow B + \{v\}$
    Satisfy graph degree
13. for each edge $\{u,v\} \in V \times V$, $\{u,v\} \notin C$, do
14.         if $u \neq v$ then
15.         $D \leftarrow D + \{\{u,v\}\}$
16. while $|C| < e$ do
17.         $\{u,v\} \leftarrow$ random edge in $D$
18.     $C \leftarrow C + \{\{u,v\},\{v,u\}\}$
19.     $D \leftarrow D - \{\{u,v\},\{v,u\}\}$
```

Figure 3.4: Pseudo-random Graph Generation Algorithm

its migration strategy. The routing process is responsible for routing incoming regular data packets to the appropriate interfaces using the routing table maintained by the agents.

3.3 Tools

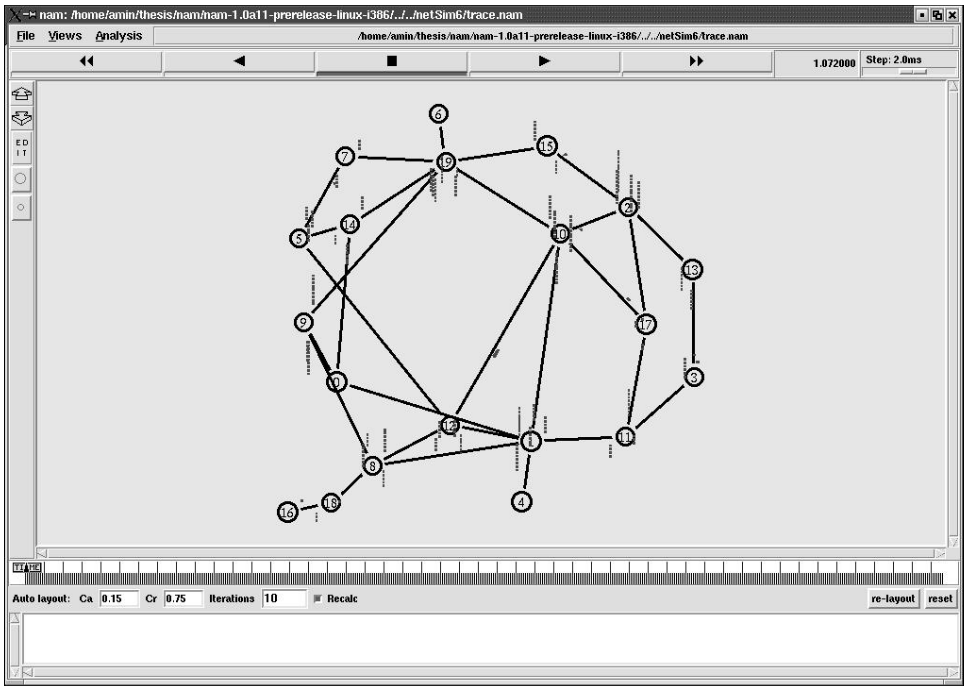
Several utility tools were developed along with RSim to accomplish certain specialized task. These tools were extensively used for the experimental analysis conducted as a part of this thesis.

3.3.1 Graph Generator

The graph generator was developed to construct a pseudo-random, connected, undirected graph $G(V, E)$ with $|V|$ nodes and $e = |E|$ edges. As shown in Figure 3.4, the graph is generated in a two step process. First, the graph generator builds a random spanning tree containing $|V| - 1$ edges in set C, hence ensuring that the graph is connected. Secondly, it adds $e - (|V| - 1)$ random edges from $S - C$, where $S = \{u \times v | u \neq v; u, v \in V\}$, to make e edges in total. Features to control the average node degree of G, $\delta(G)$, have been implemented, whereby $e = \frac{\delta(G) \times |V|}{2}$. It must be noted that this algorithm biases the graph with a few nodes with a higher degree.

3.3.2 Network Animator

The Network Animator (NAM) [14] is a Tcl/Tk based utility tool used to visualize network simulation traces. It has the capability to read large animation data recorded by simulators and animate the real world packet trace data. It is highly customizable



nam: /home/amin/thesis/nam/nam-1.0a11-prerelease-linux-i386/../../netSim6/trace.nam

/home/amin/thesis/nam/nam-1.0a11-prerelease-linux-i386/../../netSim6/trace.nam

1.072000 Step: 2.0ms

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

Figure 3.5: Network Animator

and is generally used with the Network Simulator (NS) [31]. The visualization module of RSim acts as an adapter to convert the events generated by RSim into a format understandable by NAM. It produces a trace-file with information regarding the network topology, nodes, links, queues, packet transmission, and network connectivity (See Figure 3.5). Such a visualization mechanism served as a useful tool in evaluating the influence of several parameters in RSim.

CHAPTER 4

EXPERIMENTAL ANALYSIS

The simulation environment for the experimental analysis of this thesis is discussed in the previous chapter. The analysis and performance evaluation of DVR and ADVR is conducted in this chapter. The next chapter will summarize this thesis describing the scope of future research work on this topic.

4.1 Definitions

Before discussing the experimental results, it is essential that we establish a vocabulary that will be useful in our further discussions.

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

Definition 2 Peak Instantaneous Routing Load *(Peak IRL) of a routing algorithm is the maximum IRL throughout the period of its execution.*

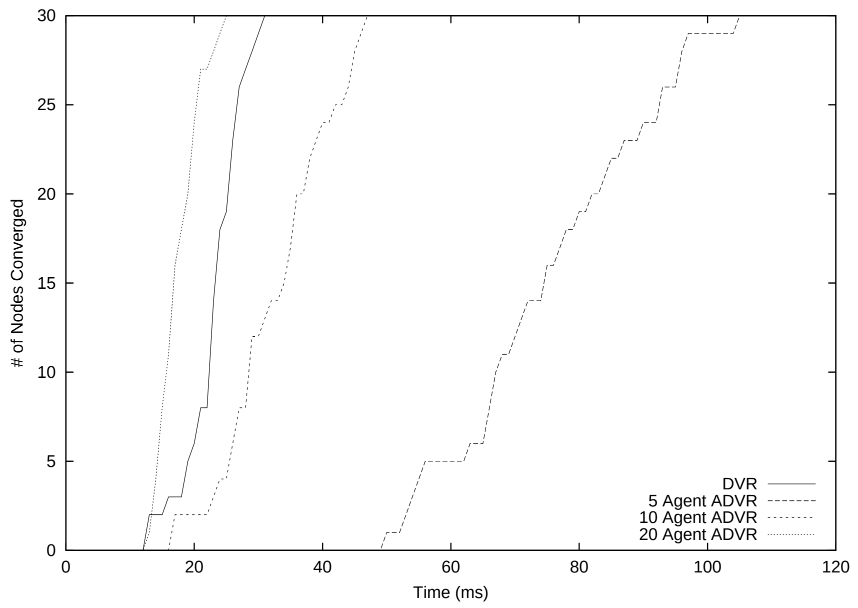
Definition 3 Network::(o, d) *is defined as a randomly connected network with order (number of nodes) o and average degree d.*

Definition 4 Path-Cost convergence *of the algorithm is defined as the process where every node in the network has a shortest-path route to every other node in the network.*

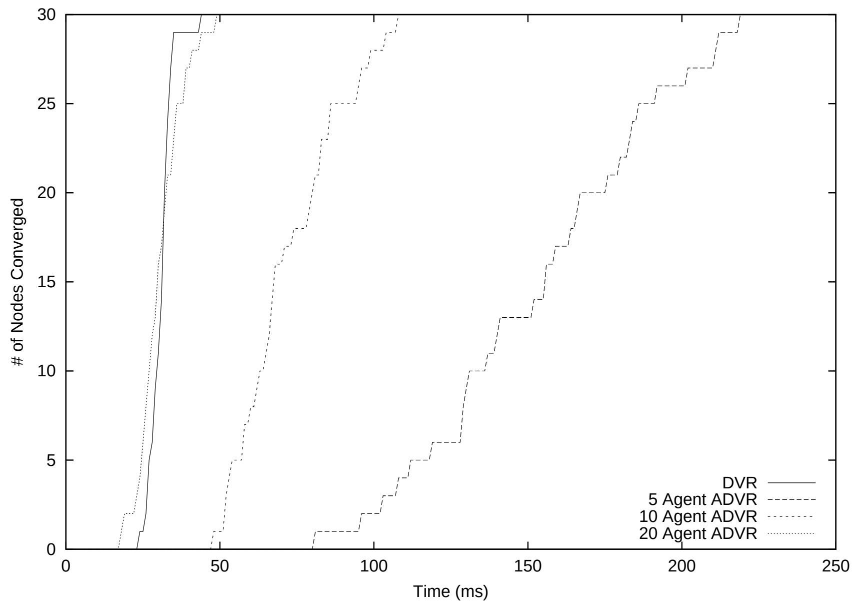
Definition 5 Route discovery *is the process where every node in the network obtains a route for every other node in the network.*

4.1.1 Analysis of Path-Cost Convergence

In order to verify the utility of ADVR, it is imperative to show that ADVR will converge with any number of agents. Further, it is vital to compare the convergence performance with conventional DVR in order to compare the two approaches in various scenarios. 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 may have a cascading effect initiating a sequence of broadcasts throughout the network. Such an aggressive parallelism in DVR may result in bursts of update messages within the network. Conversely, ADVR implements controlled parallelism characterized by the number of agents in the network. Although ADVR



(a) Network::$(30, 3)$



(b) Network::$(30, 6)$

Figure 4.1: Pat-Cost Convergence

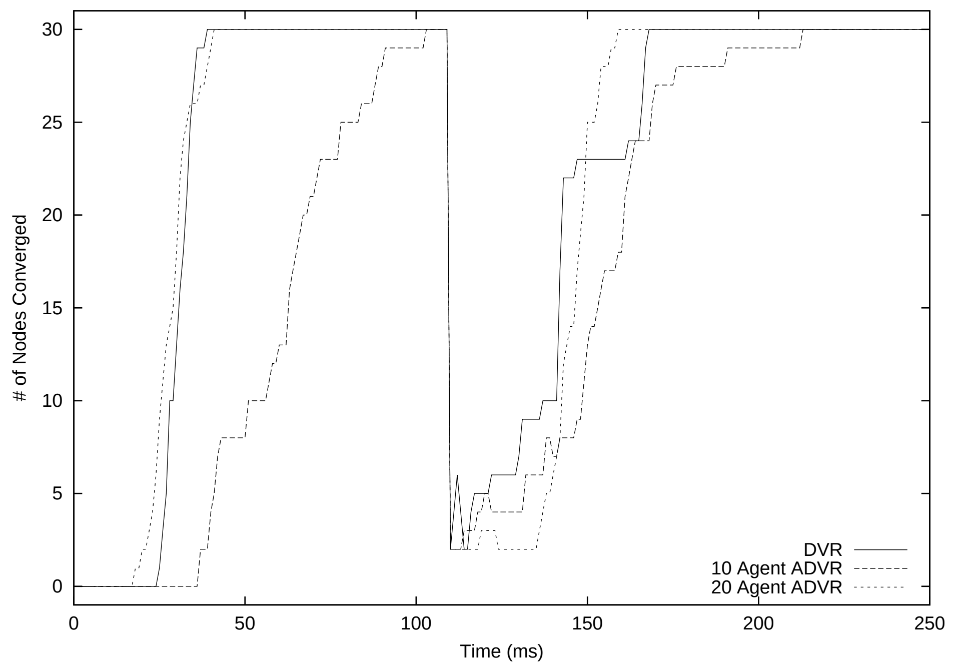


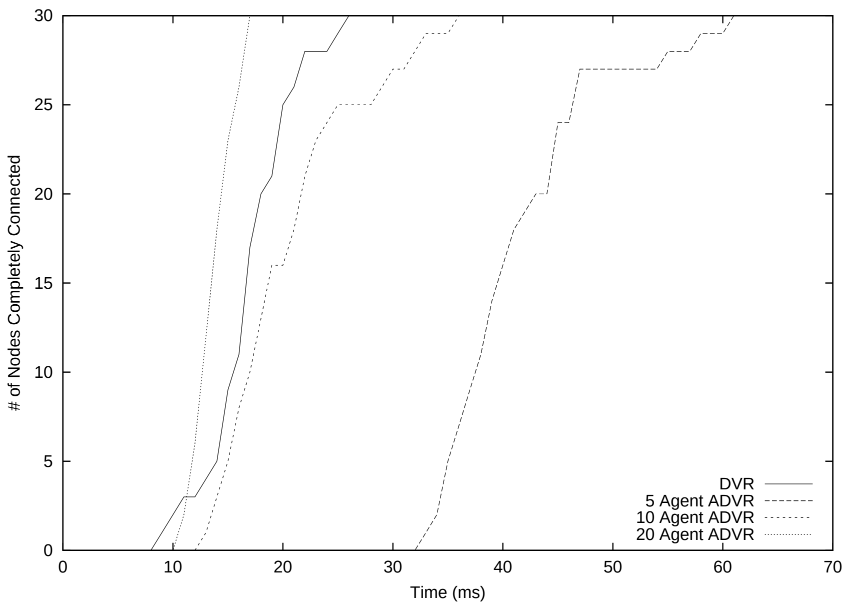
Figure 4.2: Response to Network Failure

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.

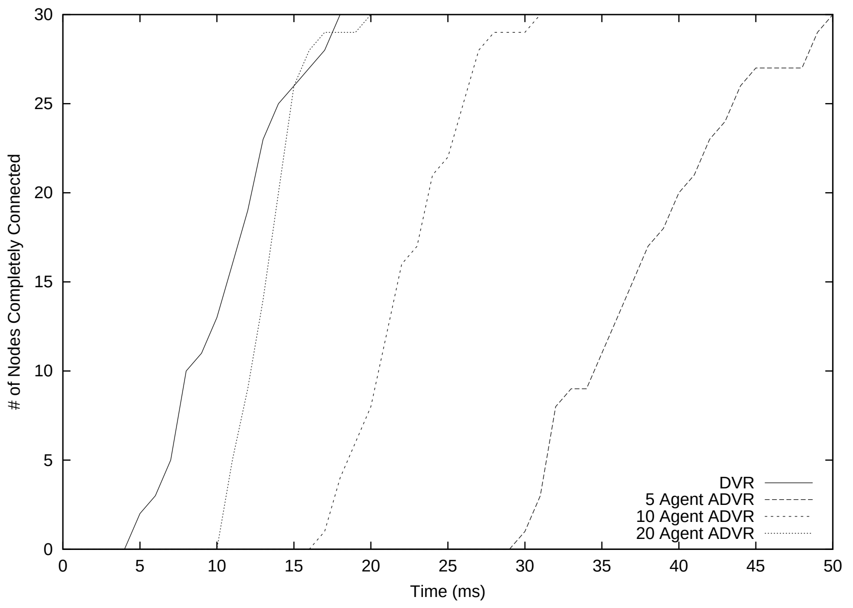
It can be shown, that in a static network, a single agent can achieve the convergence of routing tables at all nodes in the network, provided that it uses an appropriate migration strategy, which allows for complete traversal of the network. Nevertheless, a single agent is insufficient to complete this task in a time that is comparable to that of concurrent messaging in DVR. Hence, in order to achieve a comparable performance with DVR, a population of agents will have to be deployed in the network. The number of agents in the network mark the degree of parallelism in ADVR. These agents implicitly cooperate and coordinate with each other, thereby accelerating the process of path-cost convergence.

It is shown in Figure 4.1(a) that a small agent population ($\ll o$), where o is the order of the network, exhibits a slow yet comparable convergence with respect to DVR. It is evident that the small agent population cannot successfully compete with the aggressive parallelism of DVR. However, by increasing the agent population to some value ($< o$) the convergence of ADVR can outperform that of DVR. It must be noted that although a small population of agents have a degree of parallelism less than DVR, they are capable of outperforming DVR since they do not experience large queuing delays due to aggressive messaging. It can be observed in Figure 4.1(b) that the path-cost convergence in ADVR is further reduced in denser networks. The increased density causes the agents to traverse more edges, therefore taking a longer time to converge. Hence, a larger agent population is required in a dense network to achieve the same performance as that in a sparse network.

Throughout the course of its existence, the topology of the network can change



(a) Network::$(30, 3)$



(b) Network::$(30, 6)$

Figure 4.3: Route Discovery

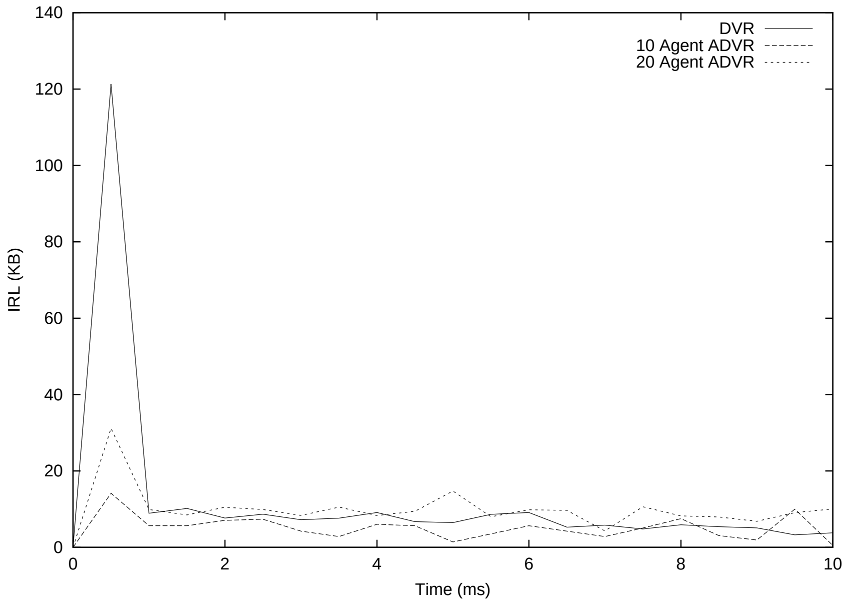
due to several reasons such as node and link failures. It is essential that a routing algorithm detect these failures and converge to the new shortest-paths without causing large unwanted delays in data delivery. Hence, it is important to analyze the responsiveness of the routing algorithm in the event of topological changes. Figure 4.2 compares the convergence time of the routing algorithms in Network::$(30, 6)$ after a major topological change. It is evident that ADVR with a reasonable population of agents using the concept of auxiliary agents can converge from topological changes in times comparable to DVR. Once again it is apparent that ADVR with a larger agent population produces convergence results better than DVR.

4.1.2 Analysis of Route Discovery

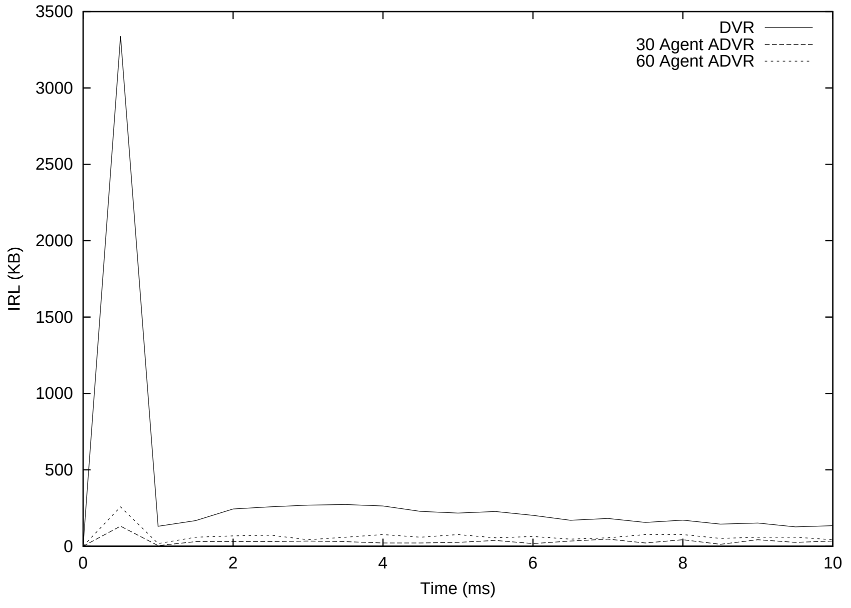
Route discovery plays an important role in the performance of communication networks. It is crucial to evaluate any routing algorithm with respect to the speed at which nodes in the network obtain a route for every other node in the network. Even if these routes are sub-optimal, they provide a benchmark to measure the availability of the network to be used by other applications. Figure 4.3(a) 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, manifesting small concurrency are insufficient to discover routes as rapidly as DVR. Like path-cost convergence, 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 lower as 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. Again, it is evident from Figure 4.3(b) that performance of ADVR is hampered by increased network density due to the increased number of edges that must be traversed by the agents.

4.1.3 Analysis of Routing Overhead

As mentioned in Chapter 2, 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 overhead, ADVR can provide a highly scalable solution to the routing problem. In order to evaluate the scalability of any routing algorithm, it is essential to analyze the instantaneous routing load (IRL) incurred in the algorithm.



(a) Network::$(30, 3)$



(b) Network::$(100, 6)$

Figure 4.4: Instantaneous Routing Overhead

For an algorithm to be scalable, the IRL in the network should be as low as possible at all times and without large variations. Figure 4.4(a) shows the average IRL for DVR and ADVR measured every 0.5 milliseconds. It was observed that in a stable state of the network, the IRL in both algorithms is fluctuating around a constant value. However, it is interesting to see the Peak IRL of the algorithms in the event of an unstable network. Owing to its aggressive parallelism, DVR experiences a large flow of messages at network startup. Although such parallelism helps DVR with a slightly faster convergence, the peak routing load in DVR is several magnitudes larger than that of ADVR. Even with a larger population of agents that converges faster than DVR, the peak routing load in significantly smaller than DVR. It is evident from Figure 4.4(b) that an increase in network size and density produces an excessive increase in the peak IRL for DVR. Such a sharp increase in routing traffic implies its lack of scalability and can cause an overflow of transmission queues, thereby contributing to jitter, packet loss, and/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 maintaining its low and stable peak IRL, proportional to the number of agents in the network.

Figure 4.5 shows the average delays experienced by data traffic in an unstable network. Application data exchanged between network nodes experience variable

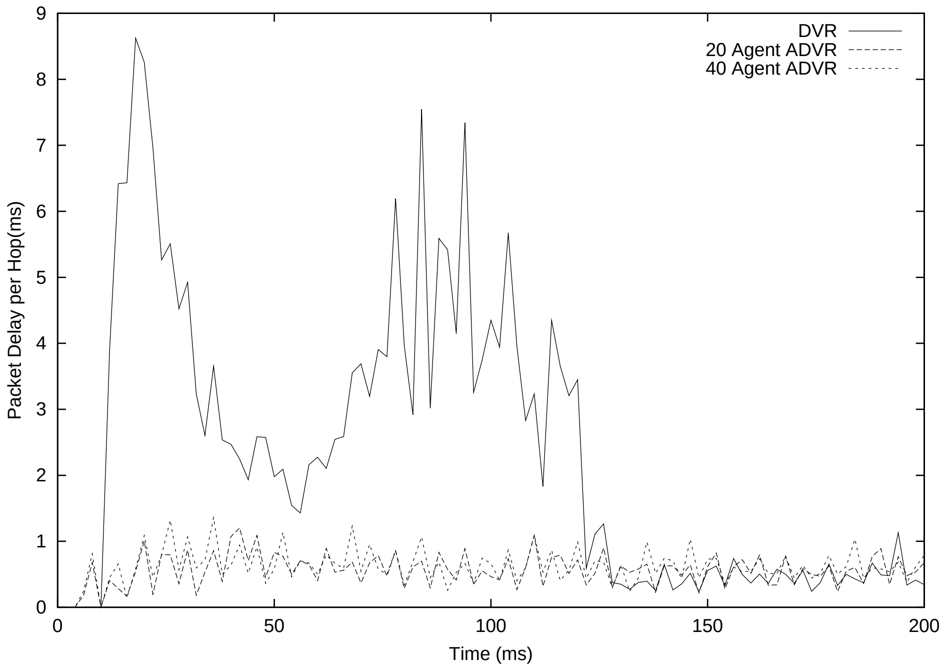


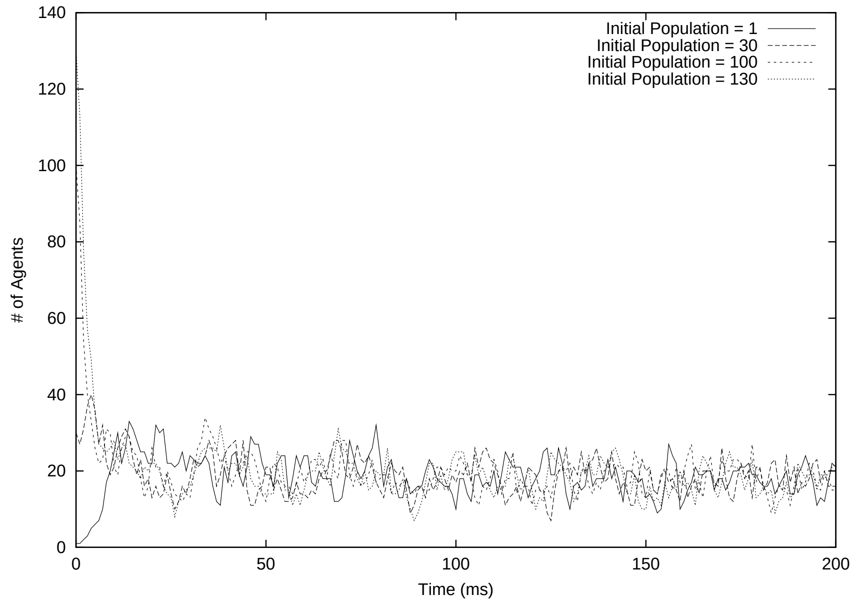
Figure 4.5: Average per Hop Packet Delay

queuing and transmission delays depending on the stability of the network. It can be seen that with DVR, any change in the network topology severely hampers the performance of the network due to increased IRL. This ultimately increases the queue lengths at individual nodes causing data packets from other applications to experience excessively large queuing delays. This problem is not encountered in ADVR since the IRL never increases beyond a certain small value which is indirectly controlled by adjusting the agent population. It must be noted that increasing the agent population to a significantly large number does not necessarily increase the average packet delays. This enables the agent-based mechanism to be used in networks that are required to satisfy strict Quality of Service (QoS) assurances even in the event of major topological changes.

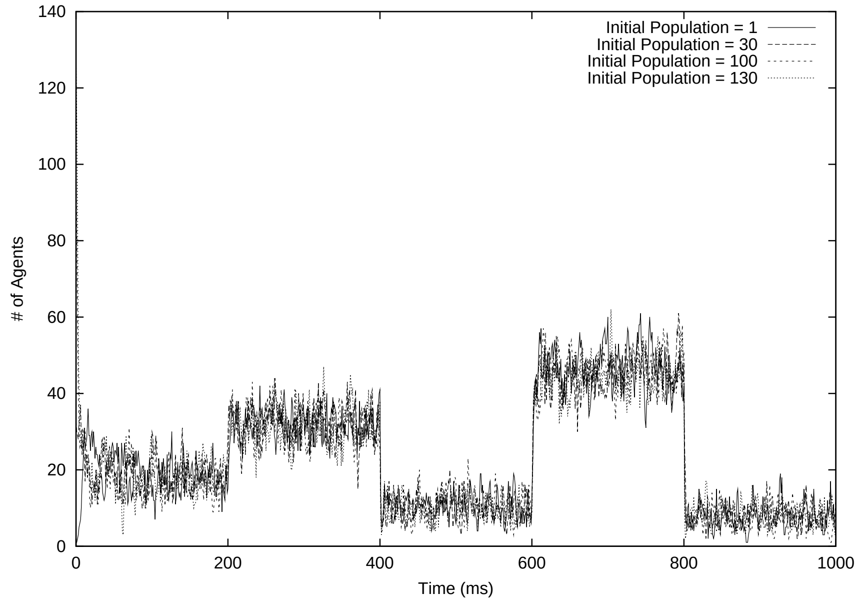
The RSim simulation model assumes that the agent code segment consumes 100 bytes of the IP packet. To reduce the resource overhead, it is imperative to consider the structure of the agents. If the agent code segment is excessively large, the agent will consume significant amounts of resources in terms of bandwidth, memory, and computing cycles. 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 can be 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 of resource consumption, it simply shifts the resource overhead from the link (i.e., bandwidth) to the node (i.e., computation).

4.1.4 Analysis of Agent Population in ADVR

Experiments conducted for this thesis assume that a mechanism to map the values of Ψ (Termination Threshold) and Ω (Cloning Threshold) to the global state of the network based on local information exists and is used by the agents in ADVR. Hence, the values of Ψ and Ω are consistent at all the individual nodes in the network. Figure 4.6(a) shows the convergence of the agent population in Network::$(30, 3)$. The experiment was conducted with constant values for the thresholds ($\Psi = 0.75$ and $\Omega = 0.25$). The purpose of the experiments was to demonstrate the ability of the population control mechanism in ADVR to control the agent population in the network. It was observed that irrespective of the initial population, the agent population converges to a small range. Networks initialized with a small number of agents escalate the agent population to a certain value. However, the escalation of agent



(a) Variation in Agent Population with $\Psi = 0.75$ and $\Omega = 0.25$



(b) Variation in Agent Population with different values of Ψ and Ω

Figure 4.6: Dynamic Control of Agent Population

population occurs at every node flooding the network with excessive agents. Nevertheless, the system responds to the overhead manifested by an excessive number of agents and autonomously reduces the population as a function of the corresponding pheromone thresholds. On the other hand, networks initialized with a large number of agents realize the per-agent overhead and continuously reduce the number until it reaches a population that has a small deviation about a mean value.

Although Figure 4.6(a) shows the convergence of agent population in the network based on some fixed values of Ψ and Ω, the resource utilization in a real network will change dynamically thereby requiring the values of Ψ and Ω to be adapted. This results in a continuous adjustment of the constituent agent population. Figure 4.6(b) shows the control of agent population in the network with varying values of Ψ and Ω. A vector, $< time(ms), \Psi, \Omega >$,was applied to the agent system, where time denotes the simulation time in milliseconds when the new values of Ψ and Ω would be reflected in the network. The values for the vectors that have been supplied in this experiment are as follows: $< 0, 75, 25 >, < 200, 90, 40 >, < 400, 60, 10 >, < 600, 95, 45 >$ $, < 800, 55, 5 >$. It can be seen that the control mechanism is highly responsive to the variation in threshold values and dynamically adjusts the agent population appropriately. It should be noted that such a control mechanism is extremely useful, not only to control the agent population based on the resource overhead, but also to control the population based on other requirements. For example, an agent-based system that desires low convergence time, irrespective of the incurred overhead, can dynamically scale the agent population by appropriately adjusting the values of Ψ and Ω. There are a variety of different mechanisms that are deemed appropriate for the adaptation of threshold values as a function system behavior. In ADVR, for example, each node may observe the number of changes to the local routing table that occur with each visit of an agent. Here, a large number of updates may indicate highly dynamic network behavior and may warrant a setting of threshold that result in an increased agent population. In contrast, few or no routing table updates for some period of time is indicative of a network whose conditions have stabilized. In the interest of preserving resources, the constituent agent population may now be reduced by adapting Ψ and Ω correspondingly. While this mechanism provides a possible solution to the problem of mapping system behavior to threshold values, it does not help to determine the appropriate population levels. Hence the adjustments of Ψ and Ω must be supported by a simple learning algorithm that can aid in suppressing oscillations of agent population.

CHAPTER 5

SUMMARY AND FUTURE WORK

An analysis of the experimental evaluations of ADVR was presented in the previous chapter. A summary of the entire thesis and the scope of future work within this research topic is depicted in this chapter. The impact of the findings in this research on several other distributed application is also discussed here.

5.1 Summary

This thesis 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 routing messages are replaced by a population of agents. The corresponding message activity is thus bounded by the number of constituent agents in the network. 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.

According to ADVR, a group of agents traverse the network visiting network nodes. At each individual node, the agent executes the distributed Bellman-Ford equation, therefore collectively converging all the nodes in the network with the shortest-path routes to every other node. The agents execute a special routing table selection algorithm by which they only propagate selected information from one node to another. Multiple agents communicate with each other using synthetic pheromones. They co-ordinate their activity in order to equally distribute themselves throughout the network without forming agent clusters. It was shown that larger the agent population, the better was the path-cost convergence. Nevertheless, the increased agent population also increases the average routing overhead. This suggests that an optimal population of agents must be deployed in the network to satisfy the performance and resource overhead requirements of the network. ADVR employs a pheromonal approach which autonomously controls the agent population. The agents can dynamically terminate or clone themselves as a function of the pheromone value detected at the node, the Cloning Threshold (Ω), and the Termination Threshold (Ψ). In order to enhance the responsiveness of ADVR in the event of a topological change, an additional concept of auxiliary agent was introduced. The auxiliary agents with a limited hop count help to propagate the new topological information within the neighborhood of the failed node or link.

Several experiments have been conducted to analyze the performance of the agent-based distance vector routing scheme. In particular, importance was given to the path-cost convergence, route discovery, and instantaneous routing load (IRL) of ADVR. Experiments were also conducted to verify the distributed and dynamic manipulation of the agent population in the candidate network. It was verified that although DVR is aggressively reactive in path-cost convergence and route discovery, ADVR with a small agent population produces a comparable result. Further, ADVR with a substantial number of agents can closely compete with the performance of DVR, at times outperforming it. It was analyzed that the use of auxiliary agents by ADVR in the wake of network failures enhances the responsiveness of ADVR to rapidly converge to the new routes. With the help of carefully crafted experiments, it was shown that the message activity in DVR was significantly higher when compared to that of ADVR. Every change in the network topology produces an excessively large number of routing messages in DVR due to its broadcast storm problem. This large message activity further increased with an increased network density. It was observed that such large routing loads in the network using DVR causes excessive queuing delays for data packets from other applications in the network. This causes unwanted jitter and data loss thereby severely hampering the performance of the network. Experiments with the dynamic population control mechanism show that the agent population can be effectively controlled by adjusting the value of Ω and Ψ to reflect the appropriate state of the network. Irrespective of the initial agent population, the distributed control mechanism results in a small range of agents deployed in the network. The population of these agents can be changed dynamically at run-time by appropriately changing the values of Ω and Ψ.

5.2 Future Work

The results gathered in this thesis are extremely encouraging. However, it is important to validate the experimental results with sufficient analytic proofs. Carefully designed theoretical derivations must prove the validity of several concepts introduced in this thesis. It is essential to prove that single and multiple agents employing the pheromone-enabled migration strategy will visit all the nodes and edges in the network and converge the network to its shortest-path routes.

In order to provide a flexible and dynamic population control mechanism, it is necessary to represent the global state of the network into a set of independent threshold values (Ω and Ψ). Further, these threshold values must be appropriately deduced form the local environment of individual nodes and must be flexible enough to respond to rapid network changes. This is a complex research project and will draw results from several aspects of computer science including artificial intelligence, decision theory, and graph theory.

The robustness of the agent-based routing mechanism can be furthered improved by designing a work sharing logic between the agents. A group of agents will communicate with each other to logically partition the physical network into equal sized sectors. The agents will then equally distribute themselves among those partitioned sectors each group taking care of its sector. All the concepts introduced in this thesis will then be applied within individual sectors. In other words, such an approach provides a hierarchical framework to the agent-based mechanism increasing its fault tolerance and scalability.

It must be noted that the feasibility and performance analysis of ADVR is a significant yet incomplete part of a larger research topic. In order to fully justify the application of ADVR to contemporary networks it is essential to have a working implementation of the new routing algorithm. The deployment of such an application onto physical machines will help to understand several requirements and limitations of the agent-based approach in the real world.

5.3 Broader Impact

The results of this thesis are expected to provide alternative ways to design and implement resource efficient and scalable 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 thesis is on routing, agent-based solutions are deemed suitable for many other distributed 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, it is expected that this thesis and its corresponding results will motivate the design of agent-based solutions for large scale system-level applications.

BIBLIOGRAPHY

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

[2] K. Amin and A. Mikler, *Dynamic Agent Population in Agent-based Distance Vector Routing.* ISDA2002: Second International Workshop on Intelligent Systems Design and Applications, Atlanta, USA, August 2002.

[3] R. Beckers, J. Deneuborg, and S. Goss, *Trails and U-turns in the Selection of a Path of the Ant Lasius Niger.* In J. theor. Biol. Vol., 1992.

[4] R. Bellman, *Dynamic Programming.* Princeton University Press, 1957.

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

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

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

[8] P. Braun, C. Erfurth, and W. R. Rossak, *An Introduction to the Tracy Mobile Agent System.* Technical Report No. 2000, Friedrich-Schiller-Universitat Jena, Institut fur Informatik, September 1999.

[9] 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.

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

[11] G. Di Caro and M. Dorigo, *AntNet: A Mobile Agents Approach to Adaptive Routing.* Tech. Report, IRIDIA/97-12, Universit Libre de Bruxelles, Belgium.

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

[13] 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.

[14] D. Estrin, M. Handley, J. Heidemann, S. McCanne, Y. Xu, and H. Yu, *Network Visualization with the VINT Network Animator NAM.* Technical Report 99-703b, University of Southern California, March 1999.

[15] R. A. Flores-Mendez, *Towards a Standardization of Multi-Agent System Frameworks*. ACM Crossroads, Summer 1999.

[16] I. Foster and C. Kesselman, *Globus: A Metacomputing Infrastructure Toolkit*. The International Journal of Supercomputer Applications and High Performance Computing, 11(2):115-128, Summer 1997.

[17] D. Freedman, *Experience Building a Process Migration System for Unix*. Proceedings of the USENIX Winter Conference, pages 349–355, January 1991.

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

[19] J.J. Garcia-Luna-Aceves, *Loop-Free Routing Using Diffusing Computations*. IEEE/ACM Trans. Networking, 1:130–141, February 1993.

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

[21] R. Gray, D. Kotz, G. Cybenko, and D. Rus, *Agent Tcl*. In: W. Cockayne and M. Zyda (eds) Mobile Agents: Explanations and Examples, Manning Publishing, 1997.

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

[23] IBM Software, *The IBM Aglets Workbench.* Website: http://www.trl.ibm.co.jp/aglets.

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

[25] D. Johansen, R. van Renesse, and F.B. Schneider, *An Introduction to the TACOMA Distributed System—Version 1.0.* Technical Report 95-23, Dept. of Computer Science, Univ. of Troms and Cornell Univ., Troms, Norway, June 1995.

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

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

[28] P. Maes, *Agents that Reduce Work and Information Overload.* Communications of the ACM, July 1994, Vol 37 No.7.

[29] 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.

[30] 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.

[31] S. McCanne and S. Floyd, *NS (Network Simulator)*. Website: http://www-nrg.ee.lbl.gov.

[32] J. M. McQuillan, I. Richer, and E. Rosen, *The New Routing Algorithm for the Arpanet*. IEEE Transaction on Communication, 28(5):711–719, May 1980.

[33] P. M. Merlin and A. Segall, *A Failsafe Distributed Routing Protocol*. IEEE Trans. Commun., vol. 27, September, 1979.

[34] D. Mills, *Exterior Gateway Protocol*. Network Information Center RFC 904, April 1984.

[35] 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.

[36] A. R. Mikler, J. S. K. Wong, and V. G. Honavar, *Quo Vadis - Adaptive Heuristics for Routing in Large Communication Networks*. Proc of the 3rd International Conference on Telecommunication Systems, Modeling and Analysis (1995): 66-75.

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

[38] N. Minar, K. 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.

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

[40] J. Moy, *OSPF version 2*. Internet Draft, RFC-2178, July 1997.

[41] S. Murthy, *Routing in Packet-Switched Networks using Path-Finding Algorithms*. Dissertation thesis, University of California - Santa Cruz (Computer Engineering), September 1996.

[42] D. Oran, *OSI IS-IS Intra-Domain Routing Protocol*. IETF RFC 1142, 1990.

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

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

[45] Y. Rekhter and T. Li, *A Border Gateway Protocol 4*. RFC 1771, March 1995.

[46] 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.

[47] D. Subramanian, P. Druschel, and J. Chen, *Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks*. IJCAI (2) 1997: 832-839.

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

[49] A. Tripathi and T. Noonan, *Design of a Remote Procedure Call System for Object-Oriented Distributed Programming*. Software-Practice and Experience, Vol. 28(1), 23-47, January 1998.

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

[51] 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.