INFORMATICA, 2006, Vol. 17, No. 3, 407–426

An Agent-Based Best Effort Routing Technique for Load Balancing

Sunilkumar S. MANVI, * Pallapa VENKATARAM
Protocol Engineering and Technology (PET) UNIT
Electrical Communication Engineering Department, Indian Institute of Science
Bangalore-560012, India
e-mail: {sunil@protocol.ece.iisc.ernet.in, pallapa@ece.iisc.ernet.in}

Received: May 2005

Abstract. Several best effort schemes (next-hop routing) are used to transport the data in the Internet. Some of them do not perform flexible route computations to cope up with the network dynamics. With the recent trends in programmable networks, mobile agent technology seems to support more flexible, adaptable and distributed mechanism for routing. In this paper, we propose a Mobile Agent based Routing (MAR) scheme with the objectives similar to Routing Information Protocol (RIP). A comparative study of both the schemes (MAR and RIP) in terms of communication overheads, convergence time, network bandwidth utilization and average session delays is presented. The results demonstrate that the MAR scheme performs better than RIP. MAR has comparatively less communication overheads and convergence time and also offers more flexibility and adaptability as compared to RIP. In addition, this paper also presents a MAR based network load balancing.

Key words: routing, mobile agents, RIP, load balancing, network convergence.

1. Introduction

Routing of Internet datagram traffic consists of two basic tasks. The first task is to collect state (local/global) information of the network nodes and keep it up-to date. The second task is to find a path (shortest/optimal) for every new connection. The performance of a routing algorithm directly depends on how well the first task is solved.

A best effort service for a connectionless routing in the Internet provides qualitatively better service and aims to increase network bandwidth utilization, minimize administrative costs and decrease average session delays. Several best effort traditional routing algorithms use either Dijkstra (link state routing) or Bellman–Ford (distance vector routing) algorithm to compute the next-hop for an arrived packet that leads the packet to its destination (Schwartz and Stern, 1980; Bertsekas and Gallanger, 1992).

The metrics used for computing the paths can be either *static or dynamic*. Static metrics are link propagation delays, link costs and hop-count. The dynamic metrics are queue lengths, queuing delays and residual bandwidth. Some of the Internet routing protocols

*Presently working at Basaveshwar Engineering College, Bagalkot, India.

developed based on these metrics are RIP (Routing Information Protocol), OSPF (Open Shortest Path First) and IS-IS (Intermediate System-Intermediate System) (RFC; Perlman, 1999).

In this paper we propose a mobile agent based best effort routing scheme which employs routing procedure (next-hop routing) similar to RIP. Mobile agent is a piece of code containing data and program which moves from node to node by executing a given task, and destroys itself after the task completion. Several research efforts based on mobile-agent-like approaches are used to solve network routing problems. In an active network, the smart messages (programs) use self routing mechanism (Tennenhouse *et al.*). In the literature we observe that there are very few works available on agent technology applications in Internet routing. The work given in (Appleby and Steward) on mobile agent based routing was vague, and did not have a loop free route computation features which was later modified to overcome this drawback (Schoonderwoerd *et al.*, 1996).

The works given in (Bonabeau *et al.*, 1998; Lipperts and Kreller, 2000) use ant like agents for routing in telecommunication networks. The ant metaphor is derived from swarm intelligence of ants in finding shortest paths to their food from a nest (Bonabeau and Theraulaz, 2000). The work given in (Caro and Dorigo, 1998) describes two ant-colony algorithms for best effort routing. Ants adaptively build probabilistic routing tables based on the information they gather while roaming in a network. The work described in (Kramer *et al.*, 1999) uses active networking and mobile agents to perform dynamic routing in complex wireless networks. Mobile agents are also used for network topology discovery which facilitates network mapping and route computations (Minar *et al.*, 1999). The work given in (Gonzales–Valenzuela and Leung, 2002) describes an application of mobile agents for routing in MPLS (Multi Protocol Label Switching) networks. A QoS routing scheme using ant like mobile agents is presented in (Oida and Sekido, 2000).

We observe from the literature that none of works address flexible route computations by using mobile agents. This paper attempts to address the flexible route computations by demonstrating the load balanced route computations. In the proposed Mobile Agent based Routing (MAR) scheme, each node in a network sends routing mobile agents to all its non-neighbor node destinations to discover the routes (source to destination). The routing mobile agent tracks a path towards the specified destination through a set of intermediate nodes by updating the routing information at each of the nodes. To provide a loop free operation, a routing mobile agent marks the visited nodes and jumps to a node other than the marked nodes whenever it plans to migrate to a next node. To balance the network load, routing mobile agents compute the routes based on the existing load and the hop-count.

We have carried out the simulations of RIP and Mobile agent based routing with and without load balancing, and compared their performance in terms of convergence time, communication overheads, computational complexity, average session delays, delayed/rejected sessions and the network bandwidth utilization.

The next following section describes a best effort routing in Internet. Section 3 covers RIP based routing and its analysis. Mobile agent based routing and its analysis are

discussed in Section 4. A comparative analysis of RIP and MAR is given in Section 5. Simulation results are discussed in Section 6. Finally, benefits of mobile agents in routing and conclusions are presented in Sections 7 and 8 respectively.

2. Best Effort Routing

In this section we briefly describe the basic principle of best effort routing. There are two types of best effort routing: *source routing and hop-by-hop routing* (David and Chapin, 2002). In source routing (connection oriented routing), source puts the routing information in a packet and then sends it towards the destination. The job of the intermediate nodes in a network is simply to read the routing information from the packet and route accordingly. In hop-by-hop routing, the source is not expected to have all the routing information about how to reach the destination, it is sufficient if source knows only about the next-hop, and that node knows its next-hop, and so on, until the destination is reached (Chen and Nahrstedt, 1998).

The best effort routing protocols compute the paths based on static or dynamic metrics at regular time intervals. The dynamic metrics such as available bandwidth and queuing delays adapt to the network dynamics whereas static metrics such as hop-count and link costs do not adapt. The intradomain best effort routing protocols like RIP uses hop-count whereas OSPF uses delay and bandwidth as metrics. In a best effort routing, a node creates a forwarding table which consists of destination and next-hop node pairs. Whenever a packet arrives, the node searches for the packets' destination in the forwarding table and retrieves the next-hop node information for the corresponding destination, and schedules the transmission of packet to next-hop node. Traditional best effort routing techniques such as RIP, OSPF, and IS-IS do not perform dynamic load balancing.

In the next section, we describe the functions of RIP which has been coded for the comparison of our MAR scheme. RIP has been chosen because of its simplicity and distributed route computing nature (since MAR is also based on distributed computing).

3. RIP Based Routing

We briefly discuss the working of RIP and analyze its performance in terms of computational complexity, communication overheads and convergence time.

3.1. *RIP*

It is a routing information protocol used for intradomain routing in Internet within an autonomous system. Each router maintains a routing table in which an entry for a destination consists of following fields: number of hops required to reach the destination, next router along the route to destination, and an interface that will be used to forward packets addressed to the destination.

Routers periodically advertise their routing tables to their neighbors. The routers compute the new routes to a destination using the received advertisements (Obradovic, 2000).

The value of hops should be between 1 and 16, where 16 indicates infinity, i.e., hops $\geqslant$ 16 means unreachable. RIP is appropriate for systems that are less than 16 hops apart. An RIP packet contains a list of destination-hop pairs and a header. The header is 32 bits long and the destination-hop pair is 144 bits long, i.e., in general, a RIP packet has $32 + 144 * (N - 1)$ bits for a N node network.

A router receiving the route message, analyzes the message and forwards it to its neighbors except the router from where it has received. While forwarding the route messages, it updates its routing table if the existing hop-counts for a destination is greater than the incremented value of the received hop-count metric. The router advertises hop-count $= 16$ on the received interface. RIP uses Asynchronous Distributed Bellman–Ford algorithm (ADBF). At any given time router is in one of the states: idle, receiving advertisement or sending advertisement. RIP introduces more non-determinism in the order of updates.

3.2. *Analysis of RIP*

To analyze RIP consider the following.

- N node (router) network where maximum distance between the two nodes is at most $\max$, and each node has nb neighbors (average value).
- The number of iterations required to find a route for a source-destination pair at a node is equal to at most $\max$, where the number of computations required for choosing a minimum path to a destination in each iteration is at most $N - 1$.
- The size of a routing message (in bits) is given as:

$$s_{rip} = 32 + 144 * (N - 1). \tag{1}$$

- A node sends routing messages to its nb neighbors for at most $\max$ times (since a node in a worst case receives routing messages from all the nodes that are less than or equal to $\max$ hops away). To get the stable routes for all source-destination pairs a node performs this operation at least twice. Thus each node sends at most $2 * nb * \max$ routing messages to its neighbors. Thus the total number of messages (n_{rip}) sent across a network are:

$$n_{rip} = N * (2 * nb * \max). \tag{2}$$

- The expected inter-generation time of routing messages among the nodes is $E(\delta_{ig}^{rip})$. Thus, the time at which a last node generates routing messages (Δ_{rip}) with respect to a first node is:

$$\Delta_{rip} = E(\delta_{ig}^{rip}) * (N - 1). \tag{3}$$

- Each generated routing message passes through at most $\max + 1$ nodes. The expected processing delay at each node is $E(\delta_{nd}^{rip})$. Thus the total processing delay (pd_{rip}) of a routing message is:

$$pd_{rip} = (\max + 1) * E(\delta_{nd}^{rip}). \tag{4}$$

- Let δ_{ld} be the propagation delay on a link. Thus link propagation delays experienced by a routing message (ld_{rip}) is:

$$ld_{rip} = \delta_{ld} * \max . \tag{5}$$

- At each node, a routing message waits in a queue until the completion of service of at most nb routing messages, hence the total queuing delays (qd_{rip}) will be:

$$qd_{rip} = \max * nb * E(\delta_{nd}^{rip}). \tag{6}$$

Now we give the analysis of RIP in terms of computational complexity, communication overheads and the route convergence time.

Computational Complexity
It is defined as the number of worst case computations required to compute the shortest/optimal routes from a source to all destinations. The computational complexity of RIP is $O(N^2 * \max)$, since a node runs the algorithm for N times, in each time it computes routes for N nodes, and needs $\max$ alternatives to get converged routes.

Communication Overheads
It is defined as the bandwidth consumed by the routing messages to find the shortest/optimal routes. The routing overheads are computed as a product of number of generated routing messages and the size of a routing message. Communication overheads in RIP (co_{rip} in bits) is given as (by using Eqs. 1 and 2):

$$co_{rip} = s_{rip} * n_{rip}. \tag{7}$$

Route Convergence Time
It is defined as the time required to compute a set of stable paths in the network. The convergence time (ct_{rip}) of the network is computed as the sum of the following parameters: the relative time at which the last routing message is generated; processing, propagation and queuing delays of the last routing message to reach a node at most $\max$ hops away from it (by using Eqs. 3–6).

$$ct_{rip} = \Delta_{rip} + pd_{rip} + ld_{rip} + qd_{rip}. \tag{8}$$

4. Mobile Agent Based Routing

We have developed a mobile agent based routing (MAR) scheme with the similar objectives of RIP, and it also performs network load balancing as well. In this section, we first introduce the mobile agents, later we discuss the designed mobile agent based routing technique and its analysis.

4.1. *Mobile Agents*

Agents are the autonomous programs activated on an agent platform of a host. The agents use their own knowledge base to achieve the specified goals without disturbing the activities of the host. They have two special properties: mandatory and orthogonal, which make them different from the standard programs. Mandatory properties are: *autonomy, reactive, proactive and temporally continuous*. The orthogonal properties are: *communicative, mobile, learning and believable* (Cetus team, 2002; Manvi and Venkataram, 2004).

Mobile agent is an itinerant agent which contains program, data, execution state information, migrates from one host to another host in a heterogeneous network, and executes at a remote host until it completes the given task (Chess *et al.*, 1995). By nature mobile agents are flexible modular entities which can be created, deployed and deleted in real time. The mobile code should be platform independent, so that, it can execute at any remote host in a heterogeneous network environment. Inter-agent communication can be achieved either by message passing, RPC (Remote Procedure Call) or blackboard architecture (Wong *et al.*, 1999).

A mobile agent platform comprises of agents, agent server, interpreter and transport mechanisms. Agent server is responsible for receiving mobile agents and sending it for execution by local interpreter. Agents can be written in Java, Tcl, Perl and XML languages. Agent interpreter depends on the type of agent script/language used. An agent platform offers following services: creation of static and mobile agents, transport for mobile agents, security, communication messaging, and persistence. Some of the Java based agent platforms are: Aglets, Grasshopper, Concordia, Voyager and Odyssey (Chess *et al.*, 1995).

Agent based schemes offer several advantages as compared to traditional approaches: overcomes latency; reduces network traffic; work in heterogeneous environment; encapsulates protocols; flexibility; adaptability to network traffic characteristics and failures; software reusability and maintainability; and facilitates creation of customized dynamic software architectures (Lange and Oshima, 1999; Perdikeas *et al.*, 1999). However, several overheads are associated with the agent based schemes: creation of a standard agent platform; security to hosts from mobile agents and vice versa.

4.2. *Design Aspects of MAR Based Routing*

- Every node in a network maintains the routing table entries similar to RIP.
- To simplify the communication among the mobile agents routing table is maintained based on the principles of blackboard architecture.
- We assume the availability of a mobile agent platform at all the routers (nodes) in a network. However in case of non-availability of an agent platform, agents gather and update routing information by using traditional mechanisms.
- The manager agent at the source generates a route finding mobile agent to find the routes to the specified destinations.
- By considering the link load, route finding agent identifies the load balanced routes.

- The manager agent at every node synchronizes the routing table updating.

We describe the notations used in MAR scheme by considering the N node network.

Hop-count bound: It defines the maximum hops that an agent can travel which is equal to $\max$.

Agent visit-node information ($M[.]$): It is a part of data segment of an agent in which the addresses of the visited nodes are stored.

Neighbor set (nb^I): A neighbor set of a node I comprises of its neighboring nodes in the network, where $I\epsilon\{1, 2, .., N\}$.

Length of neighbor set (LNB^I): It defines the number of elements in the neighbor set of a node I.

Destination set (D^I): A destination set of a node I comprises of all non-neighboring nodes in the network, where $I\epsilon\{1, 2, .., N\}$. And $D^I = \{d^{I,1}, d^{I,2}, ..., d^{I,LD^I}\}$, where LD^I=length of the destination set.

Length of destination set (LD^I): It defines the number of elements in the destination set of a node I which is equal to $(N-1) - LNB^I$.

Link load ($lkload[x][y]$*)*: It defines the total load on a link x-y, where $x \in \{1, ..N\}$, and $y \in \{1, ..N\}$.

Link capacity ($C[x][y]$*)*: It is defined as the maximum capacity of a link x-y.

4.3. *Route Finding in MAR Scheme*

The route finding mobile agent constitutes two segments: data and procedures. The data segment stores the visit-node information and hop-count estimate. The agent procedures are used to perform the following operations: loop free agent movement, updating the routing tables, selection of a next-hop, and agent disposal. These procedures are located in an agent platform at a node.

The route finding mobile agent is triggered by the source node to find a best path to a specified destination in the destination set. While tracing a route, the agent updates the routing information at the visited nodes starting from the given source to establish a loop free path. After the finding the required route the agent is disposed. The measure hop-count bound avoids the agent traversing the long paths. The pseudo-code for mobile agent based routing algorithm is described in *Algorithm MAR*.

Algorithm MAR: Route finding by using mobile agents
{**Nomenclature**: S = source node, $d^{S,j} = j^{th}$ destination in destination set D^S, LD^S = number of destinations of a source S, $M[]$ =agent visit-node information, neb =neighbor}
Begin
1. For all $S = 1$ to N do
Begin
2. For all $j = 1$ to LD^S do
Begin
3. Create a route finding mobile agent at node S;
4. Pick up a destination $d^{S,j}$ from D^S;
5. Initialize the data in mobile agent: $\max$; visited-count, $c = 1$; hop-count estimate, $h = 0$; destination $= d^{S,j}$; visit-node information $M[1, .., \max] = 0$; and $M[c] = S$;

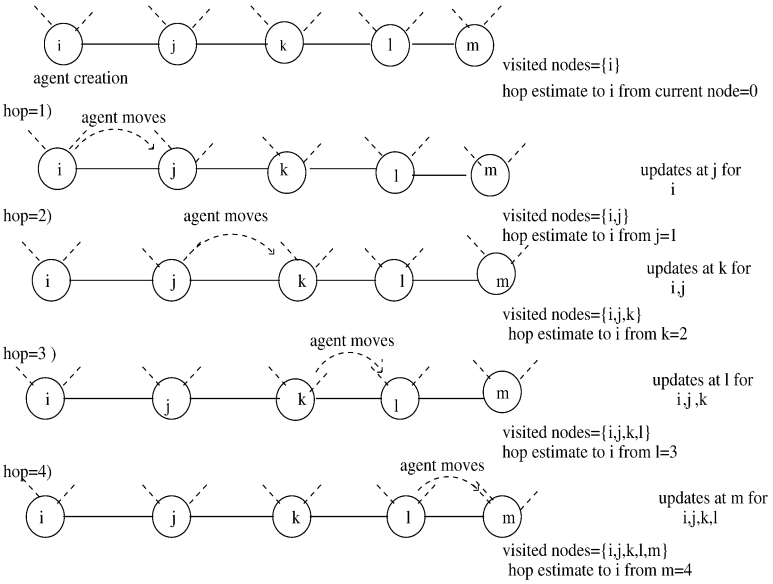

Fig. 1. Mobile agent based route establishment.

6. Mobile agent picks a neighbor neb from the neighbor set of $M[c]$ node with equal probability, such that the selected neighbor is not in array M;
7. If a distinct neighbor is found then migrate to neb otherwise goto Step 3.
8. Increment c and h ($c = c + 1$, $h = h + 1$). And update the array M with node address neb, i.e., $M[c] = neb$;
9. For all $c - 1$ visited nodes $M[1]$ to $M[c - 1]$, update the paths at $M[c]$ node:

- $th = h$;
- For $vn = 1$ to $c - 1$ do
  If existing hop-count for $M(vn)$ is greater than th then, update the path, i.e., set next-hop $= M[c - 1]$ and hop-count $= th$;
  $th = th - 1$;

10. If $M(c)$ == $d^{S,j}$ or h == max, then agent is disposed, otherwise goto Step 6;
Endfor j;
Endfor S;
11. Stop.
End.

In general, if an agent visits νk nodes, it establishes the routes to all the νk nodes from the source and updates the routing tables at all these nodes. In case, a route is not found for a specific destination, it is cheaper to find a route from its neighbors through simple message exchange mechanisms (justification to this is given in simulation results) rather than again deploying the mobile agents .

We explain the routing establishment process by considering i as a source node and m as a destination node in a network (see Fig. 1). We show in Fig. 1, a route finding mobile

agent initiated at node i traverses through nodes $j,\ k,\ l$ to reach node m by using MAR scheme.

1. The agent travels through nodes j, k, and l to reach the destination node m.
2. On the way to node m, it establishes the routes at (assumes existing hop-counts are greater than the hop-count estimates brought by the agent) :
   - destination k for the source node i and sets the next-hop as j to reach node i
   - destination l for the source node i and visited nodes j and sets the next-hop as k to reach nodes i and j.
   - destination m for the source node i and the visited nodes i, j and k and sets the next-hops as l to reach nodes $i,\ j$ and k.
3. Agent disposes itself after updating the routes at node m.

4.4. *MAR with Load Balancing*

A load balancing mechanism helps to increase the network efficiency and allows better throughput for the applications. In this work, we have experimented with MAR for dynamic load balancing of a network. We extend the MAR method to perform the load balancing (MARWL) in a network in a distributed fashion. Mobile agents gather the information besides the hop-count the existing load at every link which they traverse which will be used for establishing load balanced routes for a given source. The pseudocode given below describes the changes to be made in Step 8 and 10 of Algorithm MAR for load balancing:

Step 8:
Increment c;
Update the array M with node address neb: $M[c] = neb$;
$x = M[c-1]$; $y = M[c]$;
If($lkload[x][y] \leqslant (C[x][y]/4.0)$) then $h = h + 1$;
If($lkload[x][y] > (C[x)][y]/4.0)$ and $lkload[x][y] \leqslant (C[x][y]/2.0)$) then $h = h + 2$;
If($lkload[x][y] > (C[x][y]/2.0)$ and $lkload[x][y] \leqslant (C[x][y]/3.0)$) then $h = h + 3$;
If($lkload[x][y] > (C[x][y]/3.0)$ and $lkload[x][y] \leqslant C[x][y]$) then $h = h + 4$;
If ($lkload[x][y] > C[x][y]$) then $h = h + 5$;

Step 10:
If $M(c)$ == $d^{S,j}$ or $(c-1)$ == $\max$, then agent is disposed, otherwise goto Step 6.

MARWL translates the link load into the additional hop-counts to discourage the inclusion of heavy loaded links onto the path. The hop-count estimates may also be generated by using logarithmic or exponential relations ships between the maximum capacity of a link and the link load (in case of network load behavior following either of the relationships: logarithmic or exponential).

4.5. *Analysis of MAR*

In the analysis of MAR we consider the size of the agent, number of agents, time taken for the processing, etc., to compute the agent based overheads.

- For a node in N node network there will be $N-1$ agents visit to establish path to a single source, i.e., for all $N-1$ sources $(N-1)*(N-1)$ agents visit at each node.
- Each agent establishes route for at most $\max$ nodes.
- The data part consists of network addresses (32 bits), network family (5 bits) and $\max$ (4 bits, since maximum hop considered is 15) the maximum hop-count value. Thus the maximum bits required to store data part of an agent is $37 * \max$ bits.
- The procedures used in route finding agent are *comproute* and *snexth*. The *comproute* performs loop free agent migration, computes the routes and updates the routing table whereas *snexth* selects a next-hop for migration. Hence total number of bits required for calling these procedures and storing hop-count value is approximately 150 bits. Thus the agent size (in bits) is:

$$s_{mar} = 37 * \max + 150. \tag{9}$$

- The number of agents generated by each node in a network is $N - nb - 1$, where nb is number of neighbors. Thus the total number of agents generated in a network are:

$$n_{mar} = (N - nb - 1) * N. \tag{10}$$

  We also refer n_{mar} as an agent population.
- Average number of hops traveled by an agent is: $avg = (min + \max)/2$, where $min = 2$ (minimum hops). Thus the average number of hops traveled by the agent population in the network is given as (by using Eq. 10):

$$nh_{mar} = n_{mar} * avg. \tag{11}$$

- Each agent passes through $avg + 1$ nodes on an average before getting destroyed. At every node, it executes and migrates to a next-hop. Let the expected inter-generation time of agents among the nodes as $E(\delta_{ig}^{mar})$ time units. Thus the time at which the last node starts generating agents with respect to the first node is:

$$\Delta_{mar} = E(\delta_{ig}^{mar}) * (N-1). \tag{12}$$

- Let the expected agent execution time at each node is $E(\delta_{nd}^{mar})$ time units. Thus the total execution delays experienced by an agent on the traversed path is given as:

$$pd_{mar} = (avg + 1) * E(\delta_{nd}^{mar}). \tag{13}$$

- Let δ_{ld} be the propagation delay on a link. Thus the total time spent by an agent on the links of the traversed path is:

$$ld_{mar} = \delta_{ld} * avg. \tag{14}$$

- The queuing delays for an at a node are $nb * E(\delta_{nd}^{mar})$ time units. Thus the total queuing delays over the traversed path is given as:

$$qd_{mar} = avg * nb * E(\delta_{nd}^{mar}). \tag{15}$$

- In case, if a node does not have all the routes, it gets the routes from its neighbors, hence the delay in getting this information through message exchange from the neighbor is:

$$nbd_{mar} = 2 * (E(\delta_{nd}^{mar}) + \delta_{ld}). \tag{16}$$

Computational Complexity
It is defined as the number of agent computations required to compute the shortest/optimal routes from a source to all destinations. The worst case computational complexity of MAR is $O(N^2 * \max)$, since N^2 agents visit a node and each agent updates the route for $\max$ nodes.

Communication Overheads
It is defined as the bandwidth required by the routing mobile agents to find the shortest/optimal routes. The overheads are computed as a product of number of generated agents, an agent size, and the number of hops traveled by an agent. Thus the communication overheads in MAR is given as (by using Eqs. 9 and 11):

$$co_{mar} = s_{mar} * nh_{mar}. \tag{17}$$

Route Convergence Time
It is defined as the time required to compute all the stable paths in a network. Thus the convergence time of MAR of a network is computed as the sum of the following parameters: the relative time at which the last node generates agents; processing, propagation and queuing delays experienced by the last generated agent. It is computed by using the Eqs. 12–16.

$$ct_{mar} = \Delta_{mar} + pd_{mar} + ld_{mar} + qd_{mar} + nbd_{mar}. \tag{18}$$

5. Comparative Analysis

We consider three network configurations Net1, Net2 and Net3 to compute the communication overheads and convergence time in both RIP and MAR. The parameters of each network are: for Net1, $N = 15$, $nb = 3$, $\max = 6$ and $avg = 4$; for Net2, $N = 25$, $nb = 3$, $\max = 8$ and $avg = 5$; for Net3, $N = 40$, $neb = 5$, $\max = 10$ and $avg = 6$. Communication overheads for RIP and MAR are computed using Eqs. 7 and 17 respectively. We observe that MAR has lesser communication overheads as compared to RIP for all the network configurations (see Fig. 2).

Convergence time for RIP and MAR are computed using Eqs. 8 and 18 respectively. In the analysis we assume expected inter-generation time and expected computation time for both RIP and MAR are same: $E(\delta_{ig}^{rip}) = E(\delta_{ig}^{mar}) = 0.03$ seconds and $E(\delta_{nd}^{rip}) = E(\delta_{nd}^{mar}) = 0.045$ seconds. Let the link propagation delay be $\delta_{ld} = 0.1$ seconds. We observe that MAR converges slightly better than RIP for all the network configurations (see Fig. 3). Some times routes may not converge faster because of random walks of agents: in such a case, we may experience rise in convergence time.

We observed that number of worst case computations required to compute the paths from a source to all destinations in both RIP and MAR are same ($O(N^2 * \max)$). However the best case computational complexity will be much lesser in case of MAR, i.e., $O(N * \max)$ (a node executes an agent sent to it from all the nodes to get paths to all destinations) whereas in case of RIP, it is $O(N^2)$.

6. Simulation

We have simulated RIP and MAR (with and without load balancing) by using several network topologies (16 to 100 nodes). Both the algorithms were executed at regular intervals to compute the routes. In this section, we discuss the network traffic model used to test the proposed scheme, simulation procedure and the results.

Network configuration	RIP	MAR
Net 1	1.088 Mbits	0.245 Mbits
Net 2	4.14 Mbits	1.16 Mbits
Net 3	22.59 Mbits	4.2 Mbits

Fig. 2. Communication overheads in Net1, Net2 and Net3 configurations for RIP and MAR.

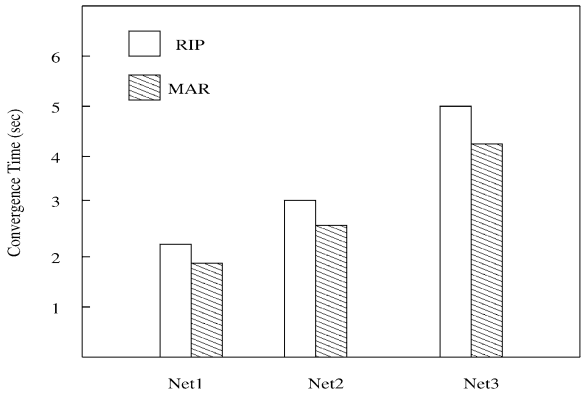

Fig. 3. Convergence time in Net1, Net2 and Net3 configurations.

6.1. *Network Traffic Model*

A network is modeled as connection of N nodes with each link capacity C. The link propagation delay is ld. The traffic in the network is generated by Ns contributing sources at any instant of time. The arrival rate λ, indicates the number of actual contributing sources among the Ns sources (for example, if $\lambda = 0.5$ and $Ns = 200$, actual contributing sources at any instant of time are 100 (0.5*200)). The source-destination node pairs are chosen randomly to generate the traffic. Bandwidth requirement of a source is randomly distributed within the range $[x, y]$ Mbps. The existing load on each link of a network is uniformly distributed within the range $[w, z]$ Mbps on some selected paths.

6.2. *Simulation Procedure*

The inputs to the simulation are as follows. N is considered as 16, 25 and 100 for different case studies; $C = 100$ Mbps; $ld = 0.1$ sec; consider $Ns = 300$, 1000, 2000 for different cases; $x = 1,\ y = 18,\ w = 20$, and $z = 120$. The maximum hop-count is 6, 8 and 10 for 16, 25 and 100 node topologies respectively. The average number of neighbors are 3, 3 and 8 for 16, 25 and 100 node topologies respectively. Inter-generation of RIP routing messages from nodes is uniformly distributed within the range, $[0.01, 0.05]$ sec. The computation time of a RIP routing message at a node is uniformly distributed within the range $[0.01, 0.08]$ seconds. The inter-generation time of agents between nodes are uniformly distributed within the range $[0.01, 0.05]$ sec. The computation time of an agent at a node is uniformly distributed within the range $[0.01, 0.08]$ sec (agent size is less than the RIP routing message, refer Eqs. 1 and 9). Simulation procedure is as follows.

Begin

1. Generate a partial connected network topology;
2. Apply the routing algorithms both RIP and MAR;
3. Compute the performance of the routing schemes;
4. Generate the required routes under the background load and load balancing method;
5. Compute the network performance under balanced load conditions with background and foreground load.

End.

The performance of RIP and MAR is evaluated in terms of following parameters:

• *Percentage of rejected or delayed sessions*: It is defined as the ratio of sessions which are unable to route immediately to the number of sessions requesting for immediate transmission (a session is considered to be rejected/delayed on its arrival at a node, if a node is unable to immediately route due to limitations of link capacity).

• *Network bandwidth utilization*: It is defined as the bandwidth utilized in all the links of a network.

• *Average session delay*: It is defined as the ratio of sum of the time required by all the sessions to transmit the information with required bandwidth to the number of sessions on the network.

6.3. *Results*

RIP generated 370 and 900 routing messages for 16 and 25 node topology respectively. MAR generated an agent link population (sum of number of hops traveled by each generated agent) of 717 and 2840 for 16 and 25 node topology respectively. The generated routing tables of both RIP and MAR were noticed to check for similarity of generated routes: RIP computed 100% shortest routes whereas MAR computed 95% shortest routes, and rest of the routes were either one or two hops longer than RIP routes. Communication overheads are less in MAR as compared to RIP and convergence time for MAR is better than RIP (see Fig. 4).

MAR performs almost similar to RIP in terms of session rejections, average session delays and network bandwidth utilization (see Figs. 5–10). We notice that MARWL shows increase in network bandwidth utilization and reduces the session rejections and delays as compared to both RIP and MAR.

Figs. 11–13 show the results of simulation of MARWL for 100 node topology. We notice that bandwidth utilization, rejected/delayed sessions and the session delays increase with increase in arrival rates and the number of sources. The communication overheads and the convergence time computed for this topology are 250.25 Mbits and 4.25 seconds, respectively. All the results indicate that mobile agent based routing performs better than RIP by providing more flexible and adaptable services.

	Convergence time		Communication overheads	
Topology	RIP	MAR	RIP	MAR
16 nodes	2.23 sec	1.92 sec	799.20 kbits	266.72 kbits
25 nodes	3.35 sec	2.98 sec	3.46 Mbits	1266.64 kbits

Fig. 4. Convergence time and Communication overheads for RIP and MAR (16 and 25 node topologies).

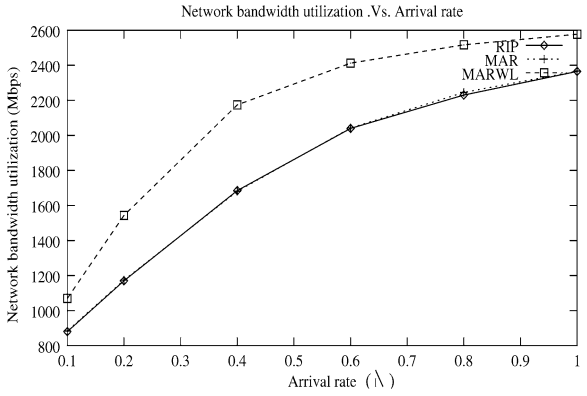

Fig. 5. Network bandwidth utilization .Vs. Arrival rate (16 node topology).

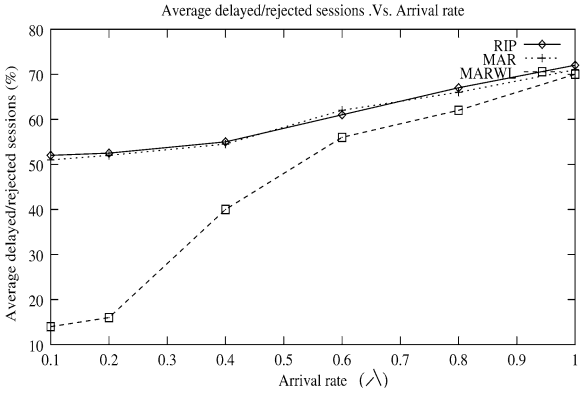

Fig. 6. Sessions delayed/rejected .Vs. Arrival rate (16 node topology).

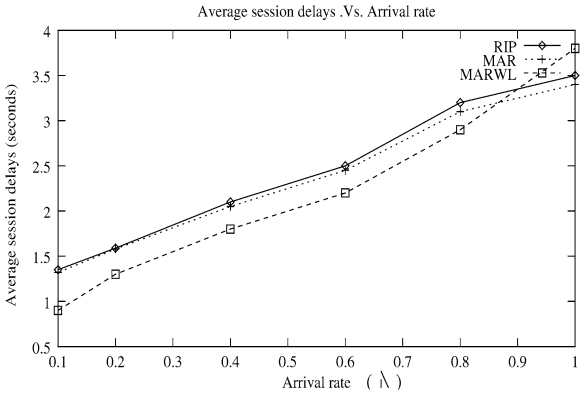

Fig. 7. Average session delays Vs. Arrival rate (16 node topology).

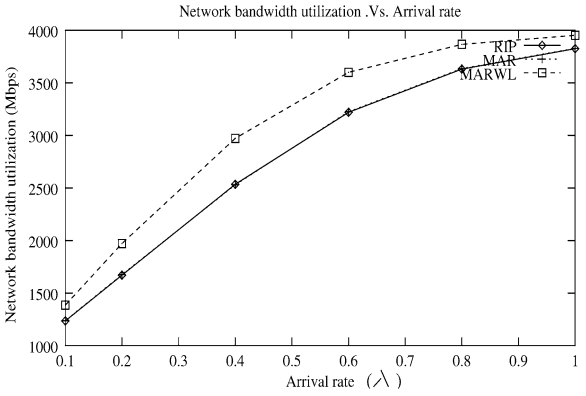

Fig. 8. Network bandwidth utilization .Vs. Arrival rate (25 node topology).

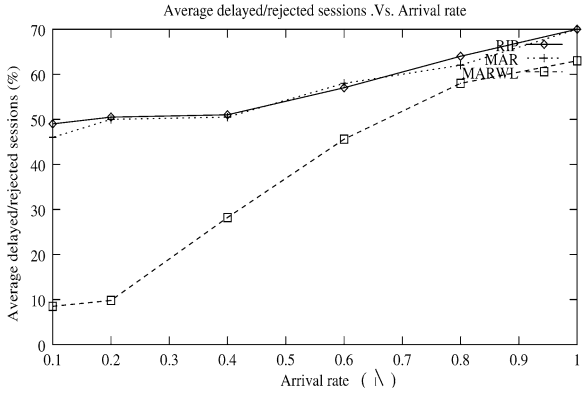

Fig. 9. Sessions delayed/rejected .Vs. Arrival rate (25 node topology).

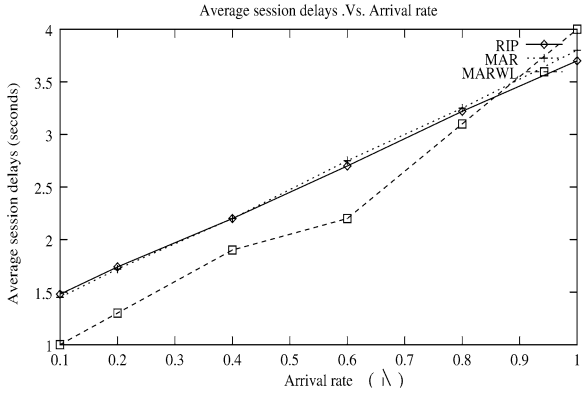

Fig. 10. Average session delays .Vs. Arrival rate (25 node topology).

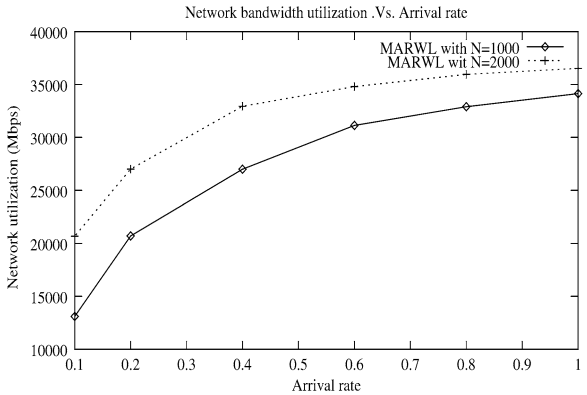

Fig. 11. Network bandwidth utilization .Vs. Arrival rate (100 node topology).

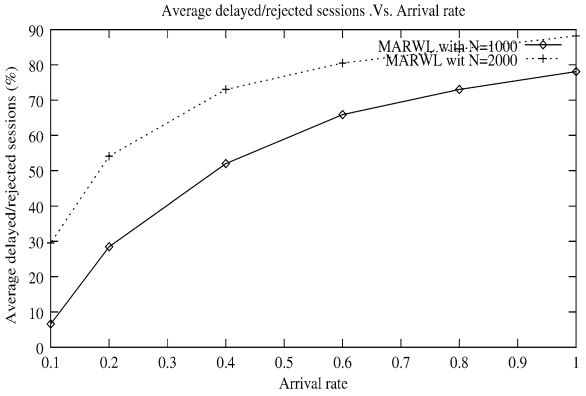

Fig. 12. Sessions delayed/rejected .Vs. Arrival rate (100 node topology).

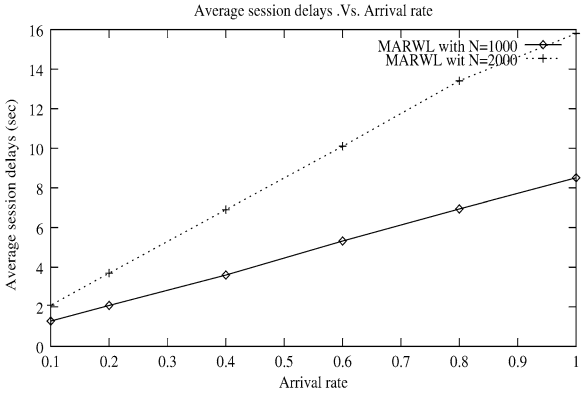

Fig. 13. Average session delays .Vs. Arrival rate (100 node topology).

7. Benefits of Agents in Routing

We experience that agent based routing scheme offers flexibility, scalability, efficiency, adaptability, software reusability and maintainability. Even though it is difficult to quantify these features, we explain below how they are demonstrated in the proposed routing scheme.

Flexibility: The agents allow learning capabilities to be incorporated in a natural way to find the routes. Asynchronous and intelligent route computations leads to smart networks. For example, the scheme facilitates change in routing metrics by encoding within the agent. Hence, at any time of day, we can have next-hop routing based on either hop-count, delay, bandwidth or mixed metrics depending on the network operating conditions and type of significant traffic (either delay sensitive or throughput sensitive).

Scalability: The proposed scheme restricts the agent movement till 15 nodes. The method is scalable to any size of the network and also can be implementable in the networks which have several clusters of different sizes without any modifications.

Efficiency: The scheme with load balancing showed significant improvements in the bandwidth utilization of a network as well as reduced the rejections and session delays.

Adaptability: The agents adapt to a network environment by creating load balanced routes. Agents can also be programmed to balance the load in a network by dynamically changing the routes based on the congestion level of the links, i.e., divert some of the traffic of randomly selected source-destination pairs by computing new patchup route for the congested link.

Reusabillity: The routing agents can be reused to perform multiple path computations with slight modifications in the agent code. Multiple paths computation are done based on different hop-counts, delays, and link loads, which facilitate routing of applications with different service requirements. This will also help in balancing the load across a network and reduce average packet delays and optimizes throughput. Multiple paths facilitate routing under link and node failures.

Maintainability: We can easily debug the agent components and also replace the old agent components with a new ones. For example, if we develop a new scheme of route finding based on current/future network situations, we can replace the route finding agent component.

Encapsulation of a protocol: Agents encapsulate the protocol in itself thereby avoids complex signaling mechanisms and the need to introduce the new protocols in the layer after going through a long process of standardization. Agents can be programmed to perform aggregate tasks of all other routing protocols.

Agent oriented programming facilitates the component based software engineering (CBSE) which is in great demand today (Griss and Pour, 1999; Jennings, 2001). Agent components can be independently developed by different developers, which can be selected and customized by a software architecture designer to create a dynamic software architecture. In future there will be enormous number of agents (agents are next generation components) which have to coordinate with each other to provide better multimedia information searching, retrieval and communication services.

8. Conclusions

We proposed an agent based routing scheme which uses routing mobile agents to generate the routing tables at every network node for best effort routing. An analysis and simulation of proposed scheme and RIP are carried out, and their performances are compared. The results demonstrated that the agents reduce the routing traffic, convergence time and perform better than RIP. We also showed how agents can be used to compute the load balanced paths to efficiently utilize the network bandwidth in case of uneven distribution of network load. Agent based routing architectures provide robust, flexible and distributed mechanisms to adapt to the network dynamics.

The work can be extended to hetergenous networks comprising of wireless and wired networks. Several set of agents can be generated that perform route computations separately in both wired and wireless ad-hoc networks. Mobile agent's asynchronous nature

can be exploited in mobile ad-hoc networks. The agent can be programmed to take into account the reliability, mobility and security factors for finding the load balanced and energy constrained paths in mobile nodes of ad-hoc networks.

References

Appleby. S and Steward. S, "Mobile software agents for control in telecommunication networks", *British Telecom Technology Journal*, vol. 12, pp. 104–113, 1994.

Bertsekas, D., R. Gallanger (1992). *Data Networks*. Prentice Hall, New York.

Bonabeau, E., F. Henaux, S. Gurin, D. Snyers, P. Kuntz, G. Thraulaz (1998). Routing in telecommunication networks with smart ant-like agents. *Proc. 2nd Intl. Workshop on Agents in Telecommunications Applications (IATA)*.

Bonabeau, E., and G. Theraulaz (2000). Swarm smarts. *Scientific American*, 55–61.

Caro, G.D., and M. Dorigo (1998). Two ant colony algorithms for best-effort routing in datagram networks. In *Proc. 10th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98)*. pp. 541–546.

Cetus team (2002). *Distributed Objects and Components: Mobile Agent* http://www.cetus-links.org/oo_mobile_agents.html

Chen, S., and K. Nahrstedt (1998). An overview of quality of service routing for next generation high speed networks: problems and solutions. *IEEE Network Mag.*, 64–79.

Chess, D., Benjamin, C. Harrison, D. Ievine, Colin, Paris (1995). Itinerant agents in mobile computing. *IEEE Personal Communication*, 35–49.

Chess, D., C. Harrison, A. Kershenbaum (1995). *Mobile Agents: are They a Good Idea?* IBM Research Division, T.J. Watson Research Center, Yorktown Heights, New York.

David, M.P, and A. Lyman Chapin (2002). *Introduction to Routing* http://www.corecom.com/html/OSNconnexions.html

Gonzales–Valenzuela, S., and V.C.M. Leung (2002). QoS routing for MPLS networks employing mobile agents. *IEEE Network Mag.*.

Griss, M.L., and G. Pour (1999). Accelerating development with agent components. *IEEE Computer Mag.*, **34**(5), 37–43.

Jennings, N.R. (2001). An agent-based approach for building complex software systems. *Communications of ACM*, **44**, 35–41.

Kramer, k.H., N. Minar, P. Maes (1999). Tutorial: mobile software agents for dynamic routing. *Mobile Computing and Communications Review*, **3**(2), 12–16.

Lange, D.B., and M. Oshima (1999). Seven good reasons for mobile agents. *Communications of ACM*, **42**, 88–89.

Lipperts S., and B. Kreller (2000). Mobile agents in telecommunications – A simulative approach to load balancing. *Proc. ISAS*.

Manvi, S.S., and P. Venkataram (2004). Applications of agent technology in communications: a review. *Computer Communications*, **15**(3), 1493–1508.

Minar, N., K.H. Kramer, P. Maes (1999). Cooperating mobile agents for mapping networks. In *Proc. First Hungarian National Conference on Agent Based Computation*.

Obradovic, D. (2000). *Formal Analysis of Convergence of Routing Protocols*. Phd Thesis, University of Pennsylvania.

Oida, K., and M. Sekido (2000). ARS: an efficient agent-based routing system for QoS guarantees. *Computer Communications*, **23**, 1437–1447.

Perdikeas, M.K., F.G. Chatzipapadopoulos, I.S. Venieris, G. Marino (1999). Mobile agent standards and available platforms. *Computer Networks*, **31**, 1999–2016.

Perlman, R. (1999). *Interconnections, Bridges and Routers*. Addison-Wesley, London.

RFC 1058 (RIP), RFC 2178 (OSPF), RFC 1142 (IS-IS).

Schoonderwoerd, R., O. Holland, J. Bruten, L. Rothkrantz (1996). Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, **5**(2).

Schwartz, M., T. Stern (1980). Routing techniques used in computer communication networks. *IEEE Trans. on Commn.*, **28**, 539–552.

Tennenhouse, D.L., J.M. Smith, W.D. Sincoskie, D.J. Wetherall, G.J. Minden (1997). A survey of active network research. *IEEE Communications Mag.*, **35**(1), 80–86.

Wong, D., N. Paciorek, D. Moore (1999). Java based mobile agents. *Communications of ACM*, **42**.

S.S. Manvi received ME degree in electronics from U.V.C.E, Bangalore University, India, in 1993 and PhD degree in electrical communication engineering from Indian Institute of Science, Bangalore, India, in 2004. He is currently working as professor and a head of Electronics and Communication Engineering Department, Basaveshwar Engineering College, Bagalkot, India. His areas of interest include multimedia communications and networking, applications of mobile agents and mobile computing. He has more than 60 national and international publications in referred journals and conferences. He has coauthored a book titled "Communication Protocol Engineering", published by PHI, India, in 2004. He has also served as program commitee member of several national and international conferences. He is a member of IEEE, USA, Fellow of IETE, India, and member of ISTE, India.

P. Venkataram received MSc degree in mathematics from Sri. Venkateswara University, Tirupathi, India, in 1973 and PhD degree in information sciences from University of Sheffield, U.K, in 1986. He is currently professor of Electrical Communication Engineering Department, Indian Institute of Science, Bangalore, India. His areas of research include wireless networks, computational intelligence in communication networks, protocol engineering and multimedia systems. He has visited several universities in India and abroad as visiting scientist and professor. He has more than 150 paper publications in referred conferences/journals, chapters in two books and edited a book. He has also authored a book titled "Communication Protocol Engineering", published by PHI, India, in 2004. He has served in various capacities in many IEEE and ICCC conferences and workshops. He is a fellow of IEE, U.K, a fellow of IETE, India, and a senior member of IEEE computer society, U.S.A.

Agentais grindžama geriausio bandymo maršrutizavimo schema srautams balansuoti

Sunilkumar S. MANVI, Pallapa VENKATARAM

Naudojami keli geriausio bandymo metodai duomenims į Internetą transportuoti. Maršrutų skaičiavimai, atliekami kai kuriais iš tų metodų, nesugeba lanksčiai susidoroti su tinklo dinamika. Mobilių agentų technologijos turėtų užtikrinti lankstesnį, adaptyvesnį ir labiau paskirstytą maršrutizavimo mechanizmą. Straipsnyje siūloma Mobiliais Agentais grindžiama Maršrutizavimo (MOM) schema, kurios tikslai panašūs į Maršrutų Informacijos Protokolo (MIP) schemą. Šių schemų lyginimo rezultatai rodo, kad MAM schema veikia geriau nei MIP schema. MAM lankstesnė, adaptyvesnė, jos konvergavimo laikas trumpesnis.