

Agent-based subsystem for multimedia communications

S.S. Manvi and P. Venkataram

Abstract: There are several issues in multimedia communications, such as resource allocation, quality-of-service (QoS) routing, synchronisation, playout compensation and so on, which affect the continuous and smooth running of a multimedia application. The Internet demands the design of flexible and adaptable multimedia services so as to offer better quality presentations to the user. In this context, agent technology is emerging as a promising solution to provide flexible and adaptable services in distributed environments. The authors propose a subsystem called Protocol Engineering and Technology unit – Agent-based Subsystem for Multimedia Communications (PET-ASMAC) at the client side, which is an intelligent multimedia communication assistant to facilitate multimedia presentation to Internet users. The PET-ASMAC comprises a set of static and mobile agents, which perform certain autonomous tasks (QoS routing, resource allocation, synchronisation and playout) that satisfy the full-service requirements of an application, and coordinate with each other to provide smooth and continuous multimedia presentations. PET-ASMAC is simulated in several network scenarios to evaluate its effectiveness. The subsystem considers presentations from a single server to a client. However, it can be easily extended for multipoint-to-point communication. The results are encouraging and the observed benefits of the agent technology in multimedia communications are flexibility, adaptability, customisability, maintainability, re-usability and support to component-based software engineering.

1 Introduction

Multimedia communications deal with transporting of various media stream data with varied service characteristics such as constant bit rate (CBR) and variable bit rate (VBR) services. Multimedia services are integration of various types of media such as text, graphics, audio, video, animation, images and so on. Multimedia stream data is huge, hence consumes enormous amount of network bandwidth. Multimedia streams such as audio and video are continuous medias that are either coded using a CBR or VBR coder. The audio is usually coded as CBR and video is coded as VBR.

Multimedia applications can be categorised as presentational (real-time or non-real-time) and conversational (interactive and live). The presentational applications include video-on-demand, audio-on-demand, news-on-demand, real-time broadcast and so on. Conversational applications include tele-conferencing, video-conferencing and so on. These applications require real-time delivery services because late arrived packets of multimedia stream data is as good as lost [1], that is, they require stringent quality-of-service (QoS) to provide better quality presentation to the user.

The notable QoS parameters are guaranteed bandwidth, bounded end-to-end delays, acceptable delay jitters, sustainable packet loss, synchronisation and continuous playout of the streams. These parameters are defined as follows along with their importance [2].

- *Bandwidth:* It is defined as the data transmission rate, that is, the amount of data to be transferred for every second. For example, consider a sequence of pictures of a movie, which have to be transferred from a source to destination in a distributed system environment. If 25 pictures have to be transferred for every second with a picture size of 640×480 pixels and 24 bits/pixel, bandwidth required will be equal to 184.32 [$25 \times (640 \times 480) \times 24$] Mbps. Even when this high bandwidth value is reduced by data compression, the system needs a guaranteed bandwidth of several megabits per second (Mbps). If the bandwidth is not guaranteed, the application data will be delayed and also causes a loss of data when the buffers are not available at nodes.
- *Transfer or end-to-end delay:* Time between the multimedia data generation at a source and its presentation at the destination is subject to stringent bound (especially for interactive applications worst sustainable delay is 400 ms). This bound is expressed as transfer delay. An important example is a telephone call. Delays longer than several tenths of a second are unacceptable as they make conversation impossible because of large gaps induced between the media presentations.
- *Jitter:* It expresses the variance of the transfer delay. For a movie presentation, the jitter should be very slight to create the impression of uniform progress of the movie. Jitter can be reduced in the end-systems by the use of buffers. For example, if the maximum inter-arrival time difference of the packets of the stream is 2 s, size of each packet is

400 Kbits, packet generation rate is 30 packets/s, then buffers required at the client to compensate jitters is $2 \times 30 \times 400 \text{ k} = 24 \text{ Mbits}$ (or 60 packet-sized buffers). However, these buffers are quite large, thereby requiring large memory resources. Thus, it is better to have the jitter already controlled by the network nodes itself so that the buffer requirement at the client is reduced. In order to control jitters in the network nodes, proper schemes for delay parameter computation and negotiation by the client are required.

- *Loss rates*: This parameter defines the transfer reliability for continuous media. The losses may be because of delay-jitters or network losses. The losses create playout gaps, which degrade the presentation quality. Sustainable losses are 1–2% and 10% for audio and video, respectively.

- *Synchronisation*: This expresses the temporal relationship between the presentation units of a stream (intra-stream synchronisation) and among the streams (inter-stream synchronisation). For example, in the case of lip synchronisation, ± 80 to ± 100 ms delay is tolerable between the audio and video segments to be played. Tolerable time for intra-stream synchronisation is 5–10 ms. If media data are not presented within their tolerable limits, a discontinuous and rough presentation will be made to the user:

The maintenance of temporal relationships within a stream or among the multimedia streams usually depends on the following parameters:

1. *Network delays*: The delays experienced by the packets in the network to reach its receiver, which varies according to network load.
2. *Network jitters*: Delay variations of inter-arrival of packets at the receiver because of varying network load.
3. *End-system jitters*: Delay variations in presentation at the receiver because of varying workstation load and protocol processing delays.
4. *Clock skew*: The clock time difference between the sender and the receiver.
5. *Clock drift*: It is the rate of change of clock skew because of temperature differences or imperfections in crystal clocks.
6. *Rate drift*: Change in generation and presentation rates because of server and receiver load variations.
7. *Network skew*: It is the time difference in arrival of temporally related packets of streams, that is differential delay among the streams.
8. *Presentation skew*: It is the time interval in which the temporally related packets of the streams are presented.

The above discussed parameters can be tackled either individually or in an integrated manner.

1.1 Related work

Here, we give a brief overview of the works proposed to solve the multimedia communication issues. The goal of bandwidth allocation schemes are as follows: minimise packet losses, delays and re-negotiations and maximise network bandwidth utilisation. Several bandwidth allocation and reduction policies are generally based on factors such as network congestion, link/node failures, alternate backup paths, pricing, temporal resolution, type of service and scalable layered coding [3–5]. Internet protocols such as Resource reSerVation Protocol (RSVP) is used to reserve the resources for a flow and real-time control protocol (RTCP) is used in conjunction with real-time transport protocol (RTP) to monitor the bandwidth

used by an application and give feedback to the source for bandwidth allocation [6].

Some of the agent-based dynamic bandwidth allocation schemes are proposed in [7–13], where agents are employed by source and intermediate nodes to allocate bandwidth at the end-hosts and the intermediate nodes by interacting with a set of agents situated at each node. The allocations are mainly based on application bandwidth requirements, network congestion level and link failures.

Another important issue in multimedia communications is routing of application data based on QoS requirements. QoS routing is a method of finding QoS routes between a source and destination. If a proper QoS route is identified, the applications will meet the guaranteed services. QoS routes are identified on the basis of metrics such as bandwidth, delays, jitters and acceptable losses or combination of these. A survey of the QoS routing schemes is given in [14] under the following categories: inter-domain (between the domains), intra-domain (within a domain), precomputed and on-demand routing. The paper also discusses the impact of staleness of resource information while computing the QoS routes.

The precomputed routing scheme computes QoS paths by exchanging resource information among the network nodes and allows hop-by-hop routing, whereas on-demand routing scheme is source-based routing, that is, source decides the QoS route whenever it has to execute a multimedia application. The works given in [15–17] propose new schemes for QoS routing in IP networks based on precomputed routes by using bandwidth and loss probability. Internet community has addressed this issue by incorporating QoS in open shortest path first (OSPF) protocol and providing differentiated services to applications [18, 19]. In differentiated services, type of service routing is performed based on packet loss and the application bandwidth requirements at each router.

A multiple path based QoS computation is addressed in [20]. Multiple paths are computed based on bandwidth availability and delays. These multiple paths compensate for the inaccuracy in link state information and offers low blocking probability. Agent-based QoS routing schemes are proposed in [21–25]. These works are based on ant-colony concept employed in shortest path finding (ant metaphor detects the shortest QoS paths based on pheromone deposited by the ants where pheromone is a chemical substance deposited by the ant; pheromone deposition can be artificially generated based on bandwidth availability), gathering resource information from network nodes by sending mobile agents, and flooding mobile agents to find QoS paths from a source to destination.

A survey of inter-stream and intra-stream synchronisation schemes are presented in [26, 27]. The various synchronisation schemes proposed are mainly based on sustainable packet losses, buffers capacity, bounded end-to-end delay variations and delay jitters [28–30]. Agent-based synchronisation scheme is presented in [31, 32] which computes playout times and perform flexible synchronisation based on the delays and rate of change of delays.

The work given in [33] proposes a middle-ware design that calculates QoS attributes from the application needs and communication demands, and automatically configures suitable communication protocols to provide these as a run-time support. But this work lacks extensibility, customisability, adaptability, re-usability and maintainability features, and also do not support component-based software engineering (CBSE), which is essential for communication software development [34, 35].

1.2 Proposed work

Most of the works in the literature have been observed to be concentrating more on solving the issues independently. The problem associated with providing independent solutions are inflexible environment for an application, service customisability is not possible, creates playout gaps because of delayed actions and does not satisfy full service requirements of an application. Hence, there is a need to design an intelligent multimedia communications assistant to ensure smooth and continuous running of multimedia applications to support flexibility in full service requirements by incorporating CBSE concepts.

This paper addresses the multimedia communication requirements in an integrated manner using agent technology, as it will provide more robust, smooth and continuous presentations to the user by having control not only over the end-host but also over the network nodes. The use of agent technology adds-on flexibility, adaptability and CBSE features. We propose a subsystem called Protocol Engineering and Technology unit – Agent-based Subsystem for Multimedia Communications (PET-ASMAC) at the client side to cater to the needs of multimedia communications. PET-ASMAC performs QoS route finding, resource allocation (bandwidth and delay parameters), stream synchronisation and dynamic playout. It comprises a set static and mobile agents to carry out particular tasks to achieve smooth multimedia transfer and presentation. The agents are autonomous programs situated within the host or the network to perform dedicated tasks. It considers presentations from single server to the client. However, it can be extended to multipoint-to-point communications by extending features of agents.

2 PET-ASMAC: an intelligent assistant for multimedia communications

We propose an intelligent assistant for multimedia communications, which uses a set of static and mobile agents to facilitate smooth and continuous presentation of an application for point-to-point communication. PET-ASMAC assumes the availability of an agent platform at every node. However, in case of unavailability of agent platform, agents use message exchange mechanisms to achieve their task. In this section, we describe agent technology in brief and explain the functioning of PET-ASMAC.

2.1 Agent technology

Agent technology is emerging as a new paradigm in the areas of artificial intelligence and computing. Agents are said to become the next generation components in software development, because of its inherent structure and behaviour supporting CBSE [34], which can be used to facilitate Internet services.

Agents are the autonomous programs situated within a programming environment. The agents achieve their goals by collecting the relevant information from the host without affecting the local processing. They have certain special properties such as mandatory and orthogonal (supplementary), which make them different from the standard programs. The mandatory properties are autonomy, reactive, proactive and temporally continuous. The orthogonal properties are communicative, mobile, learning and believable [36, 37]. An agent should possess the mandatory properties which are compulsory. The orthogonal properties enhance the capabilities of agents and provide strong notion of agents. An agent may or may not possess the orthogonal properties.

Agents can be classified as local/user interface agents, networked agents, distributed artificial intelligence (AI) agents and mobile agents. The networked agents and user interface agents are single agent systems, whereas the other two types of agents are multi-agent systems [38].

An agent platform is required in an environment to support the functioning of agents. The platform consists of agent server, agents, agent interpreter and transport mechanisms. An agent platform offers the following services: creation of static and mobile agents, agent transport for mobile agents, security, communication messaging, persistence and fault tolerance. Some of the popular Java-based agent platforms such as IBM Aglets work bench, Grasshopper, Voyager, Odyssey are discussed in [39] along with the agent standardisation bodies such as Foundation for Intelligent Physical Agents (FIPA) and Object Management Group (OMG).

In general, there are several good reasons in using agent technology: flexibility in services; adaptation to changing environment; services customisation; reduce network load; overcome latency; encapsulate protocols; execute asynchronously and autonomously; provide aggregate functionality of several individual protocols [40, 41]. Some of the applications of the agent technology are E-commerce, information management, mobile computing, telecommunication services management, network management, parallel processing and work-flows.

2.2 Problems addressed by PET-ASMAC and its placement

PET-ASMAC addresses the following crucial problems required for efficient and smooth multimedia communications in an integrated manner.

- *QoS routing*: It is very much essential to route the stream packets as per the bandwidth and delay constraints at the beginning of the session as well as during run-time of the session. A properly QoS-routed packets of a stream will maintain the throughput and bounded delay characteristics of an application.
- *Resource negotiation/re-negotiation*: It is very important to negotiate/re-negotiate and reserve the resources such as bandwidth, processing delays and so on at the intermediate nodes and the end-hosts to uphold the continuous and smooth reception of packets of an application stream. Re-negotiation is required when one or more nodes in the path violates bandwidth and delay requirements during the session.
- *Synchronisation*: It is essential to maintain continuous and smooth presentations at the client side by maintaining the required end-to-end delays and bounded jitters. This will be achieved by interacting with network nodes, server and client and also by using different types of synchronisation schemes based on network/user needs and playout compensation mechanisms (for instance, freezing the frame if next frame is not available, do not play anything for audio if samples are not available).

PET-ASMAC is placed at the client side (Fig. 1). It interacts with the application program, network nodes (intermediate nodes) and the end-host (server) to cater the required services by preserving the QoS guarantees of an application.

2.3 Objectives of PET-ASMAC

The PET-ASMAC objectives are to assist a multimedia application to meet its QoS requirements by performing the following operations.

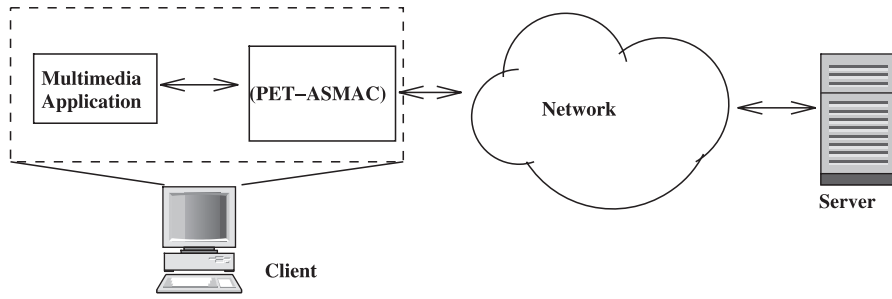


Fig. 1 PET-ASMAC interactions

- To find the paths which satisfy the QoS (bandwidth, end-to-end delay and sustainable losses) requirements of an application.
- To negotiate/re-negotiate the resources for running an application within the bounded delays in case of congestion/failure or QoS violations.
- To monitor the data rate, delays, rate of change of delays and losses at the nodes in the path of an application.
- To compute the playout timings and suggest a suitable playout compensation mechanism based on the availability of the presentation units.
- To change the transmission and playout rates to adjust to the cropped-up delays.
- To deploy different types of synchronisations (point, real-time or adaptive) as and when required [31, 32].

These synchronisation schemes can be used for both point-to-point and multipoint-to-point communications. The point synchronisation provides the point-to-point binding among the intra-presentation units (PUs) of streams of an application. Point synchronisation is normally referred for inter-stream synchronisation, for example, slide synchronisation. Here, we have used this name for intra-stream synchronisation for the following reason. A current frame can be played only when beginning bits of the next frame is received within certain duration (i.e. multiples of frame duration) as specified by an application. If a next frame is not received within the given duration, it will be rejected even if it arrives at the client. During the waiting period for next frame, previous frame will be played. Also, once the waiting duration is completed and the next frame did not arrive, current frame will be played.

In real-time continuous synchronisation, PUs of a stream (or streams) are synchronised with the real-time axis, for example, the motion video with contents of 20 s should be presented for exactly 20 s. Adaptive synchronisation can be used both for inter-stream and intra-stream [27], where the presentation of PUs of stream(s) adapt to network delay variations and the availability of network resources. Playout times of the PUs of a stream are re-adjusted to cope up with the network problems to maintain the clarity of presentation by PET-ASMAC in case of adaptive synchronisation.

The PET-ASMAC employs agents to achieve the specified objectives in an integrated manner, as agent technology allows programmable and flexible services. The benefits of using agents are flexibility, adaptability, software maintainability, software reuse, customisation, scalable service coding embedded within agents and dynamic multimedia communications software architecture creation.

2.4 Components of PET-ASMAC

The PET-ASMAC components are depicted in Fig. 2. It comprises a blackboard and a set of static and mobile

agents. The agents in PET-ASMAC communicate and coordinate with each other using the blackboard architecture principle. Now we describe all the components of PET-ASMAC.

2.4.1 Blackboard: The blackboard is a shared knowledge base structure, which is read and updated by the agents as and when required. A typical entry of an application contains following data: server address, application requirements (bandwidth, delay, rate of change of delay, sustainable loss), presentation unit (PU) generation period, monitored QoS parameters, established QoS path, negotiated/re-negotiated QoS parameters, number of PUs of a stream, playout start time and playout time of PUs (Fig. 3).

2.4.2 Communication manager agent: This is a static agent initiated at client by multimedia application with its QoS requirements. All other agents (delay estimator and QoS negotiator/re-negotiator agents) in the PET-ASMAC are triggered by this agent to perform dedicated tasks. It finds QoS route, negotiates and re-negotiates the QoS requirements of an application with the network at the beginning, and also as and when required with the help of QoS negotiator/re-negotiator mobile agent. Later it advises the application accordingly to plan the current/future playouts. It monitors the QoS parameters of an application stream at the intermediate nodes by employing delay estimator agent as well as by observing the packet arrivals [7]. It uses one of the synchronisation (either point or real-time or adaptive) mechanisms to compute the playout timings based on the application synchronisation requirements and the network condition. The agent adjusts playout rates whenever delays vary.

In case of faster packet arrivals (may be due to light network load conditions), that is, low network jitters, packets are buffered if there are enough buffers reserved for the session, otherwise, playout rate is increased by randomly skipping certain frames. Also, the subsystem informs the server to decrease the transmission rate (by using scalable coding). Whereas in case of slower packet arrivals

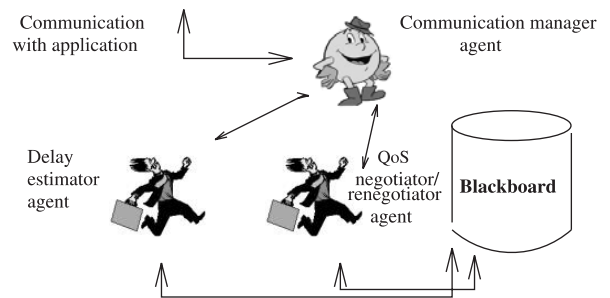


Fig. 2 Components of PET-ASMAC

Entry for an application from source to destination

server details	QoS required	monitored QoS	QoS path (c...-s)	negotiated/ renegotiated QoS	playout times of PUs
113.16.64.90:1000	0.4Mb, 200 to 400 ms, 5%	0.35Mb, 250ms, 10us/PU, 1%	8-2-10-1-45-16-20-...-56	0.39Mb, 240ms, 11us/PU	

PU numb.	playout time
1	12:45:60:20
2	12:45:60:40
...	...

Fig. 3 Blackboard entry of an application

(may be due to network congestion), that is, network jitters exceeding the jitter requirements, playout rate is decreased by freezing the frames for more than a frame playout duration.

The packet arrival timings observed by the agent are used to compute the bandwidth, end-to-end delays and the packet losses (Algorithm 1). Time-stamping may increase the communication overheads to some extent, but at the same time, it helps in providing relevant information for synchronisation of PUs of a stream. A stream is coded with scalable codecs at various levels ranging from base level to excellent level of required services either based on network congestion or QoS that an application requires.

Algorithm 1: Monitoring

Function: The communication manager agent monitors the QoS of an application in the local host. {Nomenclature: y , window size in terms of PUs; $plost$, PUs lost because of late arrivals; $pnloss$, PUs lost in network; b , bandwidth; D , end-to-end delay; d/dy , rate of change of delays with respect to dy ; $loss$, PU loss ratio; $puzise$, size of the PU in bytes; at_k , time at which k th PU arrives at the client; st_k , time at which k th PU is sent from the server.}

Begin

1. Observe the sending and arrival times of PUs within the window;
2. Monitor the number of PUs arrived late (i.e. after its scheduled display time) and the PUs did not arrive;
3. Compute the bandwidth, delay, rate of change of delays and packet loss as follows:
 - $b = (\sum_{k=1}^{y-pnloss} puzise (at_{k+1} - at_k)) / (y - pnloss)$;
 - $D = (\sum_{k=1}^{y-pnloss} (at_k - st_k)) / (y - pnloss)$;
 - $d/dy = (\sum_{k=1}^{y-pnloss} (at_{k+1} - at_k) - (at_{k+2} - at_{k+1})) / (y - pnloss)$;
 - $loss = (plost + pnloss) / y$;
4. Update the blackboard with the computed parameters;
5. Repeat steps 1–4 until the session is completed;
6. Stop.

End.

Time line synchronisation is used for all kinds of synchronisation specified in this paper at the beginning of the session as given by an application. But whenever an event occurs based on network load, the synchronisation requirements may be changed dynamically by the subsystem, that is, application may not be given its desired synchronisation for playout, instead network load may dictate the type of synchronisation to be used.

Algorithm 2: Finding synchronisation requirement

Function: Determines the synchronisation technique to be used for the playout time computations, which is one of the functions of communication manager agent. {Nomenclature: $ptsync$, point synchronisation; $rtsync$,

real-time synchronisation; $adsync$, adaptive synchronisation; $appstype$, application synchronisation requirement (either point, real-time or adaptive); $netstat$, network status (decided based on the number of QoS violations within an interval.)

Begin

1. Read the synchronisation parameters: sustainable loss, playout duration, waiting duration, playout rate, bounded delays and bounded jitters;
2. Read the type of synchronisation required $appstype$ ($ptsync$, $rtsync$, $adsync$);
3. Apply the requested synchronisation technique for playout at the beginning;
4. While session is running
 - If ($netstat$ is congested/failed) then use $adsync$ irrespective of $appstype$;
 - If ($netstat$ is lightly loaded) then use $rtsync$ irrespective of $appstype$;
 - If ($netstat$ is moderately loaded) then use $appstype$;
 - Repeat substeps in step 4 as and when the network status or application synchronisation requirement changes;
5. Stop.

End.

The agent uses adaptive synchronisation when the network is either congested or failed irrespective of given application synchronisation requirements, real-time synchronisation when the network is lightly loaded irrespective of application synchronisation requirements (see Algorithm 2). Some synchronisation parameters are to be defined for adaptive synchronisation by an application. The parameters used are sustainable loss, playout duration, bounded delays and bounded jitters. For details of adaptive synchronisation by using agents refer [32, 42]. Algorithm 3 describes the functioning of communication manager agent.

Algorithm 3: Functions of communication manager agent

Function: To maintain QoS requirements of an application by performing dynamic negotiation/re-negotiation, playout time computations and monitoring mechanisms.

{Nomenclature: $bwmon$, monitored bandwidth; $bwreq$, required bandwidth; $dmon$, monitored delay; dms , maximum sustainable delay; rdm , monitored rate of change of delay; rds , sustainable rate of change of delay; plm , monitored packet loss; pls , sustainable packet loss.}

Begin

1. Receive the QoS and synchronisation requirements of a stream from an application;
2. Trigger a QoS negotiator/re-negotiator mobile agent to negotiate the resources to find a QoS route; (see Algorithm 4a in Section 2.4.3)
3. Trigger delay estimator mobile agent; (see Algorithm 5a in Section 2.4.4)
4. Estimate the server transmission starting time (using clock difference and delay information) in consultation with server and application user;
5. Monitor the QoS of a stream (CALL Algorithm 1);
6. Find the synchronisation requirements of an application (CALL Algorithm 2);
7. Compute the new playout timings for the presentation of multimedia data units with new negotiated parameters as per the synchronisation requirements; adjust the playout timings if necessary during playout;

8. Inform presentation timings to the server, network and application;
9. Trigger delay estimator mobile agent periodically to observe the QoS of an application in the network; (see Algorithm 5b in Section 2.4.4)
- 10./* check for QoS requirement violation and network problems*/

If $((bwmon < bwreq) \text{ or } (dmon > 0.9 \times dms) \text{ or } (rdm > rds) \text{ or } (plm > 0.9 \times pls) \text{ or feedback from the network})$ then, trigger the QoS negotiator/re-negotiator agent to re-negotiate the resources with the nodes in the established path; (see Algorithm 4b in Section 2.4.3)

11. Repeat steps 6–10 until the session is completed;
12. Dispose itself and other created agents;
13. Stop.

End.

2.4.3 QoS negotiator/re-negotiator agent: PET-ASMAC uses on-demand QoS routing. Thus, a PU will have path information with it that will be used for forwarding from node to node. QoS negotiator/re-negotiator agent is a mobile agent, which is used to find the QoS route (a route satisfying bandwidth, delay and loss requirements of an application) from the server to the client at the beginning of a session as well as whenever required. It negotiates/re-negotiates the resources in the path. It follows the principle of Internet routing protocols while establishing a route from the client to the server by collecting the neighbour connectivity and resource information (bandwidth availability, link delays, node queuing delays, rate of change of delays and packet loss probabilities). The agent finds a set of multiple paths between the server and the client and finally chooses a shortest-widest (minimum hop, with higher bandwidth) path among them [20, 25] for resource reservation. Algorithm 4a describes the functioning of negotiation phase of the agent.

Algorithm 4a: Negotiation phase

Function: To find a QoS route at the beginning (QoS parameters considered are bandwidth and delay).

Begin

1. The QoS negotiator/re-negotiator agent collects the QoS requirements from the relevant communication manager agent;
2. The agent migrates from client to its neighbours, and their neighbours and so on until it reaches the server. While traversing it collects the resource availability information from each of the visited nodes and their neighbours;
3. When the agent reaches the server, it finds a set of multiple QoS paths that satisfies the required resources
 - Prune all the edges/links in collected connectivity/resource information that have less than the desired bandwidth;
 - Find K (we take $K = 2$) paths: find shortest path $P1$ using Dijkstra's algorithm, prune the link with lowest bandwidth in path $P1$, find second path $P2$ using Dijkstra's algorithm;
 - Check for eligibility of K paths for delay requirement satisfaction, that is, end-to-end delay of the paths should be less than the desired end-to-end delay
4. If (QoS path(s) available) then, select a best QoS path (path with widest bandwidth and lowest delays) and reserve the resources on the path, and inform the server and communication manager agent;

Else, inform the communication manager agent that QoS path is not available;

5. Dispose the QoS negotiator/re-negotiator agent;
6. Stop.

End.

We choose to find only two paths as it has been observed that finding more than two paths may not show significant improvement in performance [43]. Finding QoS paths does not strictly consider the disjoint paths. It finds paths that may overlap, that is, some of the links may be common in both the paths. After a feasible path is chosen among the multiple paths, agent reserves the resources on the selected path and informs the manager agent. In case of QoS violations, it goes on a mission to re-negotiate the resources along the path. The subsystem is proactive, that is, when QoS violations are nearing some threshold value, QoS negotiator/re-negotiator agent is triggered by communication manager agent to re-negotiate the resources on existing path as well as collect the resource and connectivity information along the path. If it fails in re-negotiation, it finds another QoS path using the collected connectivity information and re-negotiates the resources and informs to communication manager agent. Algorithm 4b describes the re-negotiation phase of the agent.

Algorithm 4b: Re-negotiation phase

Function: To re-negotiate resources whenever a QoS violation or congestion/failure is detected during a session.

Begin

1. The QoS negotiator/re-negotiator agent collects the QoS requirements to be re-negotiated from the communication manager agent;
2. It migrates on the specified path by visiting every node on the path and also their immediate neighbours;
3. After reaching the server, it checks whether the re-negotiation is successful at all the visited nodes;
4. If re-negotiation is successful then inform the newly negotiated QoS values on the existing path to the server and communication manager agent and goto step 6;
- Else find the multiple QoS paths that satisfies required resources (as given in step 3 of Algorithm 4a);
5. If (QoS path(s) available) then, select a best QoS path and reserve the resources on the path and inform the server and manager agent;
- Else, inform the communication manager agent that QoS path is not available;
6. Dispose the QoS negotiator/re-negotiator agent;
7. Stop.

End.

2.4.4 Delay estimator agent: It is a mobile agent responsible for detecting the timing differences between the client and server clocks [31] at the beginning of the session so as to synchronise the PUs of stream for playout. After a QoS route is discovered, agent makes several round trips to server and estimates the clock difference with respect to client. This information is used to compute the playout start time at the client and the server transmission start time. It also estimates initial delays and rate of change of delays while moving through the path and negotiates the delay parameters at each

intermediate node such that end-to-end delay is within the desired end-to-end delay (see Algorithm 5a).

Algorithm 5a: Collection of client and server clock timings

Function: Calculates the clock difference between the client and server. {Nomenclature: T_1, T_2, T_3 are the timings at which agent begins onward journey to server, reaches server and returns to client, respectively; δ , error in estimation ($\simeq 100$ ms); Δ^c , clock difference between client and server.}

Begin

1. Delay estimator agent records the timings T_1, T_2 , and T_3 while traversing from client to server;
2. Compute Δ^c as given below:

$$\Delta^c = \frac{(T_3 - T_1)}{2} - (T_2 - T_1) + \delta \quad (1)$$

3. Agent estimates the delays (propagation and queuing delays) and rate of change of delays at each one of the nodes of the specified path;
4. Inform the clock difference, delays and rate of change of delays to the server and communication manager agent;
5. Dispose the agent;
6. Stop.

End.

This agent also observes the delays and rate of change of delays periodically at each node of the specified path and informs the server and manager in case of delay parameter violations (Algorithm 5b).

Algorithm 5b: Periodic visits of delay estimator agent

Function: To observe the delays and rate of change of delays.

Begin

1. Observe the delays and rate of change of delays of a stream at each node of the specified path by traversing from client to server;
2. In the event of any QoS violations, the agent informs both the server and communication manager agent;
3. Dispose the agent;
4. Stop.

End.

2.5 Extensions to PET-ASMAC

The subsystem presented earlier does work for point-to-point communication and performs intra-stream synchronisation. However, PET-ASMAC can be easily extended to handle multiple servers (multipoint-to-point communication) and perform inter-stream synchronisation by making some modifications in the agents. In case of multiple servers, communication manager agent generates multiple QoS negotiator/re-negotiator and delay estimator agents that traverse to their respective servers to find the QoS paths and negotiate/re-negotiate the resources as well estimate the delays. To compute the server(s) transmission time, initial playout time for each stream and playout timings of PUs for each stream, communication manager agent computes another synchronisation parameter called as skew (inter-arrival difference between the PUs of streams). Discussion of inter-stream synchronisation is beyond the scope of this paper. The details of agent-based inter-stream synchronisation and its analysis can be found in [42].

3 Simulation

We have simulated the proposed PET-ASMAC and tested in several network topologies for the performance and effectiveness of the approach with respect to changes in the network resource parameters and link failures. The topologies are generated using the GT-ITM package that uses pure random graph method [44]. In this section, we discuss the network model used to test PET-ASMAC, simulation procedure and the results.

3.1 Network model

The model considers a physical topology upto 100 nodes and predefined routes are generated using the principle of the Internet routing protocol. Background load (BG) is generated on each link with certain percentage of the maximum capacity (C) of the link. The propagation delay of each link is measured as pd. The number of applications admitted at each node will be a maximum of N . λ is the actual number of QoS requests arrival rate at each node, that is, QoS request generated are $\lambda \times N$. The application request arrivals are Poisson-distributed with mean inter-arrival time ξ .

The QoS requirements bandwidth, delay and loss of an application are uniformly distributed within the ranges $[b_1, b_2]$, $[d_1, d_2]$ and $[l_1, l_2]$, respectively. Maximum sustainable delays of an application varies from application to application characteristics. Client-server pairs are chosen randomly. The rate at which the PUs are generated and displayed are the same. The rate of change of delay at each network node has been simulated to increase between 0 and 100%. The bandwidth usage at each node by an application randomly increases or decreases by a certain fraction x of the allocated bandwidth.

We also model the probability that resource parameters sensed by the agents changes by the time agent moves on the path (Pc) for resource reservation. The percentage changes in the BG load to create the changes in the resources is uniformly distributed between the range $[0, lc]$. Random link failures (fl) are modelled to test for the adaptation of PET-ASMAC with variation in network topology.

We consider packets lost because of late arrivals, link failures and packets held up at intermediate nodes because of congestion for a period larger than bounded jitters. The number of packets delivered on a failed link are assumed to be lost. Intermediate nodes are assumed to have enough buffer space to hold the multimedia flow, as such flows have been negotiated for their services. In case of congestion at an intermediate node, multimedia packet will not be lost but gets delayed because of lengthy queues. An intermediate node may reject a packet if it is residing in the node for a duration beyond the jitter requirements of an application.

3.2 Simulation procedure

The inputs considered in the simulation are BG is varied from 0 to 70% of network capacity; C , 50 Mbps; pd, 10 ms; N , 25; $\xi = 0.5$; b_1 , 3 Mbps; b_2 , 5 Mbps for CBR applications; b_1 , 2 Mbps and b_2 , 3.5 Mbps with source utilisation = 0.8 (80% on period) for VBR applications; d_1 , 150 ms, d_2 , 400 ms; application sustainable delays may vary upto 600 ms; l_1 , 0.01, l_2 , 0.1; PU generation period = 50 ms; x , random value between 0.1 and 1.0; PU size, 1024 bytes; Pc and lc are varied from 0 to 0.5.

Begin

1. Generate a network topology with random size of nodes and links.
2. Generate the background traffic on each of the links in the network.
3. Randomly select a client–server pair of an application.
4. Run PET-ASMAC at the client.
5. Generate traffic between the client and server which is monitored by the PET-ASMAC.
6. Compute performance of the PET-ASMAC.

End.

The performance parameters measured are as follows:

- *Percentage of PET-ASMAs one at each client successful in QoS routing*: It is defined as the ratio of number of PET-ASMAs successful in getting a QoS route to the number of PET-ASMAs requested for QoS routes.
- *Percentage of network bandwidth utilisation*: It is defined as the ratio of the sum of the bandwidth utilised (restricted to maximum capacity) in all the links (of network) to total network bandwidth (sum of maximum capacity of all the links in the network).
- *End-to-end delay variation*: It denotes the delay variations of PUs of an application.
- *Bandwidth variation*: It denotes the variations in bandwidth offered to an application.
- *Percentage of PET-ASMAs failed in re-negotiation*: It is defined as the ratio of PET-ASMAs failed in re-negotiations to the number of PET-ASMAs attempted for re-negotiations.

3.3 Results

The paper addressed many issues and it is difficult to provide detailed schemes, analysis and results given the restriction on the number of pages. However, detailed results for each kind of issue handled by PET-ASMAs can be found in [10, 25, 32, 42, 45]. Here, we present the results of the performance parameters considered for PET-ASMAs using the network model.

We measured the performance of PET-ASMAs at a client with the different background loads 0, 40 and 70% of network capacity. It is noticed that the network bandwidth utilisation increases with increase in application arrival rate for different background loads (Fig. 4). The PET-ASMAs's success in finding QoS routes decreases with increase in application arrival rate (Fig. 5). The success ratio is better in the case of lower background loads.

To illustrate the bandwidth and delay variations we observed one application which ran for 90 min duration. Fig. 6 illustrates the delay variation of application, whereas Fig. 7 demonstrates the bandwidth variation during the presentation of application data. We observe that most of the delays and bandwidth of the applications are well within the bounded values. In some cases of QoS violations, PET-ASMAs tries to re-negotiate and schedule the applications. If PET-ASMAs still fails to re-negotiate, the application's will be rejected without completing the session (Fig. 8).

It is also noticed that routing acceptance decreases with the increase in probability of resource parameters sensed by the agent gets changed (P_c) by the time the agent moves for reservation (Fig. 9). The acceptance of PET-ASMAs for QoS routing also reduces with increase in variation of BG load (l_c).

The failed re-negotiations increase with rise in P_c and l_c (Fig. 10). Fig. 11 depicts the impact on failed re-

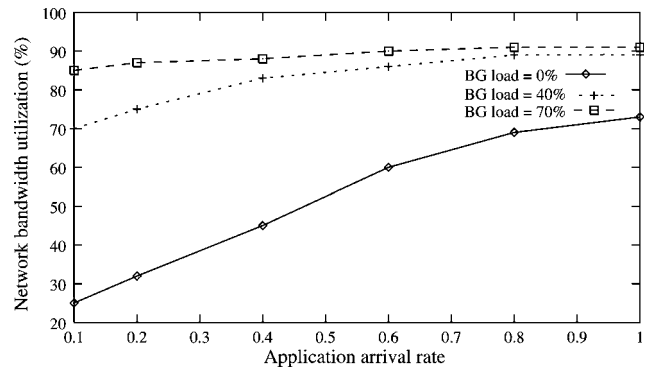


Fig. 4 Network bandwidth utilisation against application arrival rate

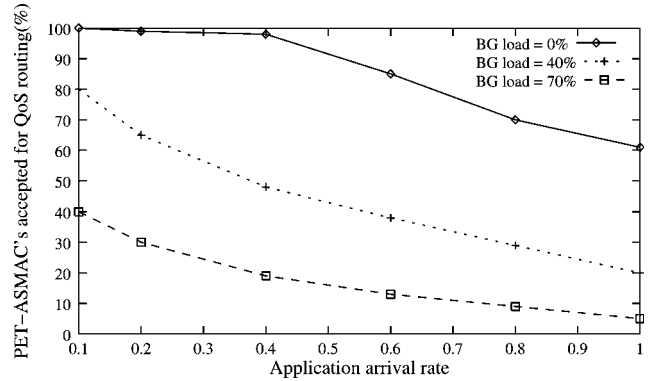


Fig. 5 PET-ASMAs successful in routing against application arrival rate

negotiations with respect to link failures. We observe that as link failures increase, re-negotiation failure percentage increases by a certain factor.

We provide some results that compares PET-ASMAs routing acceptance and network bandwidth utilisation with the traditional system that uses RSVP and (RIP) for QoS routing. Fig. 12 depicts the increase in QoS routing acceptance in PET-ASMAs in comparison with traditional systems. It decreases with increase in background load. This is because of the fact that PET-ASMAs uses multipath computation and selects feasible path for routing. Network bandwidth utilisation is improved in PET-ASMAs as compared with traditional system shown in Fig. 13. It is

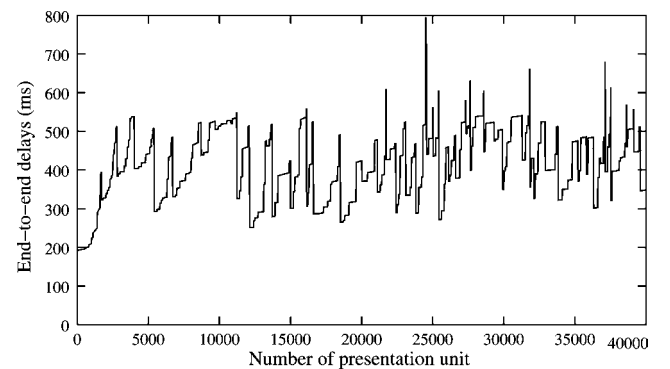


Fig. 6 Delay variations against number of PUs

We give here only the first 40 000 PUs for showing clarity in delay variations

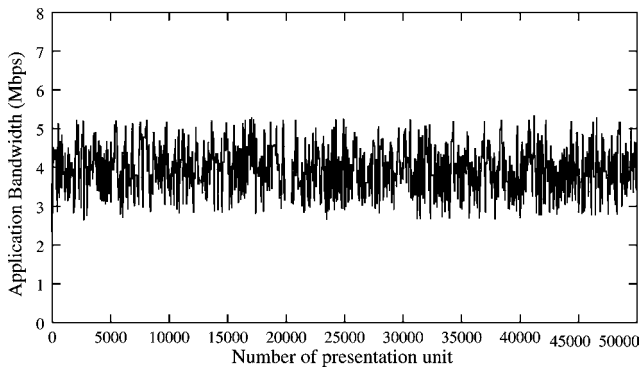


Fig. 7 Bandwidth variations (Mbps) against number of PUs

Bandwidth variations for first 50 000 PUs taken at regular intervals of ten PUs

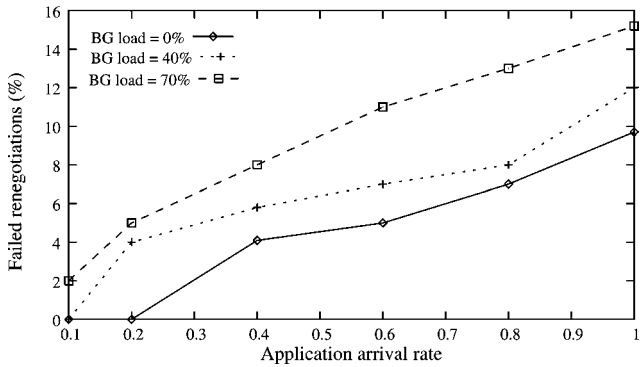


Fig. 8 PET-ASMACs failed in re-negotiation against application arrival rate

observed that bandwidth utilisation reaches a saturation level with increase in application arrival rate. It increases with increase in background load.

4 Benefits of using agent technology

We have observed that PET-ASMAC offers flexibility, scalability, efficiency, adaptability, software reusability and maintainability in the experiments conducted. Eventhough it is difficult to quantify these features, we explain below how they are achieved with PET-ASMAC.

- **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 mobile agent in PET-ASMAC can be encoded with some intelligence to negotiate the resources along the route depending on user requirements. Flexibility can also be seen in providing the mobile agent code facilitation for personalising the services of the users. For example, the PET-ASMAC can make provision to take into account the aggregate connections from a client to compute a QoS path or to negotiate the network resources in an optimal way.
- **Re-usability:** Part of the PET-ASMAC software can be reused in applications like on-demand resource allocation, dynamic playouts by making slight modifications to the software. It is possible because of autonomous operation of all the agents used in the PET-ASMAC. For example, an application may reuse the QoS negotiator/re-negotiator

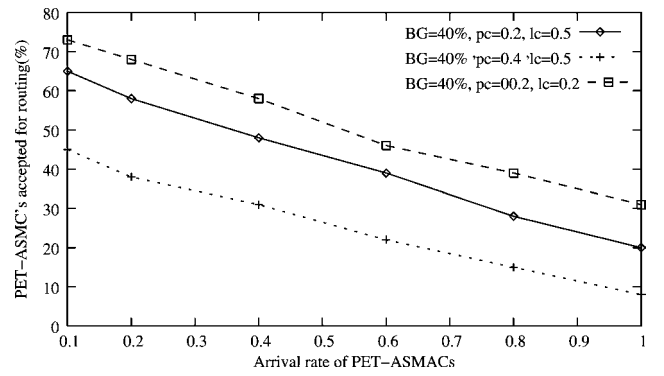


Fig. 9 PET-ASMACs accepted for routing against application arrival rate by considering changes in pc and lc with BG = 40%

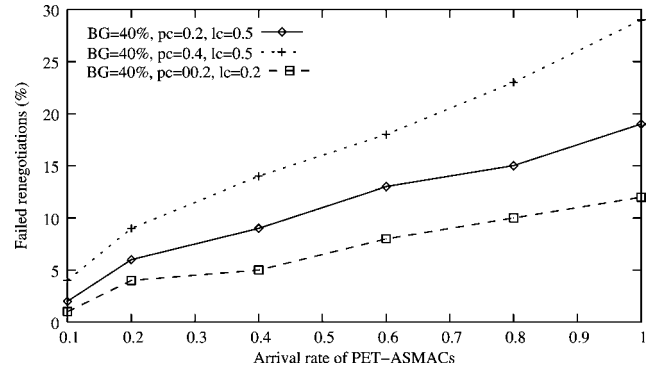


Fig. 10 PET-ASMACs failed in re-negotiation against application arrival rate by considering changes in pc and lc with BG = 40%

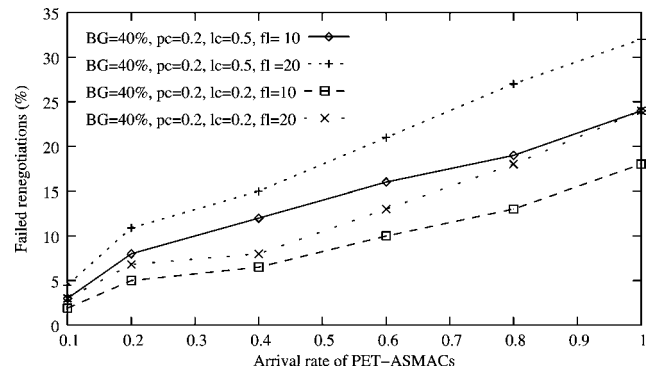


Fig. 11 PET-ASMACs failed in re-negotiation against application arrival rate by considering 10 and 20 random link failures (fl) along with changes in pc and lc

mobile agent to collect the second/third degree neighbourhood information by slightly modifying the code.

- **Maintainability:** The software in PET-ASMAC will be easily maintained, as every agent of the subsystem is developed on a modular approach. Debugging and updating the communication software in PET-ASMAC can be done with ease.
- **Adaptability:** PET-ASMAC easily adapts to rapid changes in the network conditions and an application requirement. For example, in case of QoS violations and path failure, PET-ASMAC immediately finds another QoS route and reserves the resources.
- **Encapsulation of protocol:** A mobile agent of the PET-ASMAC encapsulates customised protocols if required

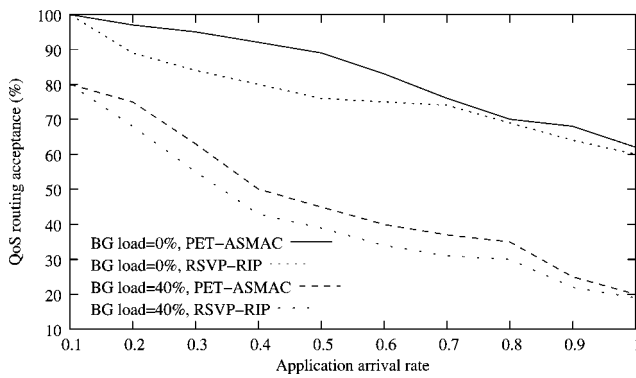


Fig. 12 QoS routing acceptance (%) against application arrival rate (RSVP-RIP against PET-ASM)

and thus overcomes the need of lengthy standardisation process of the protocols.

- **Customisation:** The services for multimedia communications can be customised by encoding the agents of the PET-ASM according to the client's requirements and can be downloaded on-demand from the client's web site whenever the client wishes to run a multimedia application.
- **Efficiency:** The use of PET-ASM increases network resource utilisation efficiency because of its adaptability to the network requirements and exchange of minimum information during the task execution and decision-making with multiple resource information.
- **Scalability:** The scalability can be achieved using the PET-ASM mobile agents at a client to perform several user's similar tasks by a minimum set of mobile agents by mutually cooperating among themselves.

Agent-oriented programming facilitates CBSE, which is needed in today's software development especially in web-based systems and Internet protocols [34, 35]. In future there will be enormous number of agents (agents are the next generation components) which have to coordinate with each other to provide better multimedia information searching, retrieval, wireless/wired communication services and market places [41].

5 Conclusions

The paper proposed PET-ASM, an intelligent multimedia communication support system to ensure continuous and smooth playout of a multimedia application. It uses a set of static and mobile agents to perform QoS routing, bandwidth allocation, stream synchronisation and playout time computations. We simulated the PET-ASM in

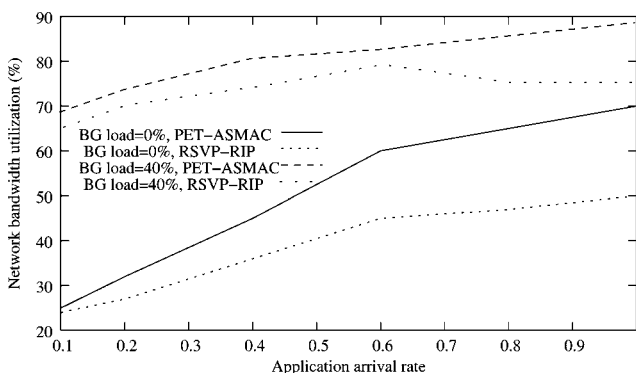


Fig. 13 Network bandwidth utilisation (%) against application arrival rate (RSVP-RIP against PET-ASM)

several network scenarios to evaluate its performance and effectiveness. We observe that the proposed multimedia communications assistant helps the application to operate within its sustainable QoS parameters with high probability. The benefits of using PET-ASM are flexibility, adaptability, scalability, efficient resource allocation, customisation, re-usability and maintainability. The PET-ASM may be extended to provide security and multicast services.

6 Acknowledgments

We acknowledge the anonymous reviewers for their valuable suggestions, which helped us to improve the quality of the paper.

7 References

- 1 Furini, M., and Towsley, D.F.: 'Real time traffic transmissions over Internet', *IEEE Trans. Multimed.*, 2001, **3**, pp. 33–40
- 2 Aurrecoechea, C., Campbell, A.T., and Hauw, L.: 'A survey of QoS architectures', *Multimedia Syst.*, 1998, **2**, (6), pp. 138–151
- 3 Manvi, S.S., and Venkataram, P.: 'A review of bandwidth allocation schemes for multimedia communications'. IETE Golden Jubilee Compendium: Evolution perspective – electronics, telecommunications, IT and broadcasting, October 2003, pp. 51–64
- 4 Lu, G.: 'Issues and technologies for supporting multimedia communications over the Internet', *Comput. Commun.*, 2000, **23**, pp. 1323–1335
- 5 Wipusitwarakun, K., Tode, H., and Ikeda, H.: 'VP's priority based restoring function enhanced self-healing algorithm', *IEICE Trans. Commun.*, 1998, **E81-B**, (11), pp. 2100–2109
- 6 Sariot Ismail, I.A.: 'Bandwidth problems in high speed networks', *IBM J. Res. Dev.*, 2000, **44**, (6), pp. 919–938
- 7 Manvi, S.S., and Venkataram, P.: 'QoS management by mobile agents in multimedia communication'. Proc. Database and Expert Systems (DEXA 2000), Agent-Based Intelligent Systems ABIS-2000, Greenwich, UK, September 2000, pp. 407–411
- 8 Manvi, S.S., and Venkataram, P.: 'Mobile agent based online bandwidth allocation scheme in multimedia communications'. Proc. IEEE Globecom, San Antonio, USA, November 2001, pp. 2622–2626
- 9 Manvi, S.S., and Venkataram, P.: 'Adaptive bandwidth reservation for multimedia traffic using mobile agents'. Proc. IEEE HSNMC (High Speed Networking and Multimedia Communications), Jeju, Korea, July 2002, pp. 370–374
- 10 Manvi, S.S., and Venkataram, P.: 'An agent-based adaptive bandwidth allocation scheme for multimedia applications', *J. Syst. Softw.*, 2005, **75**, pp. 305–318
- 11 Guedes, L.A., Olivera, P.C., Paina, L.F., and Cardozo, E.: 'An agent based approach for supporting quality of services in distributed multimedia systems', *Comput. Commun.*, 1998, **21**, pp. 1269–1278
- 12 De Meer, H., La Corte, A., Pualiafito, A., and Tomarchio, O.: 'Programmable agents for flexible QoS management in IP networks', *IEEE J. Sel. Areas Commun.*, 2000, **18**, (2), pp. 256–267
- 13 Ford, B., and Karmouch, A.: 'An architectural model for mobile agent-based multimedia applications'. Proc. Canadian Conference on Broadband Research, Ottawa, Canada, 1997
- 14 Chen, S., and Nahrstedt, K.: 'An overview of quality of service routing for next-generation high speed networks: problems and solutions', *IEEE Netw.*, 1998, pp. 64–79
- 15 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, pp. 42–54
- 16 Ghosh, D., Sarangan, V., and Acharya, R.: 'Quality-of-service routing in IP networks', *IEEE Trans. Multimed.*, 2001, **3**, pp. 200–208
- 17 Lellis Vieira, S.: 'Efficient routing with quality-of-service requirements'. Proc. IEEE Symp. on Computer Communications (ISCC), Athens, Greece, 2001, pp. 326–331
- 18 Weiss, W.: 'QoS with differentiated services', *Bell Labs Tech. J.*, 1998, pp. 48–62
- 19 Guerin, R., Orda, A., and Williams, D.: 'QoS routing mechanisms and OSPF extensions'. Proc. IEEE Globecom, Phoenix, USA, 1997, pp. 1903–1908
- 20 Ma, Q., and Steenkiste, P.: 'On path selection for traffic with bandwidth guarantees'. Proc. IEEE Int. Conf. on Network Protocols (ICNP), Atlanta, USA, 1997, pp. 345–362
- 21 Di Caro, G., and Dorigo, M.: 'Two ant colony algorithms for best-effort routing in datagram networks'. Proc. 10th IASTED Int. Conf. on Parallel and Distributed Computing and Systems (PDCS'98), Chicago, USA, 1998, pp. 541–546

- 22 Reyes, A., Sanchez, E., and Barba, A.: 'Routing management application based on mobile agents on Internet 2'. EUNICE Open European Summer School, Eschede, The Netherlands, 2000, <http://www.tgs.cs.utwente.nl/Docs/eunice/summerschool/papers/paper9-2.pdf>
- 23 Oida, K., and Sekido, M.: 'ARS: an efficient agent based routing system for QoS guarantees', *Comput. Commun.*, 2000, **23**, pp. 1437–1447
- 24 Gonzalez-Valenzuela, S., and Leung, V.C.M.: 'QoS routing for MPLS networks employing mobile agents', *IEEE Netw.*, 2002, **16**, pp. 16–21
- 25 Singh, A., Manvi, S.S., and Venkataram, P.: 'QoS routing scheme by using mobile agents'. Proc. Int. Conf. on Artificial Intelligence (IICAI), Hyderabad, India, December 2003, pp. 310–324
- 26 Blakowski, G., and Steinmetz, R.: 'Media synchronisation survey: reference model, specification, and case studies', *IEEE J. Sel. Areas Commun.*, 1996, **14**, (1), pp. 5–35
- 27 Laoutaris, N., and Stavrakakis, I.: 'Intra-stream synchronization for continuous media streams: a survey of playout schedulers', *IEEE Netw.*, 2002, **16**, (3), pp. 30–40
- 28 Zhang, A., Yuqing, and Mielke, M.: 'Netmedia: streaming multimedia presentations in distributed environments', *IEEE Multimedia*, 2002, **9**, (1), pp. 56–73
- 29 Rothmel, K., and Helbig, T.: 'An adaptive protocol for synchronizing media streams', *ACM Multimedia Syst.*, 1997, **5**, pp. 324–336
- 30 Shivkumar, N., Sreenan, C.J., Narendra, B., and Agrawal, P.: 'The concord algorithm for synchronisation of networked multimedia streams'. Proc. Int. Conf. on Multimedia Computing Systems (ICMCS), Washington, DC, USA, 1995, pp. 31–41
- 31 Manvi, S.S., and Venkataram, P.: 'A multimedia synchronisation model for Internet-based education-systems using agents', *J. Comput. Sci. Inf.*, 2003, **33**, (2), pp. 30–41
- 32 Manvi, S.S., and Venkataram, P.: 'An agent based adaptive intra-stream synchronisation scheme for multimedia applications', *Int. J. Knowl.-Based Intell. Eng. Syst.*, 2005, **9**, pp. 1–15
- 33 Stiller, B., Class, C., and Waldvogel, M., *et al.*: 'A flexible middleware for multimedia communication: design, implementation, and experience', *IEEE J. Sel. Areas Commun.*, 1999, **17**, (9), pp. 1614–1631
- 34 Jennings, N.R.: 'An agent-based approach for building complex software systems', *Commun. ACM*, 2001, **44**, pp. 35–41
- 35 Griss, M.L., and Pour, G.: 'Accelerating development with agent components', *Computer*, 2001, **34**, (5), pp. 37–43
- 36 Manvi, S.S., and Venkataram, P.: 'Applications of agent technology in communications: a review', *Comput. Commun.*, 2004, **27**, pp. 1493–1508
- 37 Chess, D., Benjamin, G., Harrison, C., Levine, D., and Paris, C.: 'Itinerant agents in mobile computing', *IEEE Pers. Commun.*, 1995, **2**, (5), pp. 34–49
- 38 Wong, D., Paciorek, N., and Moore, D.: 'Java based mobile agents', *Commun. ACM*, 1999, **42**, pp. 38–45
- 39 Perdikeas, M.K., Chatzipapadopoulos, F.G., and Venieris, I.S.: 'An evaluation study of mobile agent technology: standardisation, implementation and evolution'. Proc. IEEE ICMCS, 1999, vol. 2, pp. 287–295
- 40 Chess, D., Harrison, C., and Kershenbaum, A.: 'Mobile agents: are they a good idea?'. IBM Res. Division, T.J. Watson Research Center, Yorktown Heights, New York, March 1995
- 41 Lange, D.B., and Oshima, M.: 'Seven good reasons for mobile agents', *Commun. ACM*, 1999, **42**, pp. 88–89
- 42 Manvi, S.S., and Venkataram, P.: 'Intelligent product-information in E-commerce', *Electron. Comm. Res. Appl.*, 2005, **4**, (3), pp. 220–239
- 43 Jia, Y., Nikoladis, I., and Gburzynski, P.: 'Multiple path routing'. Proc. IEEE Int. Conf. on Communications (ICC), Helsinki, Finland, 2001, pp. 2583–2587
- 44 Zegura, E.W., Calvert, K., and Bhattacharjee, S.: 'How to model an internet work'. Proc. IEEE Infocom, San Francisco, USA, 1996, pp. 1789–1795
- 45 Manvi, S.S.: 'Agent based schemes to resolve some of the issues in multimedia communications'. PhD thesis, Indian Institute of Science, India, 2004