

# Agent-Based Distance Vector Routing

Kaizar A. Amin, John T. Mayes and Armin R. Mikler

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

**Abstract.** Mobile Agents are being proposed for an increasing variety of applications. Distance Vector Routing (DVR) is an example of one application that can benefit from an agent-based approach. DVR algorithms, such as RIP, have been shown to cause considerable 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. However, in an agent-based solution, the number of messages is bounded by the number of agents in the system. In this paper, we present an agent-based solution to DVR. In addition, we will describe agent migration strategies that improve the performance of the route discovery process, namely Random Walk and Structured Walk.

## 1 Introduction

Routing, the process of selecting a communication path over which data can be sent in a network, is an important aspect of a communication network as it affects many other characteristics of the network performance. Most of the conventional routing algorithms are based on either of the two shortest path routing strategies, namely, *Distance Vector Routing* or *Link State Routing*. This paper focuses on Distance Vector Routing (DVR), an iterative, asynchronous and completely distributed routing algorithm [2]. Certain implementations of DVR such as RIP (Routing Information Protocol) are used widely in many networks [4] as they can be easily configured and maintained [9]. However, it has been shown [3] that a large number of update messages exchanged by adjacent nodes in a network constitute considerable resource overhead. This overhead is inflated due to the fact that many of these messages have little or no effect on the route discovery process. Reducing the resource overhead may allow for DVR-class algorithms to be deployed in a wide range of networks (wireless, ad-hoc) which require a simple routing protocol due to limited availability of resources (memory, bandwidth). Motivated by the need to reduce the resource overhead associated with DVR, and following recent developments in ant routing [2], a new implementation of DVR using an agent-based paradigm known as *Agent-Based Distance Vector Routing* (ADVRR) has been developed.

Agents, Software Agents, Intelligent Mobile Agents, and Softbots are terms, which describe the concept of mobile computing or mobile code ([12], [11]). The

mobile agent paradigm has attracted attention from many fields of computer science. The appeal of mobile agents is quite alluring - mobile agents roaming the Internet could search for information, meet and interact with other agents that roam the network or remain bound to a particular machine. Agents are being used or proposed for an increasingly wide variety of applications, ranging from comparatively small systems to large, open, complex real time systems. The agent paradigm offers a rich repertoire of features and lends itself to the formulation of solutions to computational problems in large distributed infrastructures. In these types of applications, knowledge of node-based parameters is often essential to make rational decisions. Load balancing and network routing are typical examples of such applications. To efficiently route packets through a large communication network, the constituent network nodes may require topology information for generating the routing maps or routing tables [7].

In Section 2, a brief overview of DVR is given along with an overview of different agent movement strategies. The various tools used to simulate the network environment are presented in Section 3. Section 4 gives a detailed analysis of experiments and results. Section 5 provides a summary of the paper along with the scope of future work with respect to the utility of agents in distributed networks.

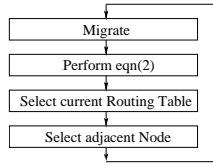
## 2 Agents in DVR

Distance vector routing (DVR) algorithms exchange a *metric* that represents the distance from a node  $n_i$  to any destination  $n_j$ . Distance is a generalized concept [5], which may include (but is not limited to) transmission delay on a link, monetary cost of traversing a link, resource reservation in sending messages, security level of links/nodes, or reliability measures. In most implementations of DVR this information (metric) is exchanged among adjacent nodes in the form of triggered updates, which is initiated when there is a change in the routing table of one of the neighboring nodes. After receiving the update information from a neighboring node, a node  $n_i$  updates its own routing table in the following manner:

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

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

While the message activity in conventional DVR can escalate to consume significant amounts of network resources, the number of messages in ADVN is bounded by the number of constituent agents in the network. In ADVN, the



**Fig. 1.** Agent Behavior

exchange of the metrics and the process of route discovery moves from the nodes to the agents [7]. Hence in this approach, the route discovery is manifested in the movement of agents carrying routing information from one node to another rather than the propagation of individual update messages. An agent can be formally described as:

$$A(i, x, y, R_x, \gamma)$$

where  $A$  is an Agent with ID  $i$  migrating from node  $n_x$  to node  $n_y$ , carrying the routing table  $R_x$  of  $n_x$  and using the migration strategy  $\gamma$  to move among adjacent nodes.

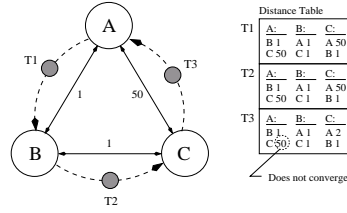
In ADVR, agents start at arbitrary nodes and migrate to adjacent nodes using  $\gamma$  as shown by Figure 1. On arriving at a node  $n_y$ , an agent  $A(i, x, y, R_x, \gamma)$  updates the routing table  $R_y$  based on the following equation:

$$D(y, j) = \min(D(y, j), [d(y, x) + D(x, j)]) \quad \forall n_j \text{ carried in the agent} \quad (2)$$

where  $D(x, j)$  is an entry in  $R_x$ . While equation(2) is based on equation(1), it is performed less frequently in ADVR as compared to DVR. The agent then selects  $R_i$  and migrates to an adjacent node using migration strategy  $\gamma$ .

## 2.1 Agent Migration Strategies ( $\gamma$ )

It has been shown in the previous section that, in ADVR, agents migrate among nodes, thereby establishing routes for every pair of nodes in the network in a distributed way. Hence, the efficiency of ADVR, in terms of the route discovery, is characterized by the migration strategy of the agents. It is important that the agents migrate intelligently, since an imprudent strategy can severely affect the performance of ADVR. To demonstrate this fact consider the following example of a three node ring graph as shown in Figure 2, with the following migration strategy: While migrating from a node  $n_i$ , the agent selects any node from a pool of nodes adjacent to the  $n_i$  at random. However, the agent will refrain from reversing its direction. This strategy assumes that a node would not benefit from consecutive visitations. Intuitively, this strategy would avoid looping between two immediately adjacent nodes. However, this may introduce an indirect looping problem, since, the agent will be forced into a loop (step T2  $\rightarrow$  step T3  $\rightarrow$  step T1 ...) not allowing ADVR to converge (see Figure 2). In general, deploying this scheme for any network topology may cause unnecessary looping and thus degrade the performance. Therefore, migration strategies for



**Fig. 2.** Example Migration Strategy

ADVR should be chosen carefully, as it might have severe side effects. An agent migration strategy ( $\gamma$ ) can be formally described as  $\gamma_{rw}(x, y, f(\cdot))$  where  $\gamma$  is the strategy to migrate from a node  $n_x$  to node  $n_y$  using the function  $f(\cdot)$  to select  $n_y$ . Different agent migration strategies can be formulated by changing  $f(\cdot)$ . This paper proposes two migration strategies, namely, Random Walk ( $\gamma_{rw}$ ) and Structured Walk ( $\gamma_{sw}$ ).

A Random Walk ( $\gamma_{rw}$ ) is an agent migration strategy in which  $f(\cdot)$  is a random function selecting an adjacent node  $n_y$  from a pool of nodes immediately adjacent to  $n_x$ . A Random Walk is a useful migration strategy due to its simplicity. It has been shown that, due to its probabilistic nature, a Random Walk will visit all nodes and edges (given infinite time) in a network thereby causing the system to converge [8].

A Structured Walk is a movement strategy which exhibits a deterministic behavior based on some criteria, such as congestion levels, topological information, and past visitations. In a Structured Walk ( $\gamma_{sw}$ ),  $f(\cdot)$  is a function that selects a node  $n_y$  for migration such that  $n_y$  satisfies the condition of minimizing or maximizing some decision criteria. For example, a Structured Walk may use  $\min(v)$  as a decision criterion, where  $v$  represents the frequency of node visitations by an agent. Efficiency of the Structured Walk depends on the calculation of  $v$  for every node. In what follow, we describe three different ways for calculating  $v$ , based on visitation of nodes, visitation of edges and a combination of both (Least First Walk).

When the selection criterion ( $f_{Node}(\cdot)$ ) for  $v$  is the number of node visitations, we refer to it as a Structured Walk on Nodes. In this case, upon visiting a node  $n_x$ , the agent increments the visit count  $v_x$  of that node. At the time of migration of the agent from a node  $n_y$  to its neighbor, the agent selects the adjacent node  $n_z$  which has  $v_z = \min[v_i] \forall n_i$  adjacent to  $n_y$ . When there is more than one node with the same  $\min(v)$ , the agent selects one at random. This scheme relies on the assumption that a node with fewer visitations will discover more routes when visited.

When the selection criteria ( $f_{Edge}(\cdot)$ ) for  $v$  is edge visitations, we refer to it as a Structured Walk on Edges. Whenever an agent traverses an undirected edge  $xy$ , connecting nodes  $n_x$  and  $n_y$ , it increments the visitation count  $v_{xy}$ . At the time of migrating from a node  $n_y$  to its neighbor, the agent selects an adjacent node  $n_z$  for which the connecting edge has a minimum  $v$ , i.e.  $v_{yz} = \min[v_{yi}] \forall$

$n_i$  adjacent to  $n_y$ . As with  $f_{Node}(\cdot)$ , multiple  $min(v)$  are resolved at random. Intuitively, a Structured Walk on nodes might improve route discovery, since in every step, the agent moves to a node that is either unvisited or least visited. This however may not be true, since route discovery involves finding the shortest path between nodes. Hence, it is important to explore all the paths that exist, making it beneficial to traverse all the edges (Structured Walk on Edges) in the network.

A combination of the above mentioned methods is referred to as a Structured Least First Walk ( $f_{LFW}(\cdot)$ ). This strategy is a slight modification to the Structured Walk on Edges. Whenever an agent traverses a node or an edge, it increments the respective visitation counts. At the time of migration from a node  $n_y$  to its neighbor, the agent selects an adjacent node  $n_z$  for which the sum of the visitation counts  $v_z$  of that node and the visitation count of the connecting edge  $v_{yz}$  is minimum. This is formally expressed as:

$$v_{ifw_{yz}} = v_z + v_{yz}$$

$$v_{ifw_{yz}} = \min[v_{ifw_{yi}}] \forall n_i \text{ adjacent to } n_y$$

Structured Least First Walk will aid multiple agents to coordinate their actions when traversing the network. Structured Least First Walk has been used for the experiments conducted as a part of the analysis of the ADVN.

### 3 Experimental Design

In this section, we describe our simulation environment and present the results of our experiments. The experiments focussed on providing a comparative analysis of ADVN vs. DVN. The simulation results indicate that agents with the most rudimentary of intelligence will bring the network to a connected/converged state. In addition, it is evident that although single agent systems will bring the network to a connected/converged state, multi-agent systems will take advantage of intrinsic parallelism and improve the connection/convergence pattern. The design and deployment of smarter agents improves the connectivity/convergence pattern, however, care must be taken when choosing an agent migration strategy.

Our performance analysis was based on the following criteria:

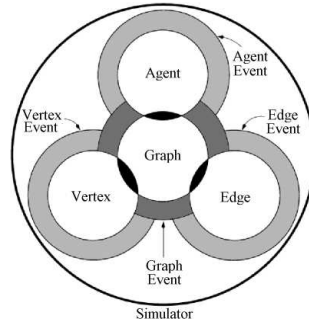
- *Connectivity* : The state of a network when every node in the network has discovered a path/route to every other node.
- *Convergence* : The state of the network when every node knows the optimal path (minimum cost) to every other node.
- *Message Efficiency*: The proportion of messages that cause an update of routing tables.

#### 3.1 Tools

To investigate the properties of agents in DVN, an event driven simulator and graph generator have been constructed. The simulator is based on an object-oriented paradigm [13] and includes methods for DVN, single agent

ADVR and multi-agent ADVR. The simulation model, as depicted by Figure 3, contains the following objects:

- Simulator - simulation engine for scheduling and dispatching events.
- Graph - container object that supplies a global view of all vertices and edges.
- Vertex - representation of a node/router that provides a routing table and methods for DVR.
- Edge - representation of a physical link between two vertices with an associated link cost.
- Agent - representation of a single agent containing methods for ADVR.
- Events - (Graph, Vertex, Edge and Agent Events) wrapper objects facilitating communication between the respective objects and the simulator.



**Fig. 3.** Simulation Model

A network is represented as a graph  $G(V, E)$  that is generated by the *graph generator*. The graph generator constructs pseudo-random, connected, undirected graphs with  $V$  nodes and  $E$  edges, given a random seed as input. A graph  $G(V, E)$  is generated in a two step process. First, the graph generator builds a random spanning tree containing  $|V| - 1$  edges as shown by Figure 4 lines 7 - 11, hence ensuring that the graph is connected. Secondly, it adds  $e - (|V| - 1)$  random edges from  $S - E[G]$ , 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, however, the details of the features of the graph generator are beyond the scope of this paper.

### 3.2 Experiments

In the experiments conducted, we have made certain underlying assumptions. We assume the network to be stable, i.e. edges and nodes are neither added nor deleted. The analysis does not cover the performance of the network after convergence of routing tables. The results of experiments that address link or node failure are beyond the scope of this paper and are discussed elsewhere.

```

MAKE-GRAPH( $V, e$ )
1.  $G \leftarrow V$ 
2.  $P \leftarrow \{\}$ 
3.  $D \leftarrow V$ 
4.  $v \leftarrow$  randomly chosen vertex of  $D$ 
5.  $C \leftarrow \{v\}$ 
6.  $D \leftarrow D - \{v\}$ 
7. for each random  $v \in D$  do
8.    $u \leftarrow$  random vertex in  $C$ 
9.    $E[G] \leftarrow E[G] + \{\{u, v\}, \{v, u\}\}$ 
10.   $D \leftarrow D - \{v\}$ 
11.   $C \leftarrow C + \{v\}$ 
12. for each  $\{u, v\} \in V \times V, \{u, v\} \notin E(G)$  do
13.  if  $u \neq v$  then
14.     $P \leftarrow P + \{\{u, v\}\}$ 
15. while  $|E[G]| < e$  do
16.   $\{u, v\} \leftarrow$  random edge in  $P$ 
17.   $E[G] \leftarrow E[G] + \{\{u, v\}, \{v, u\}\}$ 
18.   $P \leftarrow P - \{\{u, v\}, \{v, u\}\}$ 

```

**Fig. 4.** Pseudo-random Graph Generation Algorithm

For the simulation of DVR, we only consider triggered updates. However, timed updates will increase the resource overhead and further reduce the performance of DVR. Agent population is assumed to be static. Our experiments are based on three types of networks, namely small, medium and large. Small networks have 25 nodes, medium sized networks have 60 nodes and large networks have 100 nodes. Density of the network is defined by the number of links. A dense network ( $G(V, E)$ ) has number of bidirectional links  $|E|$  closer to  $\frac{V^2-V}{2}$  whereas, a sparse graph has links closer to  $|V|$ . Simulations were parameterized on the basis of network size, network density and simulation type (DVR or ADVR).

From equation(2), we see that an agent updates a routing table only if it has a lower cost to the destination. Therefore, on every update ADVR will bring the routing table closer to convergence. ADVR is characterized by reduced concurrency, as compared to DVR. The degree of concurrency in ADVR is bounded by the number of constituent agents. Figure 5a compares the convergence pattern of DVR vs. ADVR with different number of agents. DVR has a better initial convergence than ADVR, which is explained by the fact that DVR broadcasts messages to all its neighbors. On the other hand, ADVR is marked by the migration of agents which restrict the parallelism to the number of agents in the network. Hence the initial convergence rate for ADVR is proportional to the number of agents. Further, we observe that although the agents have a slow initial convergence, they compensate for it with their intelligent migration strategy. An important aspect for ADVR convergence is the agent population. Since the number of agents dictate the degree of parallelism of the algorithm, a large number of agents would exhibit better performance. However, the re-

source overhead increases proportionally with the size of the agent population. Therefore, performance and resource overhead constitute a tradeoff that must be carefully balanced by selecting an appropriate agent population. A characteristic of ADVR performance is the long convergence tail. This tail is due to the fact that there may be a small number of nodes in the network that have not yet converged. Their routing tables reflect a cost that deviates from optimal by  $\delta$ . The agents migrate among nodes until the network has converged. The size of the convergence tail is inversely proportional to the number of agents in the network. While this appears to be a drawback, in a realistic network environment, the total routing cost exhibit fluctuations larger than  $\delta$ .

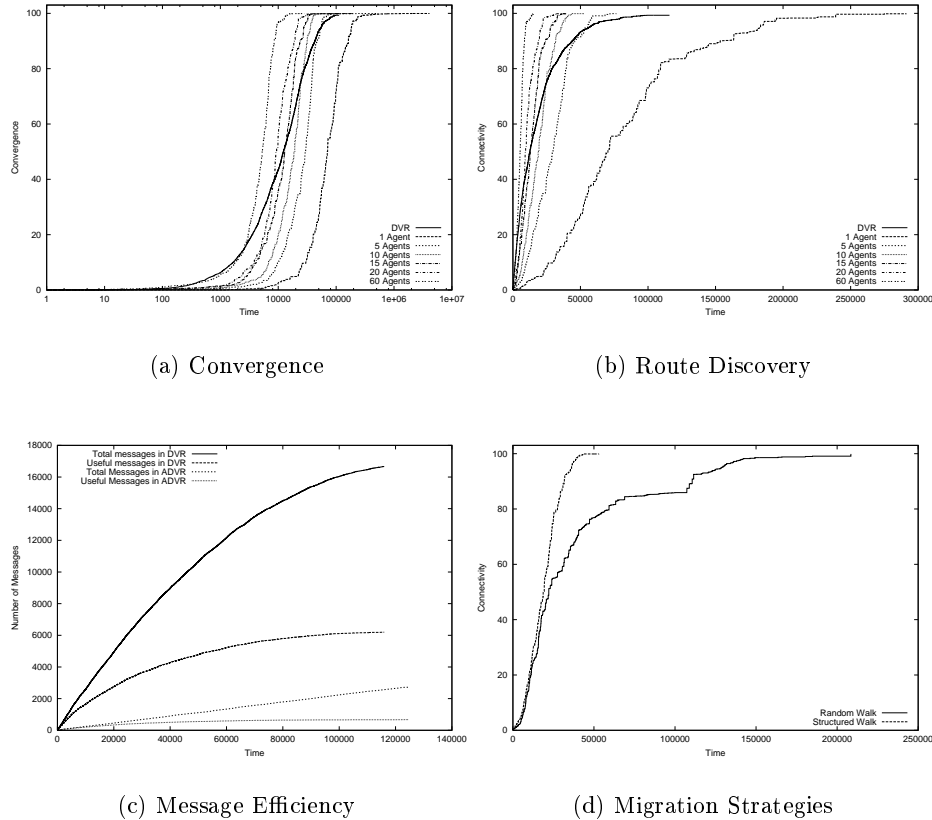
Route discovery plays an important role in network performance with respect to fault tolerance. Hence, it is crucial to evaluate any routing algorithm with respect to the speed at which routes between any two nodes can be obtained. Figure 5b depicts ADVR's progress in identifying routes in the network. The route discovery process for ADVR improves with an increase in the number of agents by exploiting concurrency.

As mentioned earlier, this paper aims at reducing the message overhead incurred by DVR. The large number of messages generated by DVR can be attributed to the highly concurrent and completely asynchronous behavior of DVR. In ADVR, the number of messages in the network is bounded by the number of constituent agents. Figure 5c compares the message efficiency for the two approaches. It indicates that the proportion of effective messages in ADVR is significantly higher as compared to DVR. Therefore, ADVR is suitable for wireless networks, with low resource (bandwidth) availability [10]. Intuitively, reducing the concurrency in an algorithm, reduces its performance. However, an appropriate migration strategy will improve the message efficiency, hence, ADVR can achieve superior performance with only  $c$  agents ( $c \leq n$ , where  $n$  is the number of nodes in the network).

As shown in Figure 2 that agent migration strategies can cause considerable side effects thereby delaying convergence of ADVR. Both, Random Walk and Structured Walk, can be applied to different classes of applications. While it can be shown that the two schemes yield comparable convergence, Structured Walk outperforms Random Walk migration with respect to the rate of route discovery (see Figure 5d). Therefore, a Structured Walk can be used in networks where early route discovery is crucial whereas, a Random Walk is applicable in systems which require a simple implementation.

## 4 Summary and Future Work

In this paper, we have described an agent-based paradigm for a Distance Vector Routing scheme (ADVR). In ADVR, intelligent mobile agents are the principle carriers of update messages transmitted between routers for the purpose of route computation. One of the major disadvantages of conventional implementations of distance vector routing algorithms is that their corresponding resource overhead is generally unbounded. That is, the overhead due to update messages will



**Fig. 5.** Simulation Results for 60 Node Network

increase proportionally with the size of the network. In the proposed ADVR, the messages are replaced by a population of agents. Hence, the overhead is bounded by the number of agents. However, by limiting the number of agents in order to control resource overhead, the degree of concurrency which the algorithm can employ is restricted as well. We have conducted a number of experiments to analyze the performance of an agent-based distance vector routing scheme. In particular, we have focused on agent migration strategies, agent population, convergence behavior, route discovery and message efficiency.

This paper has introduced the concept of a Structure Walk during which agents utilize specific runtime information which allows the agent(s) to migrate through large parts of the network efficiently. We have provided an example to demonstrate the significance of choosing an appropriate migration strategy to guarantee route table convergence. Through a number of carefully designed experiments, we have shown the quantitative improvements in route discovery

and cost convergence by increasing the number of agents in the constituent agent population. The convergence behavior, as well as the rate at which new routes can be discovered, have been compared to a conventional implementation of distance vector routing. Last but not least, we have quantified and compared the message efficiency of ADVR and DVR.

Ongoing research is focusing on fault tolerant routing in dynamic networks, tackling the *Counting to Infinity Problem*, and exploitation of dynamic agent population. While these issues are certainly important, their discussion was beyond the scope of this paper and will appear in a future publication.

## References

1. 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.
2. G. Di Caro and M. Dorigo, AntNet: Distributed stigmergetic control for communications networks, Technical Report 98-01, IRIDIA, Universite Libre de Bruxelles, 1998, (Accepted for publication in the *Journal of Artificial Intelligence Research (JAIR)*).
3. D. P. Bertsekas and R. G. Gallaher, *Data Networks*, Prentice-Hall, Englewood Cliffs, N.J., 1987.
4. J. F. Kurose and K. W. Ross. *Computer Networking, A Top Down Approach Featuring the Internet*, Addison-Wesley, 2001.
5. RFC 1058
6. N. Minar, K. H. Kramer and P. Maes, *Cooperating Mobile Agents for Mapping Networks*, Proceedings of the First Hungarian National Conference on Agent Based Computing, 1998.
7. N. Minar, K. H. Kramer and P. Maes, *Cooperating Mobile Agents for Dynamic Network Routing*, *Software Agents for Future Communications Systems*, Springer-Verlag, 1999, ISBN 3-540-65578-6.
8. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
9. The Mechanics of Routing Protocols, CISCO Press. <http://www.cisco.com/cpress/cc/td/cpress/fund/iprf/ip2907.htm>
10. M. Bui, S. K. Das, A. K. Datta and D. T. Nguyen, *Randomized Mobile Agent Based Routing in Wireless Networks*
11. A. Fugetta , G. P. Picco, G. Vigna, *Understanding Code Mobility*, *IEEE Trans. Softw. Eng.* 24(5), 342-361, 1998
12. J. M. Bradshaw *Software Agents*. AAAI Press, Menlo Park, California/The MIT Press.
13. A. R. Mikler, J. S. K. Wong and V. G. Honovar, *An Object-Oriented Approach to Simulate Large Communication Networks*. *The Journal of Systems and Software*. Volume 40, No.2. pp.61-73.