Quality-of-Service Routing in IP Networks

Donna Ghosh, Venkatesh Sarangan, and Raj Acharya

Abstract—**Multimedia applications such as video-conferencing, telemedicine, HDTV, etc. have very stringent quality-of-service (QoS) demands and require a connection-oriented service. For these applications, a path satisfying their requirements in terms of bandwidth, delay, buffer, etc. needs to be found. As conventional IP routing is based only on hop counts, it is not suitable for multimedia applications. It is clear that, to route requests that have QoS requirements, existing routers should be made QoS aware and the packet forwarding should be based on QoS parameters. Also, routing protocols like OSPF and RIP must be extended suitably to facilitate QoS routing. The goal of QoS routing algorithms is to find a loop-less path satisfying a given set of constraints on parameters like bandwidth, delay, etc. The path selection process could return either the entire path to the destination or the best next hop for the request. The first case is called "source routing" and the second is referred to as "distributed routing." In this paper, we propose a new distributed QoS routing algorithm for unicast flows, which has a very low call establishment overhead. Our algorithm makes use of existing IP routing protocols such as OSPF and RIP with minimal modifications.**

Index Terms—**Distributed QoS routing, IP networks, partial global state information.**

I. Introduction

A. Need for QoS Routing

RESOURCE reservation is a necessity for providing guaranteed end-to-end performance for multimedia applications. However, in the present Internet setup, resource reservation is not supported. Also, the data packets of these applications could follow different paths and reach the destination out of order, which is not desirable. Hence, the future networks are likely to provide a connection-oriented service for real-time applications. These applications demand a guaranteed amount of network resources like bandwidth, buffer space, CPU time, etc. Hence, given a set of quality-of-service (QoS) requirements for a connection, the routers should be able to find a path which satisfies the requirements. The current routing protocols used in IP networks are transparent to any particular QoS that different flows could require. As a result, routing decisions are made without referring to the QoS requirements of the flow. This means that flows are often routed over paths that are unable to support their requirements while alternate paths with sufficient resources exist. This will increase the call blocking probability. The goal of QoS routing algorithms is to find a path in the network that satisfies the given requirements. They may also additionally optimize the global network resource utilization. Protocols like RSVP [14] have been proposed to enable the applications to request guaranteed amount of network resources in an internet. However, RSVP is not a routing protocol and it relies on the underlying routing table of IP. Hence, the application's ability to reserve resources will be a waste, if the path specified by the IP routing table cannot support the required resources. Thus, we clearly see the need for QoS routing algorithms in the current IP networks. However, the proposed QoS routing algorithms should demand only minimal changes to the existing routing protocols. This will greatly facilitate their deployment.

Depending on the scope of the path selection process, an algorithm could either return the best next hop or the entire path to the destination. The first case is more similar to the traditional hop-by-hop routing and is referred to as "distributed QoS routing." The latter is termed "source QoS routing." Henceforth, in this paper, the term "routing algorithms" will refer to QoS routing algorithms unless specified. Also, the terms "nodes" and "routers" have been used interchangeably in this paper. In source routing algorithms, the entire path computation is done at the source router. One of the main drawbacks of the source routing algorithms is that each router in the network is required to maintain a global network state information which needs to be updated periodically. "Global state" refers to the information regarding the entire network connectivity and resource availability in all the links. Protocols like OSPF can be extended to perform such updates [11]. This frequent updating generates a lot of overhead. The global state thus maintained is inherently imprecise due to the dynamic nature of the network resource availability. There is always a tradeoff between the average number of messages exchanged and the amount of staleness (or impreciseness) in the global state maintained at each router. Clearly, the amount of impreciseness and the average message overhead, both increase with the network size. Hence, such approaches are not scalable with network size. Also in source routing, finding a path could be computationally intensive for the source router. In distributed routing algorithms, the path computation is shared by various routers in the network. Hence, there is no computation burden on any single router in the network.

In this paper, we propose a new distributed packet forwarding mechanism based on the QoS requirements of the flow. We assume a network where all the routers are QoS aware, i.e., packets are forwarded based on both their destination and QoS requirements. We do not consider a heterogeneous network where some routers are QoS aware and some are not. We limit ourselves to the case of establishing an unicast flow. Each connection request contains the destination id, and the set of QoS requirements for that flow. The routing algorithm reads the destination and the QoS requirements, and returns a path (if available) that is most likely to satisfy the requirements.

Manuscript received March 6, 2001. The associate editor coordinating the review of this paper and approving it for publication was Dr. Sadaoki Furui.

The authors are with the Department of Computer Science and Engineering, State University of New York at Buffalo, Amherst, NY 14260 USA (e-mail: donnag@cse.buffalo.edu; sarangan@cse.buffalo.edu; acharya@cse.buffalo.edu).

Publisher Item Identifier S 1520-9210(01)04322-X.

B. Related Works

Various distributed routing algorithms have been proposed in [2]–[8]. Distributed algorithms can be categorized into two types based on whether or not all the routers maintain a global state. If the routers have a global state, it could be used in the path computation to specify the best next hop. Algorithms proposed in [3], [7], and [8] fall under this category. Hence, all of them suffer from the same problem as with source routing, namely, overhead in maintaining the global state and state impreciseness. Apart from degrading the routing algorithm's performance, the impreciseness can also create looping. If no global state is stored, techniques like flooding as in [2], [4], and [5] could be used to establish a path, where a request is flooded on all the router's outgoing links (excluding the incoming link) which satisfy the QoS requirements of the request. The problem with such an approach, however, is that the overhead involved in establishing a connection could be very high. Our approach is different from [2], [4], and [5] in the sense that we use additional state information and reduce the overhead in connection establishment. To reduce the overhead in flooding, along with the QoS constraints an additional constraint is imposed on the number of hops the connection request can travel [2], [6]. Our method complements this bounded flooding approach and can be used along with it. The combined usage results in a much lower overhead than either of them used alone. We propose a new distributed routing algorithm in which a router stores information only about its immediate neighbors (routers reachable in one hop) and second-degree neighbors (neighbors of a neighbor). The advantage of this approach is twofold. First, the message overhead and the impreciseness will not be as large as maintaining the global state. A router exchanges information only with its neighbors. As a result, the impreciseness in storing the information about the second-degree neighbors will not be as big as the impreciseness in storing the entire global state. Second, using the information about the second-degree neighbors, a router can forward the connection requests intelligently instead of blindly flooding the requests. This is because every router can now see two levels downstream. Hence, the overhead in connection establishment is reduced.

Cidon *et al.* [9] have used the idea of storing information about the second-degree neighbors. However, they use it only for rerouting (deflection routing). They have proposed deflection routing schemes for source routing networks and for ATM networks. In source routing, the source router chooses the path for a connection based on its global state and then sends a control packet to reserve resources along that path. It is quite possible that, a router along the path, on getting the control packet, finds that it does not have sufficient resources (along the link connecting it to the next hop). Then it would use its information about the second-degree neighbors to route the connection along some other link to the next hop. In other words, the second-degree neighbor information is used only for bypassing a particular link. They have also suggested a bypass algorithm for ATM networks. Upon a VP construction, loaded areas are identified and bypass routes are created to be used when the primary route is blocked. Thus, in [9], having found a path (by some other means), the second-degree neighborhood information is used only for bypassing a link whereas we use the same information for building an entire path from a source to the destination. Our approach could also be extended so that a router stores information about its nth degree neighbor. However, as we increase n, the impreciseness in the information stored by a router also increases proportionally. If $n = 1$, our approach becomes same as the flooding given in [2]. If n equals the total number of routers in the network, our approach becomes same as the source routing. Our algorithm is generic and can be used with both additive metrics (such as delay and cost) as well as concave metrics (such as bandwidth). In this paper, we have explained our algorithm taking bandwidth as the metric. The rest of the paper is organized as follows. We introduce our algorithm in Sections II and III. Experimental results are given in Section IV and we conclude in Section V.

The routing algorithm we propose has two separate tasks, namely, *table maintenance* and *packet forwarding*. The table maintenance component is responsible for constructing and maintaining the entries about the second-degree neighbors in a routing table. This is explained in Section II. The packet forwarding mechanism is responsible for forwarding the connection requests using the "two-level" routing table and is explained in detail in Section III.

II. Routing Table Maintenance

We assume that 1) all nodes store their local metrics and 2) all nodes know when to send updates to their neighbors. The first assumption is valid since a node always knows the resources available in its own outgoing links. The second assumption would become valid if an update policy is prescribed. Various update policies are discussed in [1] and any of them could be used. In this paper, we have used an update policy based on *thresholding* which will be discussed in detail in Section II-B. Each node maintains a Link-to-Node (LTN) table. The LTN table basically gives which link to use to reach a given neighbor and the resource available along that link. Links are assumed to be asymmetric, i.e., the metric available in the forward direction need not be the same as that in the reverse direction. A node can easily construct this table by exchanging *Hello* packets with neighbors. Each node on booting up constructs an LTN table.

A. Building the Routing Table

Apart from maintaining an LTN table, each router also maintains a "routing" or "forwarding" table. On receiving a connection request probe, a router uses this forwarding table to decide on what outgoing links the probe must be forwarded. Let us consider a node v. Let

- $N^1(v)$ denote those nodes that are adjacent to v in the network.
- $E^1(v)$ denote the links that connect v to nodes in $N^1(v)$.
- $N^2(v)$ denote those nodes that are adjacent to nodes in $N^1(v)$.

- $E^2(v)$ denote the links that connect nodes in $N^1(v)$ to nodes in $N^2(v)$.

The forwarding table of v contains information about the metrics of all the links in $E^1(v)$ and $E^2(v)$. Entries corresponding to $E^1(v)$ are called the *first-level entries* R_v^1 of v; Entries corresponding to $E^2(v)$ constitute the *second-level entries* R_v^2 of v. The second-level entries are represented as a tuple of the form $\langle l_i^1, l_j^2 \rangle$ where $l_i^1 \in E^1(v)$ and $l_j^2 \in E^2(v)$. If a node say u is a member of both $N^1(v)$ and $N^2(v)$, it is represented only as a first-level entry. In order to construct and maintain the routing table, node v must receive updates about the QoS metrics in all its second-level entries from the nodes in $N^1(v)$. This is done by exchanging special messages called *Hello2* packets at a frequency determined by the update policy used. These *Hello2* packets are constructed by copying the neighbor list and the available QoS metrics from the LTN table of the router. At a node v, the first-level entries in the routing table are made by copying the LTN table of v. The second-level entries in the routing table are made by inspecting the received *Hello2* packets. All the existing second-level entries in the routing table are updated by the *Hello2* packet. Also, any new entry is added to the existing second-level entries.

B. Update Policies

The main idea of sending *Hello2* packets is to communicate the changes in a router's resource availability to other routers. If a router sends *Hello2* packets every time a change occurs, a lot of overhead would be created in the network. To reduce this overhead, an update policy is prescribed. The update policy used decides when these *Hello2* packets are sent. A simple update policy could be based on timers and it could be such that an update is sent every T seconds. Protocols like OSPF and RIP send updates at regular intervals of time. While such an approach is acceptable for best-effort routing, it is not suited for QoS routing. The reason being that, within the update interval, the resources available in the routers can change drastically. If this change is not communicated to other routers, they will have imprecise (or stale) information and, hence, the performance of QoS routing will degrade. Also it is very difficult to model the impreciseness in the table entries with such an update mechanism. A detailed survey on various update policies can be found in [1]. The update policy used in our work is the one suggested in [10]. Each node remembers the last advertised metric on each link. If the ratio between the last advertised value and the current value is above (or below) a threshold τ, an update is triggered. The node constructs *Hello2* packets and sends them to all its neighbors. The advantage of using such a threshold-based update policy is that, the impreciseness could be easily modeled using probabilities. If bandwidth b is advertised on a link, and if say τ is 2, then at any time, the actual metric available on that link can be modeled as a uniform distribution in $[b/2, 2b]$. Once the impreciseness is modeled, there are approaches [10], [12] to do efficient routing with such imprecise information. However, our algorithm assumes that the information available in the tables is accurate and forward the probes accordingly. As a result, the performance of our algorithm in terms of call establishment might be poorer. Experimental results in Section V compare the performance of the flooding-based approach and forwarding-based approach on the two-level table. Existing routing protocols like RIP and OSPF have to be modified slightly to have an update mechanism based on a threshold. If RIP is used, a router has to send the bandwidth available on its incident links along with the routes for the best-effort traffic. Instead of sending periodical updates, the updates have to be triggered by the thresholding policy. For OSPF, already some extensions have been suggested to support QoS routing [11]. The only additional modification would be that, a router should send the bandwidth information only to its neighbors and need not send it to all routers in the network.

III. Packet Forwarding Mechanism

The forwarding mechanism suggested could be used for any QoS metric. In this paper, bandwidth is taken as the QoS metric and all discussions and results are with respect to bandwidth. An outline of the packet forwarding mechanism is given in Section III-A and a flowchart of the same is given in Fig. 1.

A. Outline

Each node v maintains a routing table in which two kinds of entries are present. The structure of the table is given in Table I. R_v^1 is the set of entries corresponding to the immediate neighbors of v. R_v^2 is the set of entries corresponding to the second-degree neighbors, namely, $N^2(v)$. A neighbor u of node v is said to be *eligible* if the link (v, u) can support the requested bandwidth. The connection setup process has three phases: 1) probing, 2) ack, and 3) failure handling. The first phase is probing and it is started when a source sends out a connection request. Each connection request is identified by an unique identifier cid. The connection request also known as probe, is a tuple of the form $[k, QoS(Bandwidth = B), s, t, cid, \{l\}]$. The probe format is interpreted as s is the source requesting a connection cid with metric requirements as $QoS(Bandwidth = B)$ to destination t and k is the router that has forwarded the probe to v. A source would set the k field to its own id and send the probe and $\{l\}$ refers to the list of neighbors to which v should forward this probe. On receiving this probe, v checks whether it is the first probe v has received for this connection. If not, then the probe is discarded as a duplicate. If the probe is the first for the connection cid, v marks that it has received a probe for the connection cid. It stores the upstream router id, referred to as $p_v(cid)$, that has forwarded the probe to it, in a table. The table entry is maintained for a duration T, which is the maximum connection setup time. After T, the entry is flushed out. Router v checks if the destination is present in its routing table as a first-level entry or as a second-level entry. If the destination is present and if it is eligible, the probe is forwarded in the corresponding link(s).

If the destination is not present in the routing table it means that the destination is beyond two hops. In such a situation, the list of neighbors $\{l\}$ to which v should forward the probe to is examined. The probe is forwarded to all eligible neighbors in the list. If the destination is beyond two hops and the

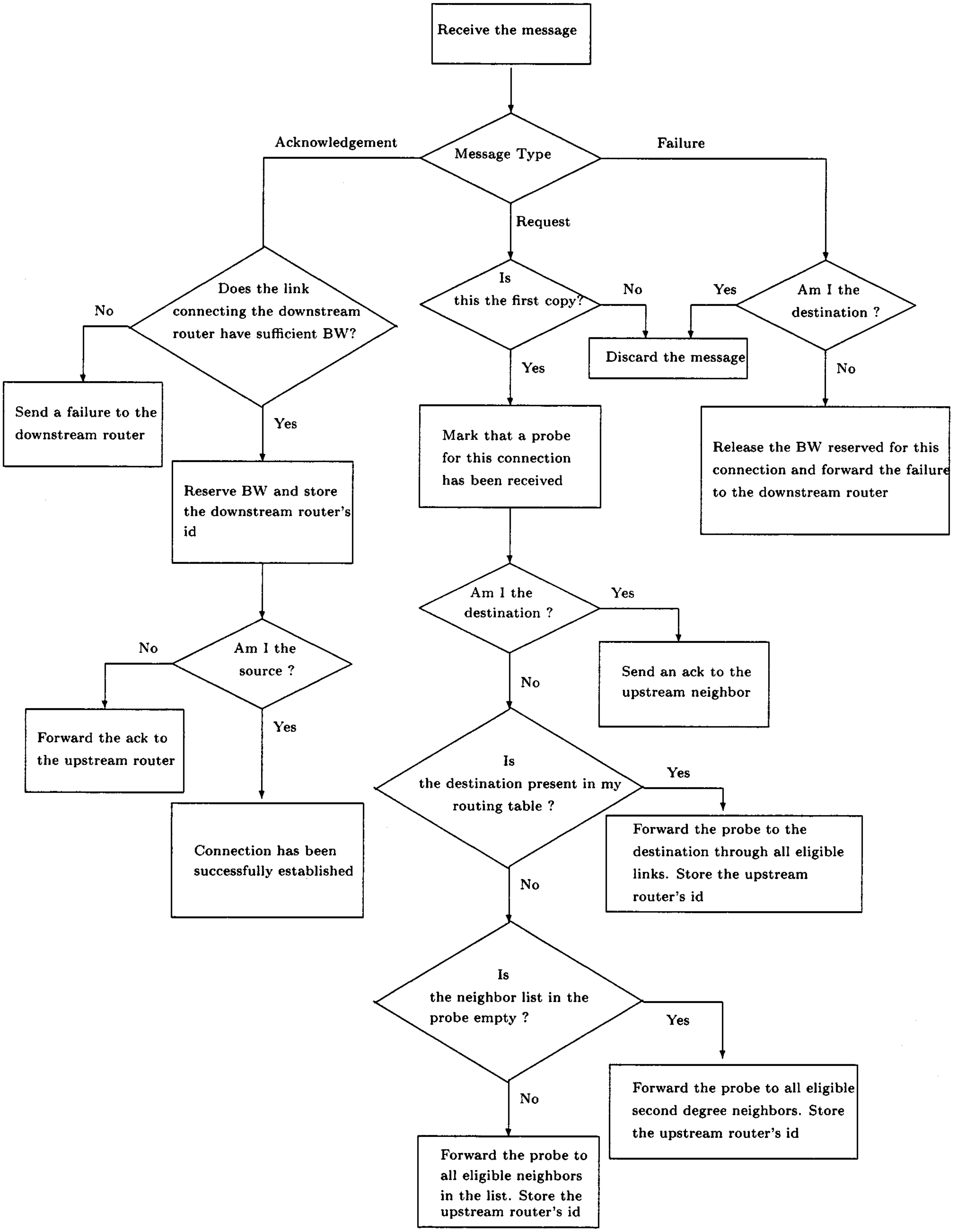
Receive the message
Message Type
Acknowledgement
Failure
Request
Does the link connecting the downstream router have sufficient BW?
No
Yes
Send a failure to the downstream router
Reserve BW and store the downstream router's id
Am I the source ?
No
Yes
Forward the ack to the upstream router
Connection has been successfully established
Is this the first copy?
No
Yes
Discard the message
Am I the destination ?
Yes
No
Release the BW reserved for this connection and forward the failure to the downstream router
Mark that a probe for this connection has been received
Am I the destination ?
Yes
No
Send an ack to the upstream neighbor
Is the destination present in my routing table ?
Yes
No
Forward the probe to the destination through all eligible links. Store the upstream router's id
Is the neighbor list in the probe empty ?
Yes
No
Forward the probe to all eligible second degree neighbors. Store the upstream router's id
Forward the probe to all eligible neighbors in the list. Store the upstream router's id

Fig. 1. Packet forwarding at a node.

list $\{l\}$ is empty, router v constructs a list on its own. If u is an eligible neighbor of v, a list of all eligible neighbors of u, l_u is constructed. Router v then forward a probe with $\{l_u\}$ to u. This is repeated for all eligible neighbors of v.

TABLE I
FORWARDING TABLE AT NODE v

Link	Next Hop	Available Metric
l_1^1	v_1	x_1B
⋮	⋮	⋮
l_n^1	v_n	x_nB
$< l_1^1, l_1^2 >$	u_1	x_kB
⋮	⋮	⋮
$< l_m^1, l_m^2 >$	u_m	x_rB

This forwarding process is repeated in all routers till the destination gets a probe. The path taken by the first probe to reach the destination is called the "tentative path." During the probing phase, only the resource availability is checked and no reservations are made.

The destination on receiving the first probe for a connection cid starts the *ack* phase by sending an acknowledgment to its sender. The acknowledgment looks similar to the probe but does not have the neighbor list field. A router on getting an ack checks whether the link on which the ack arrived has enough bandwidth to support the request. If the link has enough bandwidth, the router reserves the requested bandwidth for the connection cid on that link and stores the downstream router's id, referred to as $n_v(cid)$, in a table. This table entry is also maintained for a time T after which it is flushed out. The router then forwards the ack to its upstream router in the tentative path using the $p_v(cid)$ value it stored during the probing phase. This process continues until the source gets an ack. If the source gets an ack, the connection setup is complete and the connection is established. If any router in the tentative path is unable to reserve the requested bandwidth, it starts the failure handling phase by sending a failure message to the downstream router. If a router receives a failure message, it releases the resources it had reserved for this connection and forwards the failure message to the next downstream router on the tentative path using the $n_v(cid)$ entry.

The destination sends an ack only to the first probe it receives and discards all the duplicates. This makes sure that the resources are reserved only along one path. A router does not forward a probe more than once. This means that the tentative path found by the algorithm will be loop free. We also assume that these control messages are never dropped in the case of a congestion. A flowchart of the packet forwarding is given in Fig. 1.

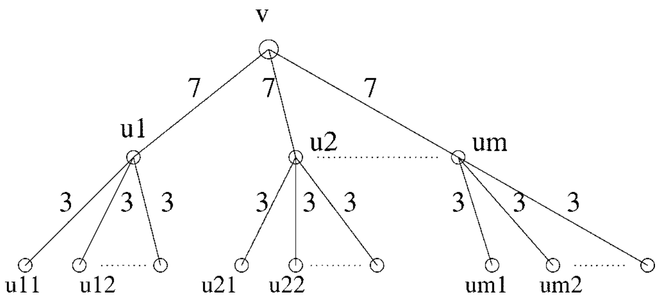

Fig. 2. Example network.

B. Bounded Two-Level Forwarding

The flooding-based approach finds a tentative path through competition among probes. If the network load is light, it is not a wise idea to blindly flood the probes on all eligible links. Often it is only the shortest eligible path that is preferred. To direct the search along the shortest path an approach was suggested in [2]. Each probe is assigned an age. The age of a probe p is defined as the number of hops the probe has traveled. Initially, $age(p) = 0$. Whenever p reaches a node, the age field is incremented by one. In order to direct the search along the shortest path, the probe forwarding condition on link (i, j) at node i is modified as

$$\begin{aligned}&\mathit{forward\ condition\ on\ link}(i, j)\\&\quad \rightarrow \mathit{bandwidth}(i, j) \geq B \wedge (\mathit{age}(p) + d_{j,t} + 1 \leq L)\end{aligned}$$

where $d_{j,t}$ is the shortest distance in terms of hops between the node j and destination t and L is a constant at least as large as $d_{s,t}$ and is the maximum age attainable by a probe. This forwarding condition would make the nodes flood the requests only along the shortest paths. Hence, this results in a much less overhead when the network load is light. When the network becomes heavily loaded, it is unlikely that this shortest path approach will succeed in establishing a path. Hence, if no path is established using $L = d_{s,t}$, the source makes a second attempt for the same connection with $L = \infty$. Flooding with $L = \infty$ is equivalent to flooding the probes blindly to all eligible nodes. In our simulation, all the sources make only one attempt with $L = d_{s,t}$.

Similar to the bounded flooding approach, we could also have an bounded approach for forwarding using the two-level table. The definition of an eligible node is modified similarly. If i is the current node of interest, a neighbor j is eligible if $\mathit{bandwidth}(i, j) \geq B$ and $d_{j,t} + \mathit{age}(p) + 1 \leq L$. Henceforth in our discussions, the term "unbounded" will refer to the probe forwarding without the hop constraint, while "bounded" will refer to the probe forwarding with the hop constraint.

IV. Experimental Results and Discussion

The motivation for forwarding based on a two-level table is to reduce the message overhead of flooding-based approaches. The example given in Fig. 2 helps in understanding how forwarding based on a two-level table could reduce the overhead. Let v be the node of interest and $u_1, u_2, \ldots, u_m$ be its m neighbors. Router v gets a connection request for a destination which is beyond two hops. Let the bandwidth requirement be five. It is clear that $u_1, u_2, \ldots, u_m$ are eligible and none of the neighbors of $u_1, \ldots, u_m$ are eligible. In such a scenario, if probe forwarding based on two-level approach is used, the probe will be discarded at v itself. However, if a simple flooding is used, router v would send m copies of the probe to $u_1, \ldots, u_m$ and the probes will finally be discarded at each of $u_1, \ldots, u_m$. Thus, blind flooding generates additional overhead. Also, in forwarding based on the two-level table, if the destination is within two hops, probes will be directed only toward the destination. On the other hand, if the probes are blindly flooded, apart from

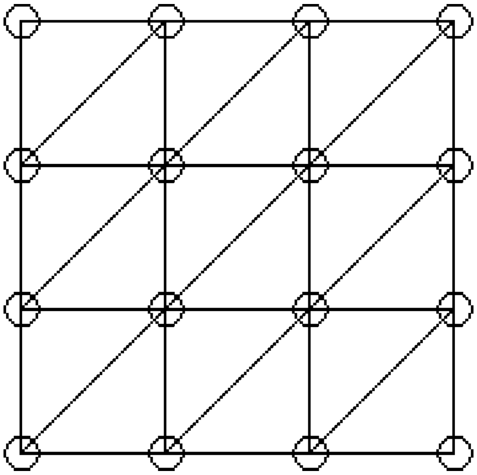

Fig. 3. MESH-I.

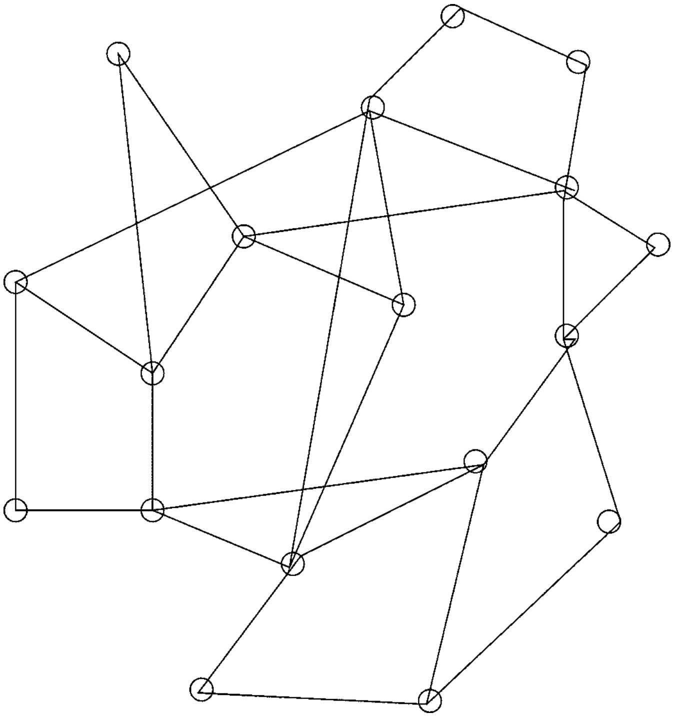

Fig. 4. Topology of a standard ISP.

the destination, many other nodes will also receive the probe. To have such a reduced overhead, additional information about the second-degree neighbors must be stored at each router. Maintaining this information creates additional overhead in the form of table maintenance. The two-level approach would be justified if the overhead created due to this table maintenance is much less than the savings in probe forwarding. The savings in the probe forwarding is dependent on resource availability and the network topology. In the discussions that follow, we shall refer to forwarding based on the two-level table as two-level forwarding.

Extensive simulations were done on varied network topologies to measure the total message overhead in both blind flooding and two-level forwarding. Due to space constraints, results are reported only from two network topologies. The two approaches were tested on the network topologies shown in Figs. 3 and 4. Fig. 4 is the topology of a standard ISP [15]. We believe that these collectively represent various network topologies that could be encountered. The simulations were done using OPNET, a commercial network simulation software. Each link is duplex and has a capacity of 155 Mb/s (OC-3). Bandwidth available in each link for reservation is set to a value in the range [0, 155 Mb/s]. All simulations were run for 2000 connection requests. The connection requests arrive at the nodes as per a Poisson distribution. The bandwidth requests are uniformly distributed between 64 Kb/s and 1.5 Mb/s. Each node in the network can generate a connection request for every other node in the network with equal probability. The connection durations are drawn as per an exponential distribution. The results could be divided into two sets. The first set is the comparison between the unbounded versions of flooding and two-level forwarding. The second set is the comparison between the bounded versions of the two approaches.

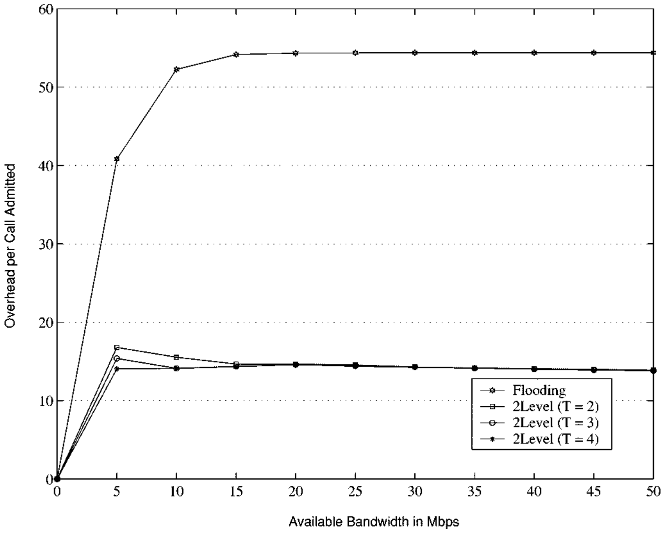

Fig. 5. Unbounded-flooding: Overhead on MESH-I.

A. Performance of the Unbounded Versions

The graph given in Fig. 5 shows the overhead in the unbounded versions of the two approaches on MESH-I. In this graph and the graphs that follow, T is the threshold value used in the update policy. It is clear that two-level forwarding has very low overhead (per call-admitted) when compared to blind flooding. When the available bandwidth in the network is less, the overhead increases significantly as the threshold T is reduced. However, when the available bandwidth is high, the value of T does not affect the overhead. This behavior could be explained as follows: When the available bandwidth is less, the current available bandwidth/last advertised bandwidth ratio will fluctuate significantly with each admitted call. As a result, if a low T value is used, the routers will send updates more frequently than at high T values. If the available bandwidth in each link is high, the current available bandwidth/last advertised bandwidth ratio will not fluctuate much with each admitted call. Hence, the routers tend to send updates less frequently irrespective of the T value.

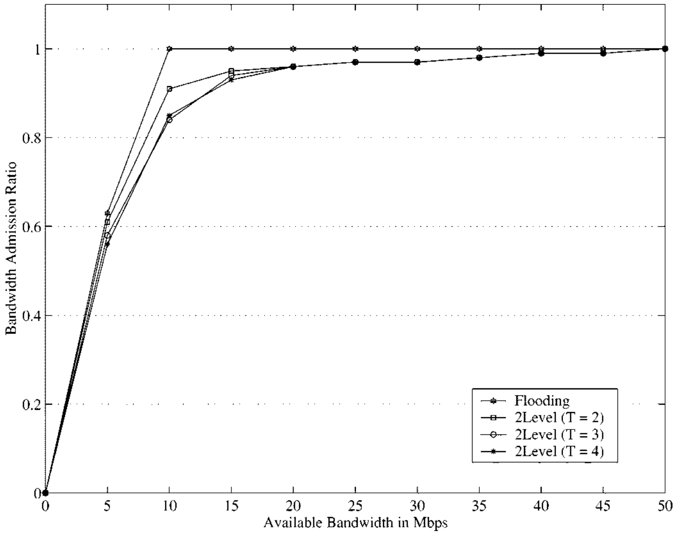

Fig. 6. Unbounded-flooding: Bandwidth admitted on MESH-I.

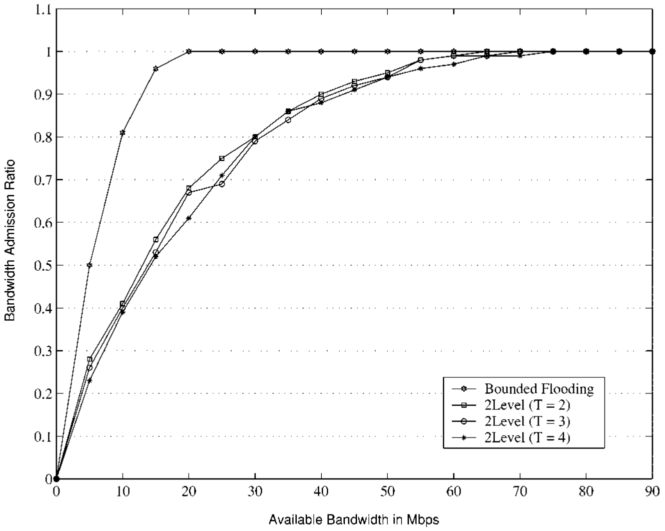

Fig. 8. Unbounded-flooding: Bandwidth admitted on ISP.

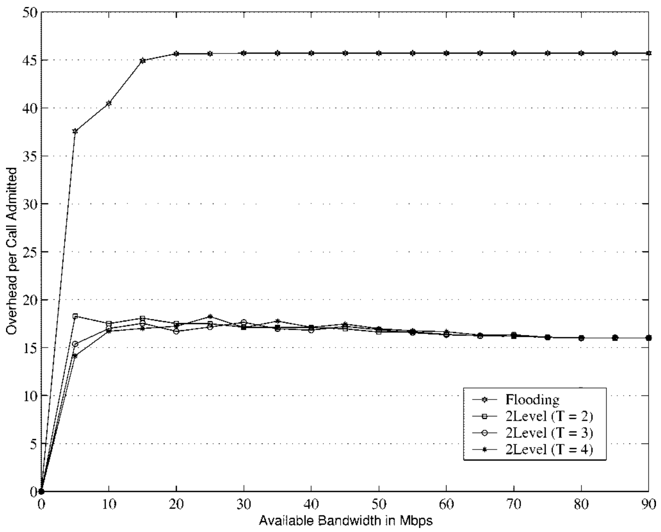

Fig. 7. Unbounded-flooding: Overhead on ISP.

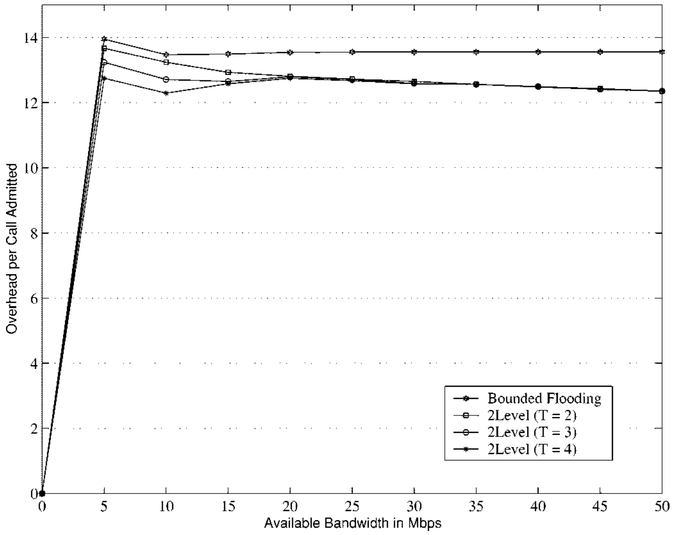

Fig. 9. Bounded (L): Overhead on MESH-I.

The graph in Fig. 6 shows the bandwidth admission ratio for the two unbounded versions on MESH-I. Bandwidth admission ratio is defined as the ratio of bandwidth admitted into the network to the total bandwidth requested [13]. The graph shows that when the load is light, both the approaches perform almost equally well. However, when the traffic is heavy, forwarding based on the two-level table admits less bandwidth into the network than flooding. Also, the bandwidth admitted by the two-level approach reduces as T increases. The reason is that the impreciseness in the table information increases with the value of T. This impreciseness makes the routers have a conservative estimate of bandwidth available in the second-level links. Hence, a router discards probes even though the second-level links are capable of supporting these requests. At low T values the impreciseness reduces. So the routers tend to be less conservative and they admit more bandwidth into the network.

Figs. 7 and 8 show the performance of the unbounded version of the two approaches on the ISP topology. As stated earlier, the reduction in message overhead depends on the topology. In MESH-I, there are a lot of alternate paths between any *(source, destination)* pair. As a result, in terms of message overhead, the two-level approach performs much better than blind flooding. Also in terms of bandwidth admitted, the two-level approach is comparable to the blind flooding. In the ISP topology, there are not many alternate paths between any *(source, destination)* pair. Hence, even though the two-level approach reduces the overhead, the reduction is slightly less than in MESH-I. The lack of alternate paths affects the bandwidth admitted by the two-level approach quite significantly. Thus by comparing the graphs in Figs. 6 and 8, the effect on topology on the two-level approach's performance can be clearly seen.

B. Performance of the Bounded Versions

The bounded version of the two algorithms were also compared. The motivation behind the bounded technique is to reduce unnecessary flooding in the network. In the bounded approach, the probes are given a maximum age L, equal to $d_{s,t}$, where $d_{s,t}$ is the number of hops in the shortest path between the

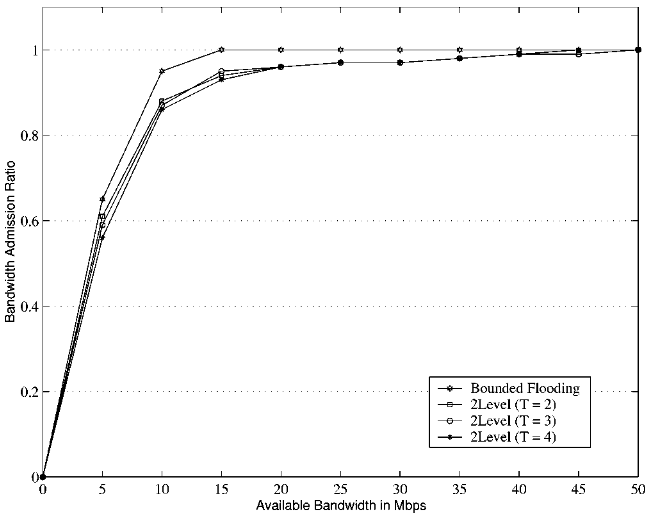

Fig. 10. Bounded (L): Bandwidth admitted on MESH-I.

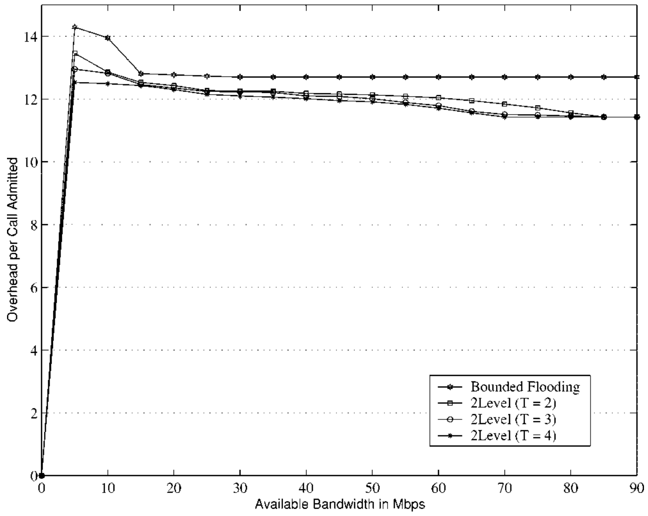

Fig. 11. Bounded (L): Overhead on ISP.

source s and destination t. The overhead in MESH-I for blind flooding and two-level forwarding is given in Fig. 9. The overhead of the bounded versions is clearly less than the overhead of the unbounded versions. Even here, the two-level approach helps in further reducing the message overhead. Fig. 10 shows the bandwidth admitted on MESH-I by the two approaches. As expected, at any given network load, the bandwidth admitted by the bounded versions is less than or equal to the bandwidth admitted by the unbounded versions. This is due to the reduced scope of the path search. Again on MESH-I, in terms of bandwidth admitted, the performance of the two-level approach is comparable to that of the blind flooding. Fig. 11 shows the overhead incurred on the ISP topology by the two bounded versions. The lack of alternate paths combined with the hop constraint make the blind flooding and the two-level forwarding comparable in terms of message overhead. In terms of bandwidth admitted, flooding performs better than the two-level forwarding. This can be seen from Fig. 12.

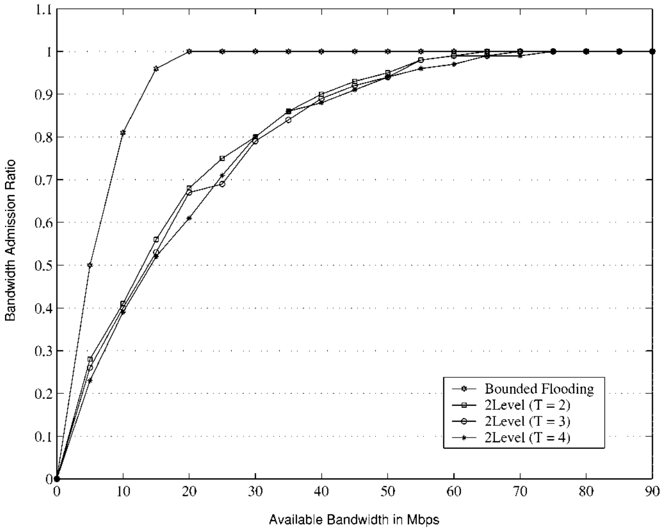

Fig. 12. Bounded (L): Bandwidth admitted on ISP.

V. Conclusion and Future Work

In this paper, we have proposed a new packet forwarding mechanism based on the QoS requirements of the connection. Our two-level forwarding has a low overhead when compared to the flooding-based call setup. Even though additional overhead is incurred in maintaining the information about the second-level links, simulation results show that this overhead is less when compared to the savings obtained by intelligent probe forwarding. Also, for networks with high connectivity, the impreciseness in the information stored does not greatly affect the total bandwidth admitted into the network. To exchange resource availability information between adjacent routers, protocols like OSPF and RIP could be extended. Future work would involve extending the proposed algorithm to do forwarding, taking into consideration the impreciseness in the table entries.

References

[1] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: A performance perspective," in *Proc. SIGCOMM*, Vancouver, BC, Canada, Sept. 1998.

[2] S. Chen and K. Nahrstedt, "Distributed QoS routing in high-speed networks based on selective probing," University of Illinois at Urbana-Champaign, Tech. Rep., 1998.

[3] F. Salama, D. S. Reeves, and Y. Vinotis, "A distributed algorithm for delay-constrained unicast routing," in *Proc. 4th Int. IFIP Workshop on Quality of Service*, Mar. 1996.

[4] K. G. Shin and C. C. Chou, "A distributed route selection scheme for establishing real-time channel," in *Proc. 6th Int. IFIP Conf. HPN*, Sept. 1995, pp. 319–325.

[5] C. Hou, "Routing virtual circuits with timing requirements in virtual path based ATM networks," in *Proc. INFOCOM*, 1996.

[6] S.-K. Kweon and K. G. Shin, "Distributed QoS routing using bounded flooding," University of Michigan, Ann Arbor, CSE Tech. Rep. CSE-TR-388-99, 1999.

[7] Q. Su and H. Langerdorfer, "A new distributed routing algorithm with end-to-end delay guarantee," in *Proc. 4th Int. IFIP Workshop on Quality of Service*, Mar. 1996.

[8] Z. Wang and J. Crowcroft, "QoS routing for supporting multimedia applications," *IEEE J. Select. Areas Commun.*, Sept. 1996.

[9] I. Cidon, R. Rom, and Y. Shavitt, "Bandwidth reservation for bursty traffic in the presence of resource availability uncertainty," *Comput. Commun.*, vol. 22, no. 10, pp. 919–929, June 25, 1999.

[10] R. Guerin and A. Orda, "QoS routing in networks with inaccurate information: Theory and algorithms," in *Proc. INFOCOMM*, Japan, Apr. 1997.

[11] R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *Proc. 2nd Global Internet Mini-Conference (jointly with Globecom'97)*, Phoenix, AZ, Nov. 1997.

[12] D. H. Lorenz and A. Orda, "QoS routing in networks with uncertain parameters," in *Proc. INFOCOMM*, Mar. 1998.

[13] Q. Ma and P. Steenkiste, "Quality-of-service routing with performance guarantees," in *Proc. 4th Int. IFIP Workshop on Quality of Service*, May 1997.

[14] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP a new resource ReSerVation Protocol," *IEEE Network*, Sept. 1993.

[15] D. E. Comer, *Internetworking with TCP/IP*. Englewood Cliffs, NJ: Prentice-Hall, 1995, vol. I, Principles, Protocols, and Architecture.

Donna Ghosh received the B.S. degree in electrical engineering from Regional Engineering College, Bhopal, India. She is currently pursuing the Ph.D. degree from the Department of Computer Science and Engineering, State University of New York at Buffalo. She is also with the Networked Multimedia and Visualization Laboratory directed by Raj Acharya. Her areas of interests are networking and information theory.

Venkatesh Sarangan received the B.S. degree in electrical engineering from Anna University, Madras, India. He is currently pursuing the Ph.D. degree in the Department of Computer Science and Engineering, State University of New York at Buffalo. He is also with the Networked Multimedia and Visualization Laboratory directed by Raj Acharya. His areas of interests are networking and fractals.

Raj Acharya received the Ph.D. degree from the University of Minnesota, Minneapolis.

He is currently the Chair of the Department of Computer Science and Engineering, State University of New York at Buffalo. He directs the Networked Multimedia and Visualization Laboratory. His areas of interest are quality of service in internet, bioinformatics, medical imaging, multimedia information retrieval, and fractals.