

WiNN: An Efficient Method for Routing Short-Lived Flows

SRINIVAS VUTUKURY
vutukury@cse.ucsc.edu
Cenus Technologies, Inc.
Scotts Valley, CA 95066

J.J. GARCIA-LUNA-ACEVES
jj@cse.ucsc.edu
Computer Engineering Department
University of California
Santa Cruz, California 95064

Abstract—Delivering real-time streaming content requires finding paths with a minimum required bandwidth. Finding such paths when requested should be fast (low startup latency) and efficient (high call acceptance rates). However, current algorithms for finding such QoS paths, are ineffective when bulk of the flows are short-lived. First, these algorithms are computationally expensive to justify invoking them on a per-request basis, and they add substantial latency to the signaling process. Moreover, they rely on frequent advertisement of residual link bandwidth, which is prohibitively expensive to perform on a short time-scale. Considering these drawbacks, we propose a simple heuristic WiNN (Widest Next-hop Neighbor) that has low startup latency and provides good call acceptance rates. The heuristic uses neither link state updates nor complex path selection algorithms.

I. INTRODUCTION

Techniques for finding *quality-of-service* paths for real-time streaming content is the focus of considerable research in the Internet community today. Many techniques for selecting and signaling QoS (such as delay and bandwidth) paths have been proposed. However, most of the methods proposed so far, such as *widest-shortest path*, though shown to be quite effective when routing long-lived flows, are unsuitable when routing flows that are of *very short duration*. The problem is further compounded when the flows are also of high bandwidth, such as in video streaming. It appears that a large part of the streaming video content over the future Internet will be from high-bandwidth short-lived flows (video clips). For example, news clips, advertisements, movie trailers, video emails, and the like, only last 10 to 60 seconds. Such flows make resource availability very dynamic. Current methods, mostly designed to handle streams of much longer duration in a quasi-static environment, are rendered ineffective and costly in highly dynamic networks.

This ineffectiveness is inherent to most QoS path selection frameworks existing today. These frameworks typically have three components: (a) advertising of bandwidth resources, (b) invoking a path selection algorithm at the source and (c) signaling of the reservation. A serious problem arises with this approach when the bulk of the streaming content tends to be of short duration. Previous works have shown that the frequency of bandwidth advertisements is co-related to inter-arrival times and duration of flows; the smaller the duration of the flows, the more frequent the advertisements must be made for obtaining the same network

utilization. Thus periodic advertisements are effective only when they are done on short intervals, which is an expensive affair. Also, the call-blocking rates increase as the periodic updates become longer.

Video clips demand low signaling overhead. For example, it is unacceptable to wait several seconds to view a 15 second news clip. The two-step process of path-finding followed by actual signaling of the path adds to the startup-latency that is significant compared to the duration of the flow. Considering these issues, we propose methods to handle such highly dynamic networks.

The key idea is to first setup *multipaths* apriori based solely on hop counts, and signal requests along the multipaths based on bandwidth[8]. There are no advertisement of bandwidth information. At each hop of the signaling process, only the available bandwidth on the local links is considered in the link admission test. There is no separate path-selection phase – the signaling phase begins as soon as the request arrives – thus reducing the startup-latency. We show that the performance and utilization of the network are quite reasonable compared to the best of the current methods.

The rest of the paper is organized as follows. Section II presents a brief survey of current research on path-selection. Section III describes in detail our path selection scheme, and section IV presents performance results to show the effectiveness of our scheme. Section V concludes the paper.

II. RELATED WORK

Though QoS routing in general refers to finding paths with multiple constraints such as bandwidth, delay, packet loss, path length, and the like, today, most researches believe bandwidth is the most critical metric. Once bandwidth is guaranteed, other assurances such as delay and packet loss can be easily given [5], [9]. Thus, several methods, including *widest-shortest path*, have been proposed for efficient QoS routing based on these two metrics: available bandwidth and path length [3], [2]. They combine path selection algorithms, link-state advertisements, location of the link state database, and the like, in variety of ways to achieve high levels of performance. Performance of QoS routing in this context is typically measured in terms of the *call-acceptance rate* of the requests made to the network.

The main conclusion of this prior work is that path lengths need to be minimized, while the bottleneck bandwidth of the path should be maximized to achieve high call-

acceptance rates. Because these heuristics are computationally expensive, several *path-caching* techniques have been proposed (e.g., [1]). These techniques avoid invoking path selection for each incoming request by reusing paths determined previously.

Some studies focused on the efficient advertising of link-state information. Various strategies for link-state advertisements, such as triggers and periodic advertisements, have been proposed and evaluated (e.g. [7] and references within.). The studies established that there is a tradeoff between the frequency of link-state updates and the call acceptance rates that can be achieved. However, there is a fundamental limitation on the accuracy of the advertised information, because of propagation delays and computing paths based on inaccurate information leads to high call-blocking rates. Thus, some works have questioned the use of bandwidth advertisements and proposed methods that use on-line measurements at a source to determine the quality of several explicit paths to a particular destination [4]. However, the drawback is that on-line measurements are computationally expensive. In addition, the approach assumes a connection-oriented architecture, such as MPLS[6], for setting up alternate paths.

The location of the link-state database on which the path-selection algorithm operates is also an important factor. For example, in the *bandwidth broker* architecture, the link database is maintained at a central location[10]. This strategy gives high performance because of the accurate information of residual bandwidth on the links. Another advantage is that it requires no link-state advertisements. However, a severe drawback is that it can have high startup delay depending on the location of the bandwidth broker, because the source has to send a request to the bandwidth broker for obtaining the path. This may be unacceptable for routing short-lived flows. In addition, the same problems that plague centralized systems (such as single point of failure and scalability) apply to the bandwidth-broker architecture. Consequently, most of the architectures are distributed and each node maintains a link database.

III. WINN SCHEME

Consider a network with N nodes, and let N^i be the set of neighbors for node i . For a given subset $S \subset N^i$, the *widest neighbor* in S is simply the neighbor in S that has the largest residual bandwidth on the link leading to it. For each node i and destination j , define two subsets of N^i , the *successor* set S_j^i and the *peer* set \hat{S}_j^i . Let D_j^i be the distance from router i to router j measured in number of hops. The *successor* set $S_j^i = \{k | k \in N^i \wedge D_j^k < D_j^i\}$ is the set of all neighbors of a node that are *closer* to the destination than the node. Similarly, the *peer* set $\hat{S}_j^i = \{k | k \in N^i \wedge D_j^k = D_j^i\}$ is the set of all neighbors of the node that are at the *same* distance from the destination as the node.

Upon arrival of a flow request at the source, a signaling message specifying the destination j and the bandwidth r is forwarded towards the destination, by choosing at each

hop the widest neighbor from either successor or peer set, performing admission test and making reservations. If an admission test fails at any point, a release message is sent in the reverse direction towards the source to release the partial reservations. To prevent loops, the signaling message carries a bit flag *e-bit* indicating whether the packet was ever forwarded to a *peer* router (i.e., neighbor in \hat{S}_j^i) on its path so far. A router forwards the signaling message to a peer neighbor only if the *e-bit* is clear; otherwise, the packet is forwarded to one of the *successor* neighbor (S_j^i). If a signaling message is forwarded to a peer neighbor, the *e-bit* is set so that all the routers visited in future will forward it only to their successor routers. Thus, a signaling packet can be forwarded to a peer neighbor *at most once*, preventing the packet from looping. Consequently, a signal packet from i to j can traverse at most $(D_j^i + 1)$ hops. The detailed algorithm is given below.

The signaling message is of the form [REQ, j , r , *e-bit*]. The following actions are taken when the request message arrives at i .

1. Find the widest neighbor k in S_j^i . If the available bandwidth on link (i, k) is *greater than or equal* to r , reserve the bandwidth r on the link and forward the signaling message to k .
2. Otherwise, check if the *e-bit* in the signaling message is set. If the *e-bit* is already set the request is considered failed and a release message is sent in the reverse direction.
3. If the *e-bit* is clear, then find the widest neighbor m in \hat{S}_j^i . If there is enough bandwidth available on link (i, m) , reserve the bandwidth, *set* the *e-bit* and forward the signaling message to neighbor m .
4. Otherwise, request has failed and a release message is sent in the reverse direction.

We now illustrate the heuristic through an example. Fig. 1(a) shows a network with residual bandwidth in each direction. At node i , $S_j^i = \{m, n\}$ and $\hat{S}_j^i = \{k\}$. At m , $S_j^m = \{j\}$ and $\hat{S}_j^m = \{n\}$. At node n , $S_j^n = \{j\}$ and $\hat{S}_j^n = \{m\}$. Assume a request of rate 1 arrives at node i for destination j . The signaling message is forwarded to node n and from there to node j as shown in Fig. 1(b). The resulting residual bandwidths after the request is processed are shown in Fig. 1(b). Now assume a second identical request arrives at node i . Since there is no bandwidth on (i, n) , this time the request is setup along path $i \rightarrow m \rightarrow j$ as determined by WiNN and shown in Fig. 1(c). Let another identical request arrive at node i . Now, there is no feasible next-hop in S_j^i . Hence, the neighbor k in \hat{S}_j^i is chosen. The *e-bit* is set and the request message is forwarded to node k . From there, the reservations are made on links (k, m) and (m, j) , as shown in Fig. 1(d). Assume another request arrives at node i . Again the neighbor k is chosen, the *e-bit* is set, reservation is made on (i, k) , and the request message is forwarded to node k . Node k forwards it to m after reserving bandwidth on (k, m) . There is no feasible link in the set

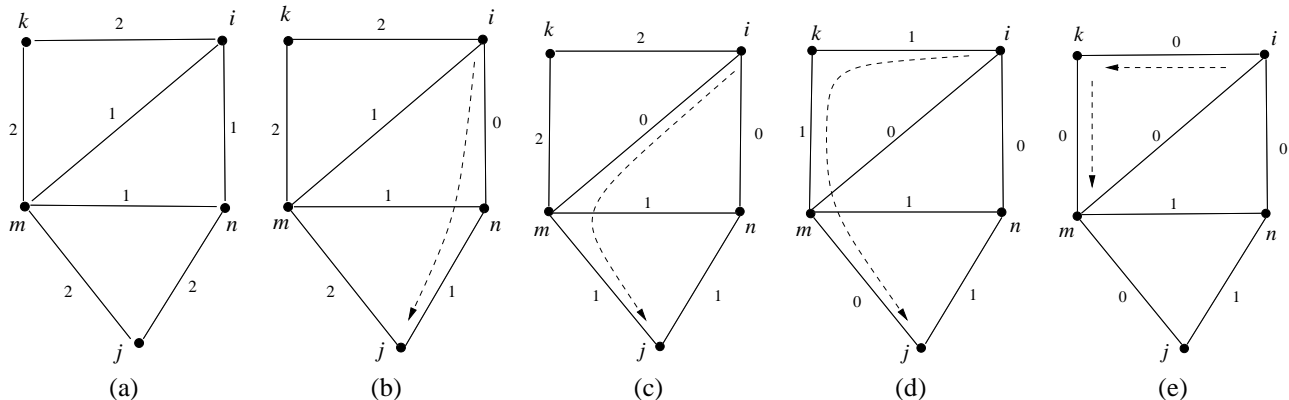


Fig. 1. Example illustrating the algorithm

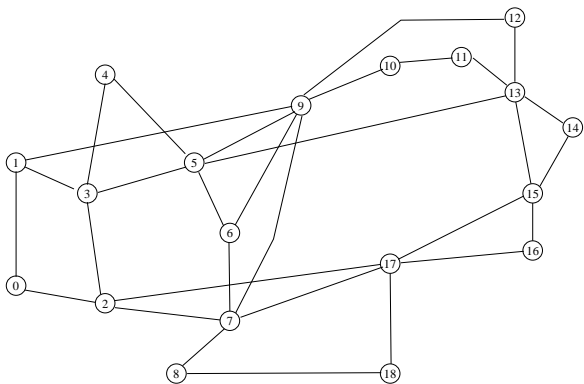


Fig. 2. ISP topology used in the experiments.

S_j^m at node m , but the link (m, n) is feasible. However, the link (m, n) is not selected because the e -bit is already set. So the request fails at node m .

WiNN differs from current approaches in several ways. The scheme has no separate path selection phase, i.e., the signaling is initiated as soon as the request arrives, which means that the signaling latency is minimal. There are no bandwidth advertisements, which contributes to the scalability of the routers. WiNN exploits the fact that a router has the most accurate information regarding its adjacent links, and the accuracy of information of other links decreases as the distance to the links increases. These is an important difference between WSP and WiNN regarding when a link is bound to the path. In the WSP approach, a link is included in the flow path at the source where the information regarding the link is most likely inaccurate. In contrast, this decision is made in WiNN at the router that is adjacent to the link and hence the information is accurate. Also, the decision to include a particular link in the path is taken in WiNN after performing the link admission test, unlike WSP, where the complete path is selected at the source and then varified by applying link-admission tests during the signaling phase.

IV. PERFORMANCE EVALUATION

A. Algorithms used in comparisons

We mainly compare WiNN with WSP and the shortest path algorithm. We compare with WSP under two modes: centralized and distributed. We briefly describe our implementation of these algorithms in our simulator CPT (an event-driven simulator from Nokia Corporation).

WSP in distributed architecture: Each node in the network maintains a local database containing information regarding all links in the network. This scheme assumes the existence of a topology broadcast routing algorithm, such as OSPF or IS-IS, for propagating the *residual bandwidth* information of the links. For our purpose, we implemented a topology broadcast algorithm based on sequence numbers that periodically advertises residual bandwidths.

The WSP algorithm is as follows[3]. When a request arrives, all the links that *do not* have the adequate bandwidth are pruned from the local database. Then the Dijkstra's shortest path algorithm is executed to find the shortest paths to the destination in the pruned network. If there is no path, the request is considered to have failed. If there is more than one path, the *widest* of the paths is chosen. Once a path is selected, source routing is used to signal the request from the source to the destination, performing an admission test at each hop along the path, and reserving the bandwidth if the admission test passes. If at any stage the admission test fails, the request is considered to fail and a release message is sent in the reverse direction to release the partial reservations.

WSP in Centralized architecture: In this scheme, instead of each node maintaining a local link database, a *central bandwidth broker* keeps the link-state of each link in the database. When a request arrives at a source, the source contacts the central bandwidth broker to select a path [10]. The bandwidth broker applies the WSP algorithm on its global link database to select a path. The broker returns the selected path to the source if successful. The source then signals the request along the selected path. Because only the central bandwidth broker executes the path selection algo-

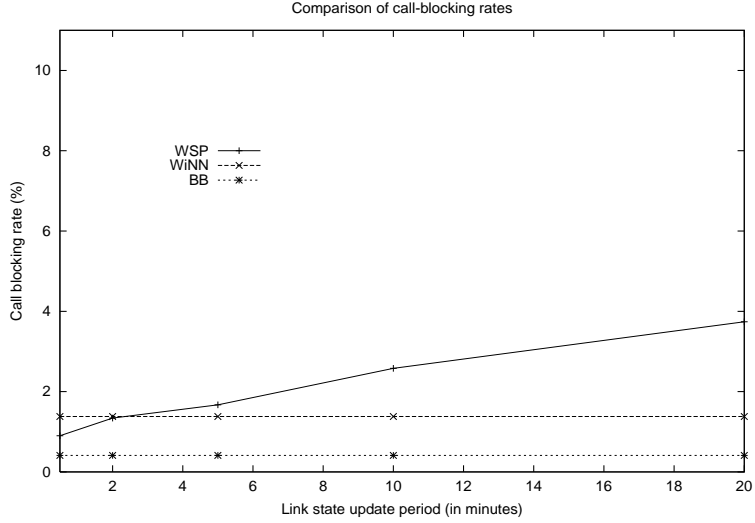


Fig. 3. Comparison of call-blocking rates as link update period increases: low load case

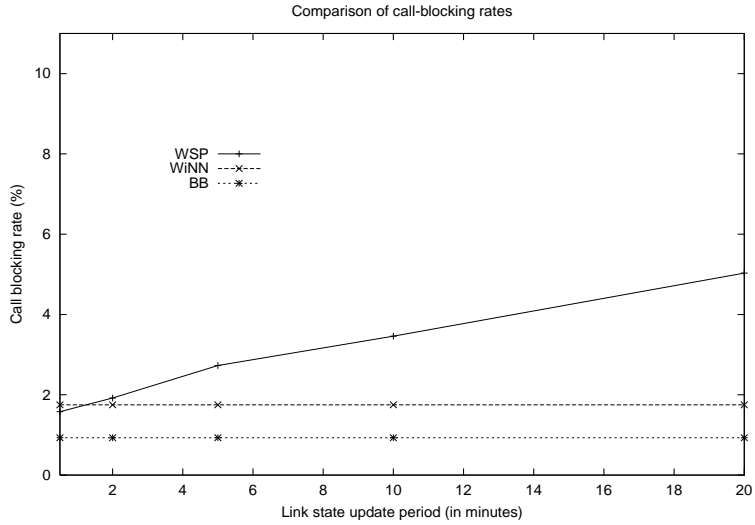


Fig. 4. Comparison of call-blocking rates as link update period increases: low-medium load case

rithm there is no need in this scheme to perform link admission tests. There is also no need of bandwidth advertisements. In our simulations, we implement an idealized version where the communication between the bandwidth broker and the nodes in the network is instantaneous.

Single Shortest Path: The shortest path scheme has some advantages when routing short-lived flows. Like WiNN, it requires no advertisements and path-selection as flows are always routed along the single shortest paths. But the drawback of shortest-path routing is its high resulting call-blocking rates. We use its call-blocking rates as a reference for worst-case performance.

We implemented a distributed shortest path algorithm for setting up the shortest path routes. When a router receives a request, a router simply performs an admission test, re-

serve the bandwidth on the next-hop on the shortest path and forwards it to the next-hop neighbor. The neighbor in turn performs the admission test and forwards it to its next hop along the shortest path. If the admission test fails at any point, a release message is sent back in the reverse direction, just as in the WSP scheme described above.

B. Call-blocking rates

The topology of the network used in the simulations is shown in Fig. 2. Each link has a capacity of 45 MB and a propagation delay of 100 μ s. The requests arrive at the nodes and are destined for other nodes in the network. For simplicity we keep the network stable during the simulations and do not simulate link and node failures. Failed requests are also not retried.

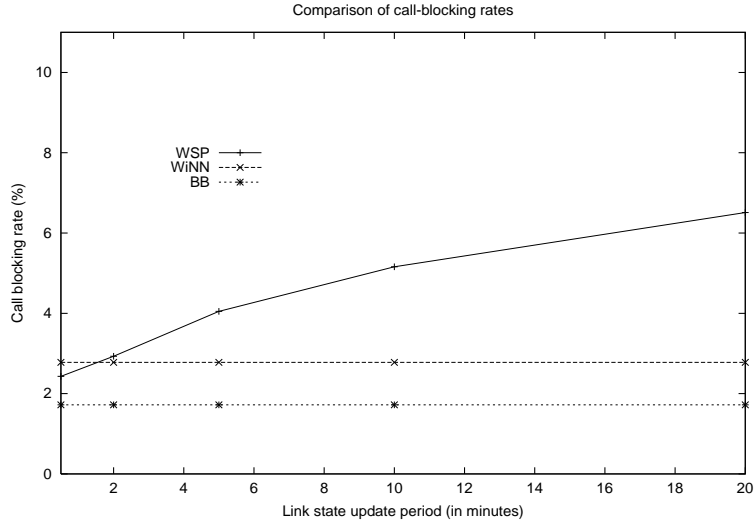


Fig. 5. Comparison of call-blocking rates as link update period increases: high-medium load case

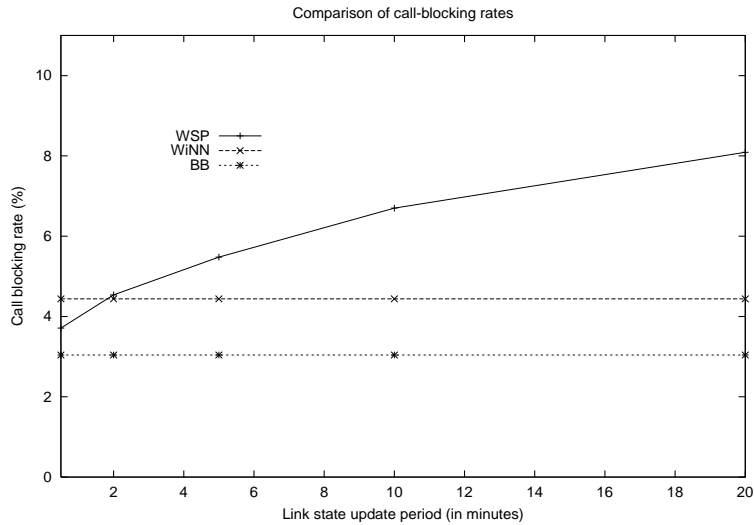


Fig. 6. Comparison of call-blocking rates as link update period increases: high load case

The request arrival rate is the same between each source-destination pair and follows a Poisson distribution. The duration of the flows is exponentially distributed with an average of 20 seconds while the size of the requests is kept constant at 1.2 MB.

Figs.3-6 show the call-blocking rates of WiNN, BB (WSP in centralized mode) and the WSP (WSP in distributed mode) under four different load conditions: low, low-medium, high-medium, and high. For WSP in the distributed mode, we varied the link update periods from 30 seconds to several minutes.

Notice that under all load conditions the call-blocking rates of WSP increase with the link-state update period, and become worse than WiNN beyond a certain point. In this case, it is 2 minutes, because of short lived nature of the

flows. We have used the 8% call-blocking rates as a cut-off point with the assumption that higher blocking rates are not unacceptable for any scheme. The link-state advertisements are only good when performed at a high frequency. In all the cases, BB performs the best since its link information is accurate. This shows that WSP scheme is ineffective when flows last for short duration.

Fig. 7 shows the call-blocking rates as the load or number of requests made to the network is increased. The flow arrival rate in the network is varied from 30 requests/min to 50 requests/min. The graph for WSP corresponds to the link-state advertisement period of 30 seconds (a high update rate). Observe that the performance of WiNN is comparable to WSP, especially relative to BB and the SP (shortest path) schemes. This indicates that WiNN can achieve the perfor-

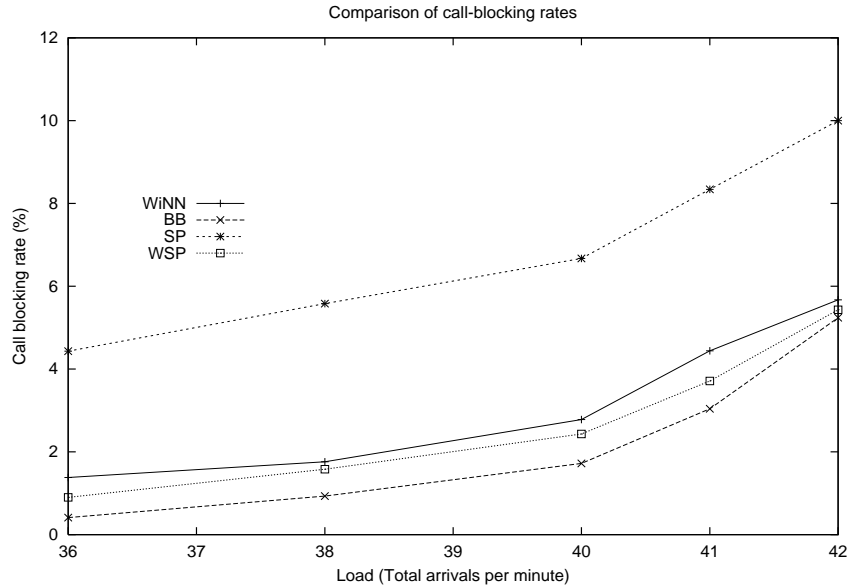


Fig. 7. Comparison of call-blocking rates as load increases. The link update period is fixed for WSP.

mance levels of WSP even when WSP uses very short update periods. The BB scheme again did better than the other schemes because of its accurate information regarding the network resources.

V. CONCLUSIONS

We presented WiNN, a simple and efficient algorithm for QoS routing, which is especially attractive when the flows are short-lived. The key feature of this scheme is that it neither uses link-advertisements nor complex path-selection algorithm, which has been the main drawbacks of conventional algorithms such as the widest-shortest path. As there is no explicit path-selection process, the flow establishment in this scheme has the lowest start-up latency, a desirable feature when majority of flows are short-lived. As there are no link-state advertisements, the routers are far more scalable. Typically when there are no link-state advertisements and explicit path selection, the call-blocking rates tend to be higher, but we have shown through simulations that WiNN's performance, measured in terms of call-acceptance, is comparable to and often better than conventional methods. The conventional methods have to use aggressive policies (e.g., very short update periods) that generate large amount of link-state updates to outperform WiNN.

REFERENCES

- [1] G. Apostolopoulos and et al. On reducing the processing cost of on-demand QoS path computation. *Journal of High Speed Networks*, 7:77–98, 1998.
- [2] G. Apostolopoulos and et al. Quality of Service Based Routing: A Performance Perspective. *Proc. of ACM SIGCOMM*, 1998.
- [3] Q. Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. *Proc. International Conference on Network Protocols*, October 1997.
- [4] S. Nelakuditi and et.al. Adaptive proportional routing: A localized qos routing approach. *Proc. IEEE INFOCOM*, 2000.
- [5] C. Pornavalai, G. Chakraborty, and N. Shiratori. QoS based routing algorithm in Integrated Services Packet Networks. *Journal of High Speed Networks*, 7:99–112, 1998.
- [6] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. *RFC-3031*, January 2001.
- [7] A. Shaikh, J. Rexford, and K. Shin. Efficient computation of Quality-of-Service routes. *NOSSDAV*, 1998.
- [8] S. Vutukury and J.J. Garcia-Luna-Aceves. A Scalable Architecture for Providing Deterministic Guarantees. *Proc. of ICCCN*, Oct. 1999.
- [9] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE Journal on Selected Areas in Communications*, 14:1128–1234, 1996.
- [10] Z. Zhang and et al. Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Services. *Proc. of ACM SIGCOMM*, pages 71–83, 2000.