Mobile agent based approach for QoS routing

S.S. Manvi and P. Venkataram

Abstract: A mobile agent based on-demand quality of service (QoS) unicast routing scheme for supporting multimedia applications is proposed that considers bandwidth, delay and packet loss as QoS metrics for feasible path computation. A mobile agent is employed to find multiple QoS paths and select a best path among them to preserve resources so as to increase call success ratio and network bandwidth utilisation as well as adapt to network dynamics. The scheme is simulated in various network scenarios (sparse and dense networks) to verify performance and operation effectiveness, and compared with RSVP-based QoS routing by using an internet routing protocol. The results demonstrate significant improvements in call success ratio and network bandwidth utilisation compared with RSVP-based QoS routing, both in case of sparse and dense networks. Benefits of the agent-based scheme are adaptability, flexibility, and support for component-based software engineering features such as software reuse, customisation and maintainability.

1 Introduction

Multimedia applications such as video-on-demand (VoD), news-on-demand (NoD), video conferencing, etc., are more delay sensitive and less loss-sensitive, which require real-time delivery without retransmission. Two major communication modes commonly used are one-to-one (unicast communication such as VoD) and one-to-many (multicast communication such as video conferencing). Multimedia applications demand stringent quality-of-service (QoS) requirements from a network. Some of the important QoS metrics are guaranteed bandwidth, bounded end-to-end delays, minimal delay jitters and minimal packet-loss probability. Hence there is a need to develop a suitable routing scheme that satisfies QoS requirement of an application, increases the call-acceptance probability and optimises network bandwidth utilisation. QoS routing deals with identifying a route based on multiple QoS metrics and reserving the resources on the route to satisfy real-time delivery requirements of an application.

QoS routing is connection-oriented in contrast to connectionless best-effort routing [1, 2], which is categorised into two types: source and distributed routing. In source routing, all the information about how to get from source to destination is first collected at the source. Path computations are done at source, where each source maintains a complete topology by using link-state routing protocol. In distributed routing, the source is not expected to have all information about how to reach a destination; it is sufficient to know only how to get to a next-hop that leads to the destination. Path computations are distributed across the network nodes. Each node maintains local state and a list of neighbours by using distance vector protocol.

Source and distributed routing can be further classified into precomputed and on-demand QoS routing mechanisms. Precomputed routing mechanism maintains more state information at the routers, but reduces computational load on arrival of each connection request [3]. However, it suffers from inaccurate state information. To overcome this inaccuracy, state update messages can be sent frequently at the cost of larger consumption of network bandwidth. On-demand QoS path computation is independent of the aforementioned problem, but the routes are computed separately for each connection request. Feasibility and cost/benefit analysis for QoS routing has been studied by Apostolopoulos *et al.* [4], which shows that the cost of QoS routing is not excessive but involves some operational configurations.

Several QoS routing protocols have been developed based on the combination of different metrics such as hop-count, residual bandwidth, loss probability, delays, delay-jitter and buffers. These metrics can be either additive (delay), multiplicative (loss probability) or concave (bandwidth). The selection criteria of these metrics for QoS routing was discussed by Wang and Crowcroft [5]. Internet routing protocols such as RIP (routing information protocol), OSPF (open shortest path first) and IS–IS (intermediate system–intermediate system) are intradomain routing protocols employed for performing routing tasks within an autonomous system or a domain [6].

RIP uses hop-count as a routing metric and distance vector protocol for computing paths, but it does not support QoS routing. Resource reservation protocol (RSVP) and other QoS mechanisms may be used to support QoS routing over RIP that find a path satisfying the required bandwidth and resources reserve. OSPF uses only the delay metric for computing shortest paths even though it supports delay, throughput and reliability as routing metrics with link-state routing protocol. IS–IS employs cost, error rate and delay as metrics with link-state routing protocol. It computes several next-hops based on type of service (TOS). BGP (border gateway protocol) is an interdomain routing protocol used to route between autonomous systems (ASes). The draft, given by Bonaventure [7], proposes a flexible QoS attribute that can be used to distribute QoS information such as bandwidth and transit delay with BGP. Some of the QoS routing schemes are discussed in the following Section.

doi:10.1049/iet-com:20050457

Paper first received 25th January 2005 and in revised form 20th May 2006

The authors are with the Protocol Engineering and Technology Unit (PET)-UNIT, Electrical Communication Engineering Department, Indian Institute of Science, Bangalore 560012, India

S.S. Manvi is presently working at Basaveshwar Engineering College, Bagalkot, India

E-mail: agentsun2002@yahoo.com

This paper proposes a mobile-agent based on-demand QoS unicast routing scheme that can be used for multimedia services such as VoD and NoD. A mobile agent is a small code and data that migrates from one node to another to accomplish its task. The agent gathers state information about the visited nodes (nodes on the default route as computed by internet routing protocol) and its neighbours. Subsequently it constructs a partial network topology to compute multiple QoS paths and selects a feasible path among them. After a feasible path is found, the agent reserves resources along the path. If the agent fails in reserving resources at some link due to network load variations, it travels backward to release the reserved resources.

2 QoS routing techniques

Internet routing protocols such as OSPF and IS–IS have facilities to perform QoS routing. The IP community made two efforts to develop QoS based routing, that is integrated services and differentiated services [8]. Integrated services use RSVP (resource reservation protocol) to manage QoS requirements for individual flows. This supports both unicast and multicast flows. It requires maintenance of more state information.

RSVP uses existing IP routing protocols to determine the path by sending a PATH message from a source. A PATH message consists of bandwidth available along the path and application requirements. Receiver initiates soft reservation (periodic reservation refreshments) by sending RESV message across the path leading to source after receiving a PATH message [9]. Soft reservations create a greater amount of control traffic.

In differentiated services (Diffserv), QoS requirements are not exchanged as in RSVP. Type of Service (ToS) routing is performed based on packet class at each router, thus reducing state information. It uses a ToS field in the IP header to classify packets. Classification of packets is performed at ingress routers. Diffserv supports two types of PHB (per hop behaviour) classes, that is EF (expedited forwarding) and AF (assured forwarding). EF forwards faster with more packet drops, whereas AF forwards slower with fewer packet drops.

QoS routing with some extensions to OSPF (QOSPF) and efficient path management using RSVP with fewer modifications to routing protocols are presented in the work of Guerin *et al.* [10, 11]. Link bandwidth and propagation delay metrics are added to LSAs (link state advertisements) of existing OSPF; LSAs are triggered only when there is a significant change in the value of the routing metrics to reduce control overheads. RSVP is used to reserve the resources on explicit path computed by the QOSPF.

Flexible QoS attributes are used to distribute QoS information with BGP [7]. The attributes allow the BGP speaker to associate a set of supported PHB (AF, EF and, BE (best effort) forwarding), transit delay and bandwidth information of each supported PHB to a routing UPDATE message. Once a path for connection request is found, RSVP is used to reserve resources on the path.

The work of Apostolopoulos *et al.* [12] studies the performance of precomputed and on-demand QoS routing in terms of protocol overheads depending on policy for triggering updates, sensitivity of policy and update rates. Some of the QoS path-selection techniques for IP and ATM networks have been widely discussed [13–24]. The schemes deal with specific packet-scheduling disciplines such as RCSP (rate controlled static priority) and FCFS (first come first serve) to provide delay guarantees.

Internet QoS routing using the Bellman–Ford algorithm [25] deals with the main objective function as a number of hops and solves the bandwidth and delay-constrained problem. Multiple-path QoS routing [26] addresses the bandwidth-constrained multipath routing problem. K paths are computed to accommodate a stream. Shortest K-path routing compensates for the inaccuracy in link-state information and offers low blocking probability and balanced link utilisation. Some K-path computation algorithms using trellis graph and path-deletion methods have been presented [27, 28].

The link-state protocols requires more routing information database and often pose a problem of inaccurate state information, causing connection setup failures. The origins of this uncertainty are network dynamics, aggregation in large networks, hidden information and approximate calculations [29]. Several models (rate and/delay based) have been developed to provide QoS routes considering the inaccurate state information [30, 31]. The work discussed by Shaikh *et al.* [32] evaluates the impact of state link information on different network topologies, setup failures, pruning, long-tailed connection durations, and so on.

Ant-like agents and mobile-agent techniques have been proposed for best-effort routing in computer networks. An ant metaphor is derived from the swarm intelligence of ants in finding the shortest routes to their food from the nest [33]. The method of Di Caro and Dorigo [34] 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. They create probabilistic next-hops to reach a destination. The work given by Singh *et al.* [35] brings out certain advantages of mobile agents in QoS routing such as flexibility and adaptability in finding the paths based on given requirements. A bandwidth-constraint-based Antnet algorithm is presented by Oida and Sekido [36]. Other agent-based techniques for QoS unicast routing have been presented [37, 38].

From the literature, we have noticed that a dynamic QoS path computation, based on rapidly changing network conditions and capable of providing adaptability, flexibility, software reuse and customisability features, has not been addressed. Our work in this paper offers a solution to this problem by using mobile agents.

3 Mobile agents

Software 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 differentiates them from standard programs. Mandatory properties are autonomy, reactive, proactive and temporally continuous, while the orthogonal properties are communicative, human-like interactions, mobile, learning and believable [39, 40]. There are four different types of agent used in problem solving: local or user interface, network, distributed AI (artificial intelligence) and mobile agents. User interface and network agents are single-agent systems whereas distributed AI and mobile agents are multiagent systems.

A mobile agent is an itinerant agent that comprises code, data and execution state information. It migrates from one host to another in a heterogeneous network and executes at a remote host until it completes the given task [41, 42]. By nature, mobile agents are flexible modular entities which can be created, deployed and deleted in real time. Mobile code should be platform-independent, so that it can execute at any remote host in a heterogeneous network environment. Interagent communication can be achieved by message passing, RPC (remote procedure call) or common knowledge base (blackboard).

A mobile-agent platform comprises 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 either Java, Tcl, Perl or XML. Agent interpreter depends on the type of agent script/language used. An agent platform offers the 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.

Agent-based schemes comprising of static or mobile agents offer several advantages as compared with traditional approaches: reduced latency, works in heterogeneous environment, reduced network traffic, encapsulates protocols, flexibility, adaptability, software reusability and maintainability, and facilitates the creation of customised dynamic software architectures [43, 44]. However, mobile-agent technology is still in its infancy and has certain problems that have to be resolved. The problems are security, compatibility of agent coding languages with traditional programming methods, portability and standardisation. Standard bodies such as FIPA (Foundation of Intelligent Physical Agents) and OMG (Object Management Group) are involved in solving these problems and may come up with some solutions in the coming years to popularise the implementation of mobile-agent technology.

4 Mobile agent QoS routing

Proposed routing scheme identifies a set of multiple paths that meet the QoS requirements of a particular application, and selects a path which leads to highest overall resource efficiency. The scheme uses a mobile agent to perform this operation. Every node in a network comprises of an agent platform to support mobile agents. However, if an agent platform is not available, the agent uses a message passing mechanism to exchange information. Before discussion of the proposed routing scheme we describe the QoS routing metrics considered in the proposed work.

4.1 QoS routing metrics

Consider a network as an undirected graph $G(V, E)$ to describe QoS metrics used in the proposed scheme, where V is a set of nodes and E is a set of edges. A path P from source s to destination d, $P(s, d)$, is a sequence of edges belonging to set E. The proposed scheme uses residual bandwidth (bwe) and delay (de) metrics of a link for QoS routing of an application. A QoS of an application is specified as a tuple $Q = \{B, D, L\}$, where B is the maximum bandwidth required for an application, D is the bounded end-to-end delay for delivery of information, L is acceptable path loss without much degradation in service. The metric L is used to find the minimum guaranteed bandwidth $bw_{\min} = B - (B * L)$ required for an application.

We reduced three routing metrics B, D and L to two metrics $bw_{\min}$, and D by applying bounds on bandwidth requirements considering acceptable path loss L. Maximum and minimum bandwidth requirements for an application are $bw_{\max} = B$ and $bw_{\min}$, respectively. Network guarantees bw_{min} and can provide up to $bw_{\max}$ in the case of light loads. The application can be viewed by a user with acceptable QoS by guaranteeing bw_{min} and D.

Bandwidth is concave metric whereas delay is an additive metric. These additive and concave properties are defined below for a path $P = \{l_1, l_2, \ldots, l_n\}$, where l_n is the nth link and $m(P)$ is the metric value on the path P.

- *Additive:* A metric m is said to be additive for a given path P, if $m(P) = m(l_1) + m(l_2) + \cdots + m(l_n)$.
- *Concave:* A metric m is said to be concave for a given path P, if $m(P) = \min\{m(l_1); m(l_2), \ldots, m(l_n)\}$.

$P(s, d)$ should satisfy the following bandwidth and delay criteria (1) and (2) for an application to begin and progress:

$$bw_{P(s,d)} = \min_{(i,j)\varepsilon P(s,d)} bwe(i,j) \geq bw_{min} \tag{1}$$

$$delay_{P(s,d)} = \sum_{(i,j)\varepsilon P(s,d)} de(i,j) \leq D \tag{2}$$

Notations $bw_{P(s,d)}$ and $delay_{P(s,d)}$ are bandwidth and delay values associated with path $P(s, d)$, respectively, whereas notations $bwe(i, j)$ and $de(i, j)$ denote residual bandwidth and delay on a link connecting nodes i and j on the path $P(s, d)$, respectively.

4.2 QoS routing scheme

We assume that every node in a network maintains an agency for QoS routing (see Fig. 1). An agency consists of a static routing manager agent, static link monitoring agents, mobile agents for setting up a feasible path, and a QoS status profile.

- *QoS status profile* consists of information such as residual bandwidth, delays, jitters and percentage of packet losses on links that connect a node (see Fig. 2).
- *Routing manager agent (RMA)* is a static agent running at each node to serve the applications requesting for on-demand QoS routes and also support route finding operations when the node is acting as an intermediate node. This agent is responsible for creation of all the agents and the status profile in the agency. All the operations, either communication, updating the status profile, reading the status profile and so on, take place with the permission of routing manager agent.
- *Monitoring agent (MOA)* is a static agent that monitors the link. Such agents are created for each link by the

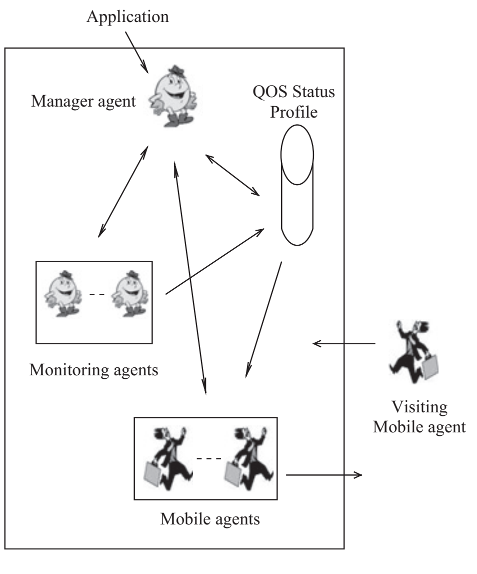
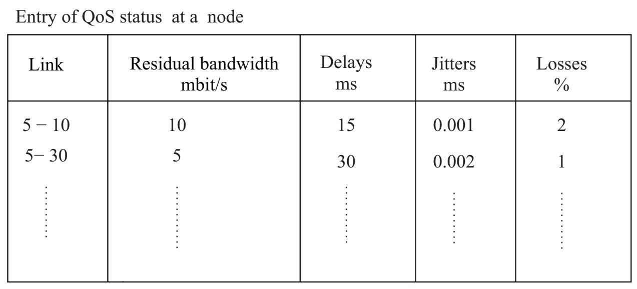


Fig. 1 *Agency for QoS routing*


Entry of QoS status at a node

Link	Residual bandwidth mbit/s	Delays ms	Jitters ms	Losses %
5 − 10	10	15	0.001	2
5− 30	5	30	0.002	1
⋮	⋮	⋮	⋮	⋮

Fig. 2 *Entry of QoS status at node 5*

routing manager agent. A monitoring agent computes the residual bandwidth, delays, jitters and packet losses for each link and updates the QoS status profile at regular intervals. All the parameters are computed within a given continuous time window. Residual bandwidth is calculated by monitoring flow of information on link of a node and the available bandwidth. Link delays are computed by averaging the queuing delays taken for all the packets on the link within a given time window. Link jitters are calculated by averaging the interarrival time of packets for all the connections passing through a link. Losses are computed as the percentage of packet losses among the received packets due to unavailability of buffers on a link.

• *Mobile agent (MA)* is a mobile agent invoked by the routing manager agent to find QoS routes on demand for an application/connection request. The details of QoS route finding using mobile agent is described subsequently.

The QoS routing scheme using mobile-agent functions is in two phases: path selection and path setup phases. The complete sequence of operations for the path selection phase is as follows.

4.2.1 Path selection phase

(i) *Request Arrival*. On arrival of a connection request for a source s at destination d with QoS requirements (minimum guaranteed bandwidth, maximum bandwidth, and bounded end-to-end-delay), MA is created by RMA at destination d to traverse on the path from d to s.
(ii) *Mobile agent gathers partial topology from d to s*. MA collects the neighbourhood connectivity (either first- or second-degree neighbours or as specified by an application) and status (residual bandwidth and delays) information of the visited nodes on the default path from d to s (default path as generated by internet routing protocol). First-degree neighbours are the directly connected nodes whereas second-degree neighbours are the nodes connected to directly connected neighbours. Probability of finding a feasible path increases by considering higher degree of neighbours, since MA gathers more connectivity information. If MA observes that all the links of neighbours of visited nodes do not have the required bandwidth and delays, MA informs RMA at d to reject application and disposes itself.
(iii) *Find QoS path*. If MA successfully reaches s, it finds the QoS paths that satisfies specified QoS by using following steps (refer to example in the following Section for an understanding of these steps).

• *Link pruning*. MA prunes all the links in the collected connectivity information that do not satisfy the minimum guaranteed bandwidth.
• *Find K multiple paths*. MA finds shortest path P_1 by using Dijkstra's algorithm [45], removes a link with lowest delay and finds path P_2 by using Dijkstra's algorithm, and so on up to P_K. If paths are not available, MA informs the RMA at d and disposes itself.
• *Check path for delay satisfaction*. Once the paths are found, MA checks the paths for eligibility of delay requirement satisfaction. If more than one path is available with delay requirements, the widest path is selected (path with maximum residual bandwidth) among the eligible paths. In the case of path unavailability, MA informs RMA at d and disposes itself.

The complete sequence of operations for the path setup phase is presented as follows.

4.2.2 Path setup phase

(i) *Resources reservation successful*. After a feasible path is selected, MA travels on selected feasible path from s to d to reserve the resources at all the nodes on the path. If reservation is successful at all the nodes on the path, then MA informs s and d about the resources reserved and disposes at d.
(ii) *Resources reservation unsuccessful*. If MA is unsuccessful in resource reservation at a node on the path (due to occupation of resources by other applications by the time agent reaches the node for reservation), MA sends message to visited nodes on path to release the reserved resources and informs s and d to reject the application.

This sequence of operations is repeated as and when required by an application such as in case of link failures and node failures. Thus the proposed scheme adapts to dynamic situations of a network. A mobile agent with multiple path finding capability performs adaptive QoS routing. Proposed scheme performs feasible path discovery among the discovered multiple paths and does not really perform multipath routing of data. However, available multiple paths may be used for mulipath routing of data to conceal the errors. Multiple path routing is better than a single optimum path because a single path between source-destination pair may suffer from network congestion. The proposed scheme gathers most recent state information, hence the probability of path setup failures are less.

As we observe from agent-based end-to-end schemes implementation by using IBM Aglets workbench, computation overheads are greater compared with traditional techniques due to immature agent platforms [47]. Agent platforms based on Java require supporting files to be imported for processing an agent, secure an agent and the routers. With the ongoing efforts by FIPA and OMG we may soon get an efficient agent platform which can work faster than traditional techniques. Moreover, importance of agent's capability of human like interactions for service agreements leads us to cognitive network management architectures.

4.3 Example

The functioning of the scheme is described by using Figs. 3–5. The topology shown in Fig. 2 comprises of six nodes with residual bandwidth and delays as indicated on the links; s is the source (server) and d is destination (client) which has the information of QoS requirement of a requesting application. Consider the QoS requirements of an application as $Q = \{4\,\text{Mbit/s};\ 6.5\,\text{s},\ 0.2\}$. Minimum bandwidth $bw_{\min}$ to be reserved is 3.2 Mbit/s $(4 - 4 * 0.2)$ and $bw_{\max} = 4$. A description of the two phases of the proposed scheme is as follows.

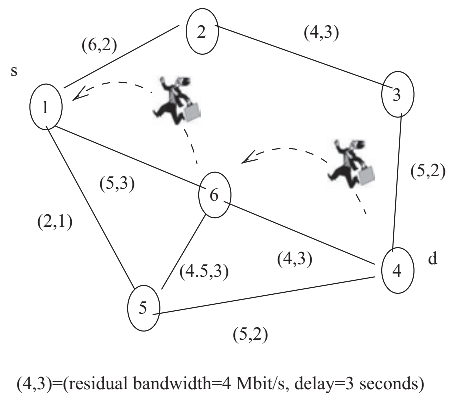


Fig. 3 *Network topology and agent migration*

4.3.1 Path selection phase: An agency at node d instantiates a mobile agent with the QoS specifications $bw_{\min}$ and D and sends agent across a network to traverse until it reaches s. A mobile agent routes itself from d to s as per the existing routing table in the nodes which are computed based on internet routing [45]. The mobile agent gathers the neighbourhood information of all the visited nodes in the path from d to s (in this example, path computed from $s = 1$ to $d = 4$ by routing protocol is assumed to be 1-6-4). The gathered information is used by the mobile agent to construct node connectivity status information from s to d, which is used to find a set of multiple paths from s to d (see Fig. 3).

Once the mobile agent reaches s, it finds QoS routes from server to client; $K = 2$ (number of multiple paths) is used to find the QoS routes. The agent prunes all the links (edges) with residual bandwidth less than 3.2 Mbit/s ($bw_{\min}$). Multiple paths computed after pruning are 1-6-4 and 1-6-5-4 that have residual bandwidth as 4 and 5 Mbit/s, respectively (Fig. 4). Path 1-6-5-4 fails in delay-bound satisfaction since path delay (8 s) is greater than the required delay (6.5 s). Thus the selected feasible path by agent is 1-6-4. In cases of availability of more than one path with both bandwidth and delay satisfaction, the agent chooses a path with maximum residual bandwidth (widest path).

4.3.2 Path setup phase: After a path is selected, the mobile agent travels on the selected path 1-6-4 to reserve the bandwidth and other required resources. In this phase, the mobile agent may fail to acquire the resources (called a path setup failure) because of network dynamics (change in load). In this case, if the mobile agent fails to reserve resources in any node on the path, it sends a message to all the visited nodes to release the reserved

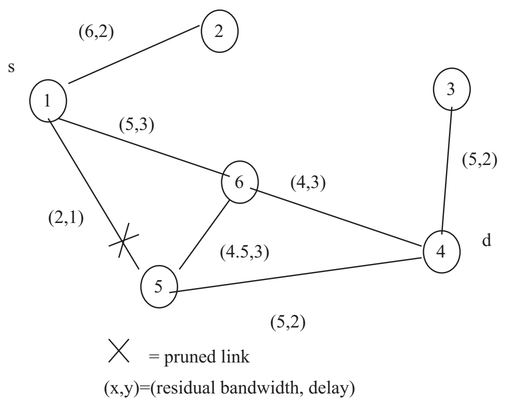


Fig. 4 *Network connectivity information gathered by mobile agent and pruning of links*

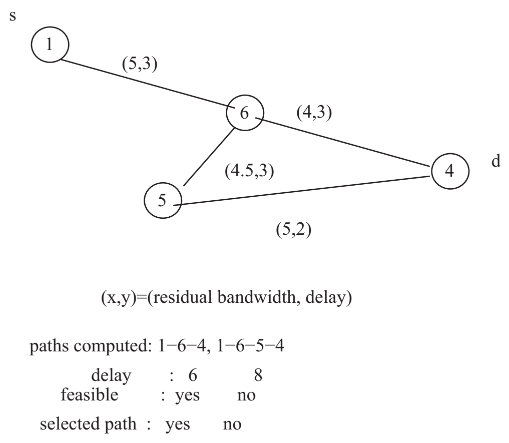


Fig. 5 *Multiple paths found by mobile agent*

resources, informs application rejection to d and destroys itself. However, the setup failure probability is low since the mobile agent has collected very recent information. Assuming no changes to the load in the considered example, bandwidth on the path 1-6-4 is reserved and the application's data will be routed on this path.

5 Analysis of path setup success

To analyse the mobile-agent-based QoS routing scheme in terms of probability of path setup success consider the following parameters.

- The existence of multiple shortest paths depend on neighbours of nodes on the default route and their connectivity. The probability that a mobile agent finds P_k shortest paths among existing Ka shortest paths $p_{\text{path}}^{\text{mul}}$ is given in (3)

$$p_{\text{path}}^{\text{mul}} = \begin{cases} P_k/K_a & P_k \text{ paths found} \\ 1/K_a & \text{single path found} \end{cases} \tag{3}$$

For example, if we assume $K_a = 2$, $P_k = 2$, then $p_{\text{path}}^{\text{mul}}$ will be 1 and 0.5 for multiple path and single path, respectively.

- Probability of finding a feasible path p^{fp} depends on residual bandwidth, staleness of information and required QoS. Say $bw_{\min}$ = minimum bandwidth required and, D = delay bound, then p^{fp} is expressed as (4)

$$p^{fp} = \left(1 - \frac{bw_{\min}}{(\gamma_{\text{res}}^{\text{bw}} - p_{\text{info}}^{\text{stale}} * E(\text{stale}))}\right) * p_D^{\text{satisfy}} * I_c \tag{4}$$

where $\gamma_{\text{res}}^{\text{bw}}$ = residual bandwidth on the path, $p_{\text{info}}^{\text{stale}}$ = probability of imprecise (stale) information about the links, $E(stale)$ = expected value of staleness which is uniformly distributed within the range $[x, y]$ (residual bandwidth might have been reduced by the time an agent travels on a mission to setup resources), and p_D^{satisfy} = probability that delay requirements are satisfied on the path.

- p_D^{satisfy} = probability that delay requirements are satisfied on the path is given in (5)

$$p_D^{\text{satisfy}} = \begin{cases} 1 & \text{if } \sum_{(i,j)\in path} de_{ij} \le D \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where I_c is an indicator function to indicate the availability of residual bandwidth on the path as given in (6)

$$I_c = \begin{cases} 1 & \dfrac{B}{(\gamma_{\text{res}}^{\text{bw}} - p_{\text{info}}^{\text{stale}} * E(\text{stale}))} > 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

- The residual bandwidth $\gamma_{\text{res}}^{\text{bw}}$ depends on $C =$ capacity on the path, and $\omega$the existing load. Hence its value can be computed as in (7)

$$\gamma_{res}^{bw} = C - \omega \tag{7}$$

where $\omega = \lambda_a * Na * E(bw_{\text{uti}})$, λa is the arrival rate of sources among Na sources, and $E(bw_{\text{uti}}) =$ expected bandwidth utilisation of a source. The maximum number of sources that can reserve a path can be computed by using $\lambda_a = 1$, $E(bw_{\text{uti}})$, ω and C.

The probability of path setup success is defined as the probability of an agent to find a QoS path and reserve the resources on it. This depends on several factors such as number of neighbours of nodes on the default route and their connectivity, staleness of state information, residual bandwidth on the links and end-to-end path delays. The probability that a mobile agent successfully finds a feasible route and reserves the resources ($p_{\text{success}}^{\text{setup}}$) is a product of $p_{\text{path}}^{\text{mul}}$ and p^{fp}. It is derived using (3)–(5) and is given in (8).

$$p_{\text{success}}^{\text{setup}} = p_{\text{path}}^{\text{mul}} * \left(1 - \frac{bw_{\min}}{(\gamma_{\text{res}}^{\text{bw}} - p_{\text{info}}^{\text{stale}} * E(\text{stale}))}\right) * p_D^{\text{satisfy}} * Ic \tag{8}$$

Observe from (8) that if $p_d^{\text{satisfy}} = 1$, then $p_{\text{success}}^{\text{setup}}$ increases with either increase in $p_{\text{path}}^{\text{mul}}$ or a decrease in the product of $p_{\text{info}}^{\text{stale}} * E(\text{stale})$, or an increase in $\gamma_{\text{res}}^{\text{bw}}$. If $p_D^{\text{satisfy}} = 0$, then $p_{\text{success}}^{\text{setup}} = 0$.

The analytical result for mobile agent success in finding a feasible path is depicted in Fig. 6. It shows the variation of $p_{\text{success}}^{\text{setup}}$ for different values of λ_a and $p_{\text{path}}^{\text{mul}} = 1$ (two paths found among the existing two paths, that is $P_k = 2$, $K_a = 2$) and 0.5 (single path found) by considering the constant values for $N_a = 12$, $p_D^{\text{satisfy}} = 1$, $bw_{\min} = 4$ Mbit/s, $p_{\text{info}}^{\text{stale}} = 0.2$, $E(\text{stale}) = 4$ Mbit/s, and $C = 50$ Mbit/s. We observe that the probability of setting up a feasible route is higher with availability of multiple paths. These paths are computed by a mobile agent itself. Thus proposed scheme increases the call success ratio of the QoS connection requests.

6 Simulation

We simulated the proposed scheme and RSVP-based QoS routing on sparse and dense networks to test the performance and operation effectiveness. In this Section, we describe the network traffic model used to test the proposed scheme, simulation procedure and the results.

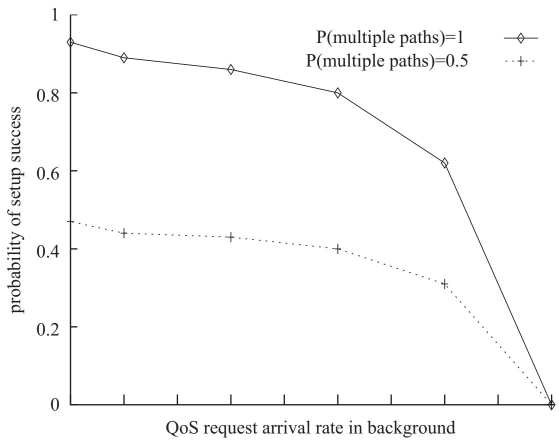


Fig. 6 *Probability of setting up feasible route by mobile agent against QoS request arrival rate in background* λ_a

6.1 Network traffic model

The model considers a physical topology of n nodes with same link capacity C Mbit/s and the propagation delays τ ms. The probability of node connectivity is defined to be pn, which varies for sparse and dense networks. The background traffic (existing load) generated on all the links is a certain fraction of the maximum capacity of a link. This background load is varied to create changes in the available network capacity. Imprecise (staleness) state information is modeled probabilistically as follows. The probability of increase in the link load after the state information is gathered by mobile agent is p_{inc}, probability of decrease in the link load after the state information is gathered by mobile agent is p_{dec}, probability of no change in the link load is p_{nc}, such that $p_{\text{inc}} + p_{\text{dec}} + p_{\text{nc}} = 1$. The agent computes K multiple paths by gathering information from dn-degree neighbours of the nodes on the default route. Agent size is given as as kbyte. Agent execution time (including migration) is uniformly distributed within ($ae1$, $ae2$) seconds, and agent waiting time in queues of nodes is uniformly distributed between ($w1$, $w2$) seconds.

The QoS requirements of an application are specified as a tuple $Q = \{\text{bandwidth, delay, loss}\}$, where all the metrics of Q are uniformly distributed. The maximum number QoS request arrivals at a node is N and the arrivals are Poisson-distributed with a mean interarrival time ξ; λ is the QoS request arrival rate from each of the client nodes; sn is the number of nodes randomly selected as multimedia servers (VoD/NoD) and other nodes as clients. The probability of selecting a server for a client is equally likely. Connection duration for an application is considered as infinite (long-lived applications). Internet routing protocols are used to generate routes and create routing tables at each of the nodes before applying the RSVP-based or the proposed routing scheme.

6.2 Simulation procedure

Consider the following simulation inputs to study the effectiveness of the proposed scheme. The simulation inputs are $n = 100$; $pn = 0.2$ and 0.6 for the sparse and dense network, respectively; $C = 50$ Mbit/s; $\tau = 100$ ms (propagation delay); background traffic on each link considered are 0, 40 and 70% for different cases; $p_{\text{inc}} = 0.4$, $p_{\text{dec}} = 0.2$, $p_{\text{nc}} = 0.4$; increase and decrease of link loads for staleness simulation are uniformly distributed, that is $U(3, 10)$ and $U(3, 5)$ Mbit/s, respectively; $sn = 10$; dn is considered as 1 and 2 for different cases; $K = 2$ and 3 for different cases; bandwidth, delay and loss requirements of an application are uniformly distributed within the ranges [3, 5] Mbit/s, [100, 500] ms, and [0.01, 0.2]%, respectively; λ is varied from 0.1 to 1.0; $N = 20$; $\xi = 0.5$; $as = 2$ kbyte; $ae1 = 0.1$ s, $ae2 = 0.15$ s; $w1 = 0.1$ and $w2 = 0.3$ s. Simulation procedure is as follows:

```
Begin
(i) Create a physical topology.
(ii) Choose random servers and clients.
(iii) Generate the background traffic.
(iv) Generate the application arrivals between a client and a random server.
(v) Deploy the proposed scheme.
(vi) Compute the performance parameters.
End
```

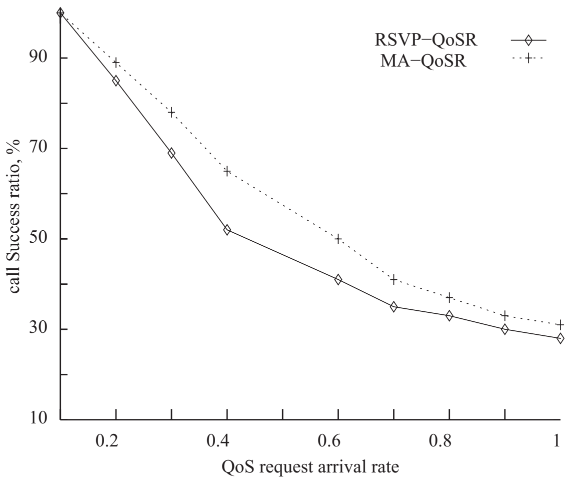


Fig. 7 *Call success ratio against QoS request arrival rate with 0% background load*

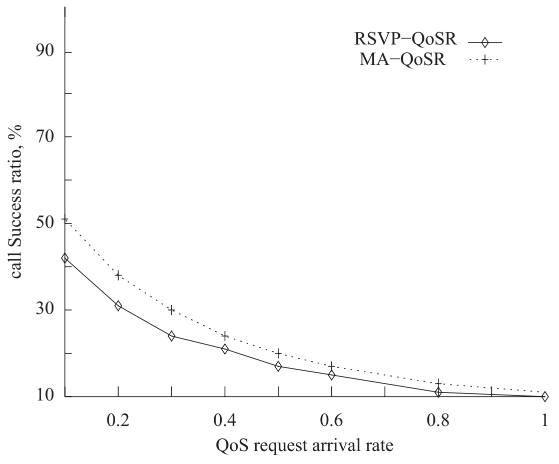


Fig. 9 *Call success ratio against QOS request arrival rate with 70% background load*

Performance parameters measured in the model are as follows:

- *Call success ratio:* defined as the ratio of number of QoS request arrivals accepted to number of QoS request arrivals generated.
- *Network bandwidth utilisation:* defined as the ratio of the sum of utilisation of all the links to the total capacity of a network.
- *Failed applications:* defined as the ratio of number of failed applications among the accepted applications due to link failures to the number of accepted applications (QoS routed applications).
- *Agent overheads:* defined as the ratio of sum of bandwidth used by all mobile agents successful in feasible path discovery to number of feasible paths discovered.
- *Agent response time:* defined as the ratio of sum of time taken by mobile agent of each successful discovered feasible route of QoS requested connections to number of QoS connection requests that are successful in discovering a feasible route.

6.3 Results

Figs. 7–9 depict the call success ratio for RSVP based and proposed QoS routing with different background loads (0, 40, 70%) for a sparse network with $K = 2$ and $dn = 1$ (the notations MA-QoSR and RSVP-QoSR denote mobile-agent-based QoS routing and RSVP-based QoS routing, respectively). We observe that agent-based routing has a higher call acceptance in all the cases as compared with RSVP-based routing. The improvement is due to capability of mobile agents to compute multiple paths and select a feasible path among them. Intuitively, the call success ratio declines with increase in the background load and QoS request arrival rate.

Figs. 10–12 depict network bandwidth utilisation for RSVP-based and the proposed QoS routing with different background loads (0, 40, 70%) for a sparse network with $K = 2$ and $dn = 1$. Network bandwidth utilisation is improved in mobile-agent-based QoS routing scheme compared with RSVP-based routing. The utilisation reaches a saturation level with the increase in values of λ.

We also observed call success ratio performance of the proposed scheme with $K = 2$ and 3 for both sparse and dense networks for $dn = 1$ as depicted in Fig. 13 (MA-QoSR : SP and MA-QoSR : DE denote mobile-agent-based QoS routing for sparse and dense networks, respectively). Call success ratio showed significant improvement in dense network for $K = 3$ compared with $K = 2$. Only marginal improvements are observed in a sparse network for $K = 3$ compared with $K = 2$. Hence it is better to compute two paths for a sparse network and more than two paths for a dense network.

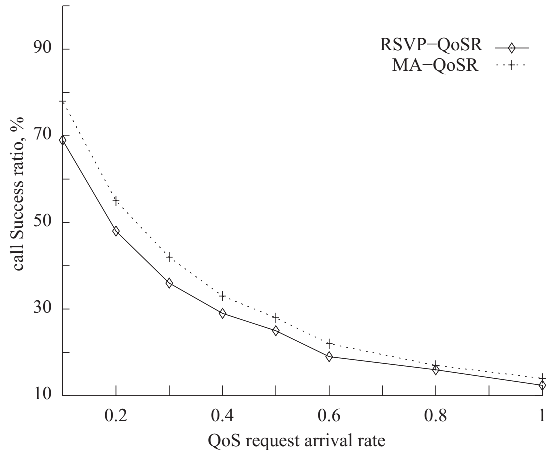


Fig. 8 *Call success ratio against QoS request arrival rate with 40% background load*

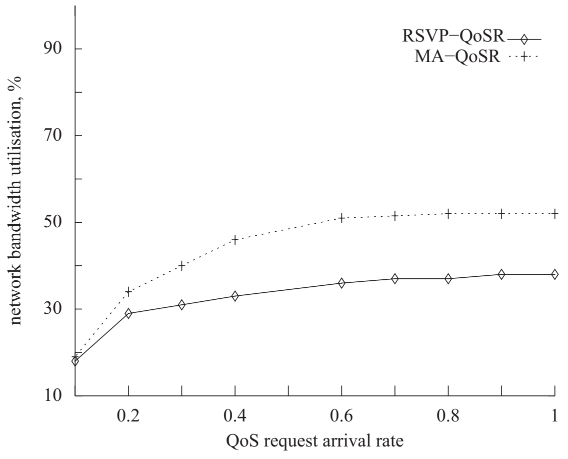


Fig. 10 *Network bandwidth utilisation against QoS request arrival rate with 0% background load*

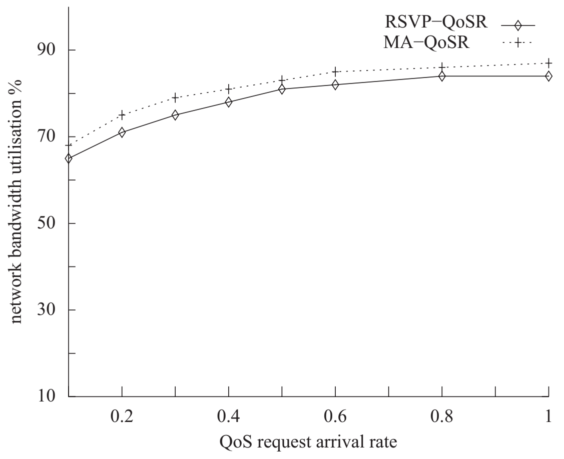


Fig. 11 *Network bandwidth utilisation against QoS request arrival rate with 40% background load*

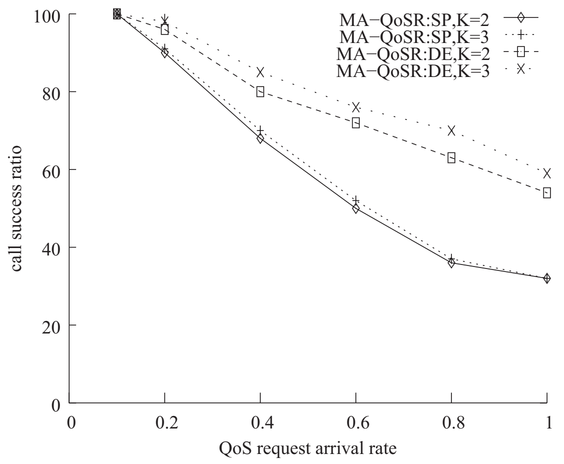


Fig. 13 *Call success ratio against QoS request arrival rate (with background load = 0) for dn = 1*

Effects of mobile agent's visibility while collecting the connectivity information is also studied by considering second-degree neighbours, that is $dn = 2$ (see Fig. 14). It is noticed that call success ratio is improved in the case of dense networks as compared with sparse networks with are increase in agent visibility while gathering connectivity information (see Figs. 13 and 14).

We experimented by injecting certain percentage of link failures when the applications are started. A mobile agent tries to find another QoS route for an affected application. However, in some cases it may fail and request the application to terminate (see Fig. 15). As the percentage of link failures increase, failed applications increase.

Agent response time increased with increase in QoS request arrival rate and network density (for $dn = 2$ and background load = 0). For a sparse network, minimum (for arrival rate = 0.1) and maximum (for arrival rate = 1.0) response times observed are 3.1 and 3.4 s, respectively, whereas for a dense network, minimum (for arrival rate = 0.1) and maximum (for arrival rate = 1.0) agent response times observed are 3.3 and 3.7 s, respectively. Agent response time includes both execution, waiting and migration time.

Agent overheads increased with increase in QoS request arrival rate and network density (for $dn = 2$ and background load = 0). For a sparse network, minimum (for arrival rate = 0.1) and maximum (for arrival rate = 1.0) agent overheads observed are 60 and 80 kbyte, respectively, whereas for a dense network, minimum (for arrival rate = 0.1) and maximum (for arrival rate = 1.0) agent overheads observed are 90 and 120 kbyte, respectively.

7 Benefits of using mobile agents

The proposed QoS routing scheme offers flexibility, scalability, efficiency, adaptability, software reusability and maintainability in the experiments conducted. Even though it is difficult to quantify these features, we explain how they are achieved with mobile-agent-based QoS routing scheme.

- *Flexibility:* The agents allow learning capabilities to be incorporated in a natural way to support delay predictions, bandwidth predictions, and playout decision-making based on the host architecture and network loads. For example, a QoS routing mobile agent can be encoded with some intelligence to trade the resources along the route depending on the user or service provider requirements. Flexibility can also be seen in providing the mobile agent code for personalising the services of the users. For example, a QoS routing mobile agent can make provision to take into account aggregate connections from a client to compute a QoS path or to negotiate the network resources in an optimal way.
- *Reusability:* Part of the QoS routing mobile-agent software can be reused by different types of multimedia applications by making slight modifications to the software. It is

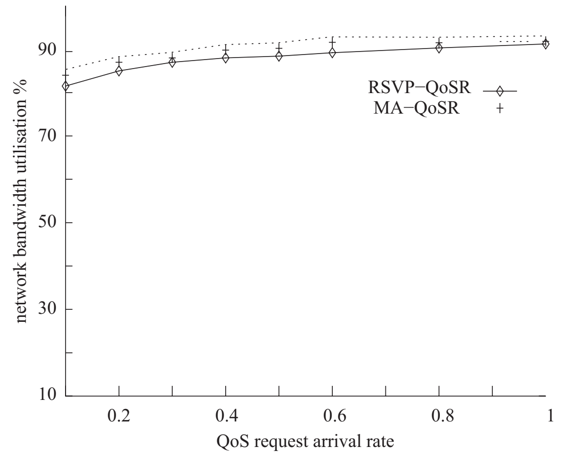


Fig. 12 *Network bandwidth utilisation against QoS request arrival rate with 70% background load*

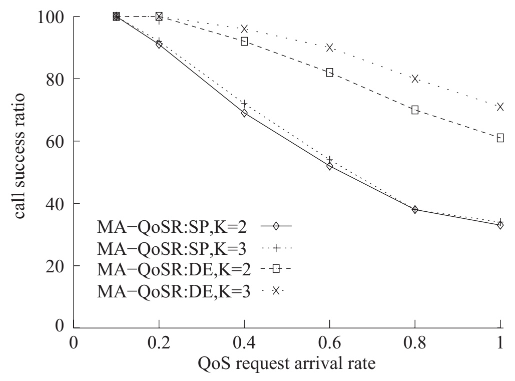


Fig. 14 *Call success ratio against QoS request arrival rate (with background load = 0) for dn = 2*

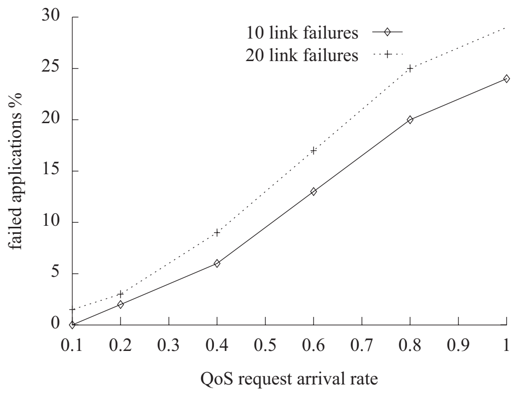


Fig. 15 *Failed applications (%) against QoS request arrival rate (with background load = 0)*

possible because of autonomous operation of the agents in agent-based systems. For example, an application may reuse the QoS routing mobile agent to collect the network management parameters from second/third degree neighbours of the visited nodes.

- *Maintainability:* The software can be easily maintained since every agent of an agent-based system is developed on a modular approach. Debugging and updating of the mobile agent software can be done with ease.
- *Adaptability:* The application using QoS routing mobile agent can easily adapt QoS routes to rapid changes in the network conditions (congestion/failure) and user requirements. For example, a QoS routing mobile agent dynamically computes the QoS routes as and when required (either due to congestion or link problems or user requirements).
- *Efficiency:* The use of a QoS routing mobile agent increases network resource utilisation efficiency because of its adaptability to network and user requirements and exchange of minimal information during task execution and decision making with multiple resource information.
- *Scalability:* The scalability can be achieved by using the QoS routing mobile agent at a client to perform similar tasks of several users. The scheme uses only certain degree neighbour's information to compute the multiple paths. Thus the scheme is scalable.

Agent-oriented programming facilitates component-based software engineering (CBSE), which is needed in today's software development process especially in web-based systems and Internet protocols [44, 46]. Agents are said to be the next-generation software development components. In future there will be enormous number of agents that co-ordinate with each other to provide flexible and cognitive multimedia information searching, retrieval, wireless/wired communication messaging services, and virtual market places.

8 Conclusions

A mobile-agent-based QoS routing scheme using three metrics, bandwidth, delay and loss for a feasible path computation, has been proposed. A comparison of RSVP-based and the proposed QoS routing presented in terms of call success ratio and network bandwidth utilisation for sparse and dense network scenarios. The results demonstrate that call success ratio and the network bandwidth utilisation of the proposed scheme is better than the RSVP-based QoS routing. The performance of the scheme is dependent on the richness of the network connectivity information gathered by a mobile agent, that is the scheme performs better in the case of dense networks. The agent's visibility of its visited nodes also plays an important role in improving the call success ratio. It is observed that second-degree neighbourhood information collection increases the richness of gathered topology and improves the call success ratio especially in the case of dense networks. Important benefits of the agent-based scheme as compared with traditional methods of software development are flexibility, adaptability, software reusability and maintainability. The scheme can be easily adapted to heterogeneous networks comprising of wired and mobile *ad hoc* networks by making minor modifications in the scheme to include mobility features and power constraints.

9 Acknowledgment

We thank the reviewers for valuable suggestions to improve the quality of the results and the paper.

10 References

1 Lee, W.C., Hluchyi, M.G., and Humblet, P.A.: 'Routing subject to quality of service constraints in integrated communication networks', *IEEE Netw.*, 1995, **9**, (4), pp. 46–55
2 Chen, S., and Nahrstedt, K.: 'An overview of quality of service routing for next-generation high-speed networks: Problems and solutions', *IEEE Netw. Mag.*, 1998, **12**, (6), pp. 64–79
3 Orda, A., and Sprintson, A.: 'QoS routing: The precomputation perspective'. Proc. IEEE Infocom, 2000, pp. 128–136
4 Apostolopoulos, G., Guerin, R., Kamat, S., Orda, A., and Tripathi, S.K.: 'Intradomain QoS routing in IP networks: a feasibility and cost/benefit analysis', *IEEE Netw.*, 1999, **13**, (5), pp. 42–54
5 Wang, Z., and Crowcroft, J.: 'Quality-of-service routing for supporting multimedia applications', *IEEE J. Sel. Areas. Commun.*, 1996, **14**, pp. 1228–1234
6 RFC 1058 (RIP), RFC 2178 (OSPF), RFC 1142 (IS-IS)
7 Bonaventure, O.: 'Using BGP to distribute flexible QoS information', IETF Draft, 2001
8 Weiss, W.: 'QoS with differentiated services', *Bell Labs Tech. J.*, 1998, **3**, (4), pp. 48–62
9 Braden, R., Clark, D., and Shenker, S.: 'Integrated services in Internet architecture'. RFC 1633, June 1994
10 Guerin, R., Orda, A., and Williams, D.: 'QoS routing mechanisms and OSPF extensions'. Proc. IEEE Globecom. Conf., 1997, pp. 1903–1908
11 Guerin, R., Kamat, S., and Herzog, S.: 'QoS path management with RSVP'. Proc. IEEE Globecom. Conf., 1997, pp. 1914–1917
12 Apostolopoulos, G., Guerin, R., Kamat, S., Orda, A., and Tripathi, S.K.: 'Quality of service based routing: a performance perspective'. Presented at ACM Conf. SIGCOMM, 1998
13 Matta, I., and Shankar, A.U.: 'Type-of-service routing in datagram delivery systems', *IEEE J. Sel. Areas Commun.*, 1995, **13**, pp. 1411–1425
14 Ghosh, D., Sarangan, V., and Acharya, R.: 'Quality-of-service routing in IP networks', *IEEE Trans. Multimed.*, 2001, **3**, pp. 200–208
15 Vogel, R., Guido, R., Kalfa, W., Wittig, H., *et al.*: 'QoS-based routing of multimedia streams in computer networks', *IEEE J. Sel. Areas. Commun.*, 1996, **14**, pp. 1235–1244
16 Ma, Q., and Steenkiste, P.: 'On path selection for traffic with bandwidth guarantees'. Presented at IEEE Conf. ICNP, 1997
17 Claypool, M., and Kannan, G.: 'Selective flooding for improved QoS routing'. Presented at SPIE Conf. on Quality of Service over Next-Generation Data Networks, 2001
18 Lee, K.-I., Kim, K.-I., *et al.*: 'QoS-based routing for integrated multimedia services'. Proc. IEEE Globecom Conf., 1997, pp. 1047–1056
19 Chen, S., and Nahrstedt, K.: 'Distributed QoS routing'. Technical report, http://citeseer.nj.nec.com/174923.html
20 Orda, A.: 'Routing with end-to-end QoS guarantees in broadband networks', *IEEE/ACM Trans. Netw.*, 1999, **7**, pp. 365–374
21 Vieira, S.L.: 'Efficient routing with quality-of-service requirements'. Proc. of IEEE Conf. ISCC, 2001, pp. 326–331
22 Layuan, L., and Chunling, L.: 'QoS-based routing algorithms for ATM networks', *Comput. Commun.*, 2001, **24**, pp. 416–421
23 Bolla, R., Davoli, F., *et al.*: 'QoS-aware routing in ATM and IP-over-ATM', *Comput Commun.*, 2001, **24**, pp. 811–821

24 Hou, C.J.J.: 'Routing virtual circuits with temporal QoS requirements in virtual path-based ATM networks', *IEEE Trans. Comput.*, 1999, **48**, pp. 1228–1243

25 Cavendish, D., and Gerla, M.: 'Internet QoS routing using the bellman–Fords algorithm'. Presented at IFIP Conf., 1998

26 Nikoladis, Y.J., and Gburzynski, P.: 'Multiple path QoS routing', http://citeseer.nj.nec.com/415156.html

27 Nikolopoulous, S., Pitsillides, A., and Tipper, D.: 'Addressing network survivability issues by finding *K*-best paths through trellis graph'. Presented at IEEE Infocom, 1997

28 de Queiros, E., Martins, V., Pascoal, M.M.B., and Esteves dos Santos, J.L.: 'A new improvement for *K* shortest-path algorithms', http://citeseer.nj.nec.com/article/martins00new.html

29 Lorenz, D., and Orda, A.: 'QoS routing in networks with uncertain parameters'. Proc. IEEE Infocom, 1998, pp. 3–9

30 Guerin, R., and Orda, A.: 'QoS routing in networks with inaccurate information: theory and algorithms', *IEEE/ACM Trans. Netw.*, 1999, **7**, pp. 350–363

31 Chen, S., and Nahrstedt, K.: 'Distributed QoS routing with imprecise state information'. Proc. IEEE Conf. ICCCN, Oct. 1998, pp. 614–621

32 Shaikh, A., Rexford, J., and Shin, K.: 'Evaluating the impact of stale link state on quality of service routing', *IEEE/ACM Trans. Netw.*, 2001, **9**, pp. 162–178

33 Bonabeau, E., and Theraulaz, G.: 'Swarm smarts', *Sci. Am.*, 2000, **282**, (3), pp. 72–79

34 Di Caro, G., and Dorigo, M.: 'Two ant-colony algorithms for best-effort routing in datagram networks'. Proc. IASTED Conf. PDCS, 1998, pp. 541–546

35 Singh, A., Manvi, S.S., and Venkataram, P.: 'QoS routing scheme by using mobile agents'. Presented at Indian Int. Conf. on Artificial Intelligence, Hyderabad, India, Dec. 2003

36 Oida, K., and Sekido, M.: 'ARS: an efficient agent-based routing system for QoS guarantees', *Comput. Commun.*, 2000, **23**, pp. 1437–1447

37 Reyes, A., Sanchez, E., and Barba, A.: 'Routing management application based on mobile agents on the Internet2', http://wwwtgs.cs.utwente.nl/Docs/eunice/summerschool/papers/paper9–2.pdf

38 Papavassiliou, S., Puliafito, A., Tomarchio, O., and Ye, J.: 'Integration of mobile agents and genetic algorithms for efficient dynamic network resource allocation'. Proc. ISCC, 2001, pp. 456–463

39 Manvi, S.S., and Venkataram, P.: 'Applications of agent technology in communications: a review', *Comput. Commun.*, 2004, **27**, (15), pp. 1493–1508

40 Mobile agent publications: http://www.cetus-links.org/oo_mobile_agents.html'

41 Chess, D., Benjamin, N., Harrison, C., Levine, D., and Paris, C.: 'Itinerant agents in mobile computing', *IEEE Pers. Commun.*, 1995, pp. 35–49

42 Wong, D., Paciorek, N., and Moore, D.: 'Java based mobile agents', *Commun. ACM*, 1999, **42**, (3), pp. 92–102

43 Chess, D., Harrison, C., and Kershenbaum, A.: 'Mobile agents: are they a good idea?'. IBM Research Division, T.J. Watson Research Center, Yorktown Heights, New York, March 1995

44 Lange, D.B., and Oshima, M.: 'Seven good reasons for mobile agents', *Commun. ACM*, 1999, **42**, pp. 88–89

45 Bertsekas, D., and Gallanger, R.: 'Data networks'. (Prentice-Hall, 1992)

46 Griss, M.L., and Pour, G.: 'Accelerating development with agent components', *IEEE Comput.*, 2001, **34**, (5), pp. 37–43

47 Manvi, S.S., and Venkataram, P.: 'Agent-based synchronization scheme for multimedia applications', *J. Syst. Softw.*, 2006, **79**, (5), pp. 701–703