Anycast Routing with QoS

Valerie Hajdik, Jianer Chen, Ge Xia
Department of Computer Science,
Texas A&M University

Abstract

Effective anycast routing algorithms are becoming more important as the rapid expansion of the Internet leads to increased use of replicated servers and other anycast services. At the same time, the popularization of real-time multimedia applications has led to a need for QoS to maintain desired levels of quality. Our research focuses on randomized anycast algorithms with QoS. We also consider techniques that can improve anycast routing performance under inaccurate link-state information.

Keywords: *QoS, anycasting, network routing, randomized algorithm*

Introduction

In order to keep up with the explosive growth of the Internet in the past decade, it has become necessary to find creative ways of using network resources. One popular technique for improving reliability and performance is to host the same service or data in several locations. Replicated servers of this type help to balance network load and guarantee that if one server should fail, there will be no loss of service. Anycast is a method by which one address can refer to any one of several locations. With anycast routing, a request for one anycast address leads to the selection of the "best" of several possible destinations. Anycast is especially useful because the end user does not need to know details of the selection, or even that a selection is taking place at all.

While the Internet grows in size, it also grows in scope. Where Internet traffic was once predominantly text-based, it now includes a wide variety of services including many real-time or resource-intensive multimedia services. The emergence of new types of Internet traffic raises new network resource issues. Under a heavy network load, an email can be delayed considerably but will still serve its purpose adequately when it does arrive. A real-time application such as an IP phone conversation, however, could be distorted by delay to the point of losing its usefulness. One solution to this problem is the introduction of priority to certain types of traffic using QoS. QoS stands for Quality of Service, a system which allows extra constraints to be placed on a request, such as a minimum bandwidth or maximum delay.

To ensure a certain level of performance, resource-intensive applications that require QoS are likely to be hosted on replicated servers, making it logical to combine anycast and QoS requirements. Little work, however, has been done in this area.

The purpose of this paper is to present a basic study of anycast routing with QoS considerations. We examine several elements: randomization of anycast algorithms, the role of QoS in anycast routing, and methods for coping with inaccurate link-state information. We use analysis and simulation to evaluate two proposed anycast algorithms.

We begin by presenting background information on anycast routing and QoS. We then take an in-depth look at our proposed methods. Next, we describe the structure of our simulation and present experimental results. Finally, we analyze the results and consider other work done in the field.

Background

Anycast

The two widely known types of routing are unicast and multicast. Unicast routing involves communication between one source and one destination. Multicast routing takes place between one source and several destinations. In contrast, anycast routing involves the selection of the "best" of several possible destinations. The goodness of a particular destination is determined by the routing algorithm, and may be based on metrics such as the destination's nearness in number of hops. In addition to providing redundancy, anycast services can help balance load across the network by distributing traffic among several locations. Because of its usefulness, anycast has been defined as a service in IPv6.

There are two approaches to anycast routing. Network layer anycast is applied at router level and involves only the information about the network that can be extracted from routing tables. Application layer anycast takes place when requests are sent to a central resolver, which selects a destination for each request and then forwards it using unicast. Application layer anycast has less fault-tolerance than network layer since it depends largely on a central resolver, but may include additional information about destinations such as the current load on a server. This paper concentrates on network layer anycasting, but also suggests a variant of application-layer anycast with QoS as a topic for future research.

Several challenges must be considered when designing a network-layer anycast routing algorithm. There are many possible paths a request can take, and the intended destination could change midway through a path. It is important, therefore, that the algorithm take care to prevent loops, which waste network resources. If the same destination is repeatedly selected, congestion can occur in one or more links leading to that destination. A good algorithm should attempt to balance the load across several destinations, even when one destination is consistently considered better than all the others. Finally, a routing algorithm is useless if it cannot be practically implemented, so an effective algorithm should be compatible with the current infrastructure of the Internet and should not be unreasonably costly.

QoS

QoS stands for Quality of Service, and refers to constraints placed upon a request to guarantee that a connection maintains a certain quality level. While an ordinary request may specify only the source and destination, a QoS request may also specify certain conditions that must be met, such as a minimum level of bandwidth, maximum delay, and maximum delay jitter. If no connection exists that satisfies all these constraints, the request will fail. QoS is mainly used to guarantee the success of real-time or resource-intensive applications such as video-on-demand or IP phone. These services require a certain level of quality in order fulfill their purpose, since a delayed or erratic connection could render the data unrecognizable. QoS is becoming increasingly important to guarantee a sufficient level of quality to the applications that need it.

Inaccurate Link-State Information

A major obstacle in network-layer routing is the inaccuracy of link-state information. With link-state routing each router in the network makes its decisions about where to send packets based on its view of the current status of the network. The state of the Internet changes when routers go offline or come online, or when the available bandwidth on a certain link increases or decreases. When a significant change occurs, an advertisement is flooded across the network to inform other routers of the new conditions. There is a period after the change has occurred and before all routers become aware of the change, when some routers may make routing decisions based on information that does not reflect the true state of the network. How long the danger period lasts depends largely on the update policy used by the routers. Some examples of udpate policies are threshold and exponential. With a threshold update policy, some threshold is defined, based on a percentage of the total available bandwidth in a link. When the bandwidth available on that link differs by more than the threshold amount from the last advertised bandwidth, a new advertisement is triggered. An exponential update policy partitions possible bandwidths into several ranges, with low-bandwidth ranges being very small and high-bandwidth ranges being large. If the available bandwidth crosses into a different range from the one last advertised, a new advertisement is initiated. The logic behind using large ranges for high bandwidths is that once the available bandwidth becomes sufficiently high, additional bandwidth does not make much of a difference. Update policies are an interesting research topic in their own right, but in the scope of this paper we limit our discussion to precautions that may improve the performance of routing algorithms in the presence of inaccurate link-state information.

Related Work

Related work that has been done in the field of anycast routing includes a distributed anycast algorithm, and application-layer anycast algorithms. In [2], the authors propose Distributed Admission Control for anycast flows. A distributed approach is useful because it divides the overhead among different nodes and it has a high fault tolerance since all computation is not dependent upon one central location. The downside to this approach is that the performance of the algorithms depends on the amount of information available to each node, which can quickly bloat the size of information that must be passed with each packet.

Related work has also been done on application-layer anycast, where an anycast resolver maps an anycast request to a destination.[3] Application layer anycast is useful because it can be

implemented without making any low-level changes to the Internet or its routing paradigms. Furthermore, many dynamic DNS services already exist which could be modified to handle anycast queries.

Our Approach

There are three major elements in our approach to the anycast problem. First, we use weighted random selection to assist in load balancing. Weighted random selection is a common approach to anycasting, discussed in [1]. Second, we take QoS into consideration by checking certain parameters to make sure they fall within a range of desired values. Third, we examine the benefits of safety-based routing, a method which depends on the likelihood that available bandwidth falls within an acceptable range.

We consider two algorithms, RBH and RBHS, which were proposed in [6]. RBH stands for Random Based on Hops. This algorithm combines the first two elements of our approach: randomization and QoS. RBHS stands for Random Based on Hops and Safety. In addition to randomization and QoS, RBHS also takes steps to improve performance under inaccurate bandwidth information.
The Random Based on Hops (RBH) algorithm is as follows:

Algorithm RBH (Random Based on Hops)

Input: a network G = <V, E> and an anycast QoS request R = (S, A, B, N)
Output: a routing path *p* that satisfies the anycast QoS request R.

(1) for each link $l \in E$ in G
  if $B_l < B$ **then** delete *l*;
Let *G′* be the resulting graph;

(2) for each destination *D* in *G(A)*
  for each node *v* in *G′*
    find the length *H(v, D)* and the next link *L(v, D)* of the shortest path from *v* to *D*.

(3) for each node *v* in *G′*
  if *H(v, D)* > *N* for all destinations *D* in *G(A)* **then** delete node *v*.
Let *G″* be the resulting graph;

(4) Search a path satisfying the anycast QoS request *R*;
  now = S; *nowdelay* = 0;
  repeat
    if there are links $e_1, e_2, \ldots, e_k$ where for each *j*, $e_j = [now, v_j]$
    then with probability p_j pick the link e_j; (the probability p_j will be explained below)
    $now = v_j$; $nowdelay = nowdelay + 1$;
  until *now* is in *G(A)*;
  output the routing path;

The probability p_j in the above algorithm is decided in the following manner. Suppose that in the **repeat** loop in step 4 of the algorithm, a partial path P from S to the node *now* has been constructed, such that the length $|P|$ (i.e., the delay) of the path P is *nowdelay*. We also assume that there are links *e1, e2, ..., ek*, where for each *j, ej = [now, vj],* and there exists at least a destination D in *G(A)* such that

$$nowdelay + 1 + H(now, D) \leq N$$

Then the probability of selecting the destination D^i in the node *now* is given by the following formula:

$\mathbf{A} = \sum_{t = 1 \text{ to } w} \prod_{j \neq t} [\, H(now, D^t) + nowdelay \,]$
$\mathbf{B} = \prod_{t = 1 \text{ to } w} [\, H(now, D^t) + nowdelay \,]$
$\mathbf{C} = 1 / H(now, D^i)$

$P_{D\,i} = \mathbf{A} / \mathbf{B} * \mathbf{C}$

Note that if $H(now, D^i) + 1 + nowdelay > N$ then $P_{D\,i} = 0$, and $H(now, D^i) + nowdelay$ is removed from the above expression.

Given the probabilities of selecting each destination from the formula above, the probability with which we pick the link $e_j = [\, now, v_j \,]$ is

$$p_j = \sum_{L(now, D\,i) = e\,j} p_{D\,i}$$

In general, a routing path with short delay has a higher probability of being picked with the RBH algorithm. The most complex step of the RBH algorithm is Step 2, finding the shortest paths. Using a modified version of Dijkstra's algorithm, this step can be performed in time O(w*(m + n log n)), where w is the number of designated recipients in *G(A), m* is the number of edges in the graph, and n is the number of nodes. Step 1 of the algorithm can be performed in time O(*m),* since each link will be checked for sufficient bandwidth only once. Step 3 takes time O(*n)*. Step 4 can be done in time O(*m*), because no edges will be repeated in the path that is produced.

The RBH algorithm is a reasonable adaptation of weighted random selection to fit the needs of an anycast algorithm with QoS, but it has a major setback; RBH assumes perfect knowledge of the state of the network at all times. In practice, information about the state of the network is frequently inaccurate. The Random Based on Hops and Safety (RBHS) algorithm seeks to compensate for inaccurate information about network resources by introducing a safety variable. When the update trigger policy of the network is known, it is possible to predict the likelihood that available bandwidth on a particular link is greater than or equal to the requested bandwidth. If the bandwidth that is advertised for a particular link, for instance, is $B_{adv,}$ and the network uses a threshold update trigger policy with threshold h, then the lower bound and upper bound of the range of possible bandwidths can be computed as follows.

$B_l = B_{adv} * (1 - h)$ and $B_u = B_{adv} * (1 + h)$

Using this information, the "safety" level of a link, S_l, can be defined according to B_{req}, the requested amount of bandwidth. If $B_{req} \geq B_u$, it is certain that link l does not have sufficient bandwidth to fulfill the request, and $S_l = 0$. When $B_{req} \leq B_l$, link l definitely has enough bandwidth and $S_l = 1$. For $B_l < B_{req} < B_u$, we can compute a probability with which the link bandwidth is larger than B_{req} with the formula $S_l = (B_u - B_{req}) / (B_u - B_l)$.

Bandwidth Safety

When $B_{req} \geq B_u$, $S_l = 0$.
When $B_{req} \leq B_l$, $S_l = 1$.
When $B_l < B_{req} < B_u$, $S_l = (B_u - B_{req}) / (B_u - B_l)$

It should be noted that this method of calculating bandwidth safety assumes that the values of actual available bandwidth are distributed uniformly between B_l and B_u, as proposed in [4].

The general idea is that the safety calculated above can be incorporated into the randomized algorithm such that, all other things being equal, a link with higher safety is more likely to be picked. The safety concept was discussed in [5], but to our knowledge has not been applied to anycast routing. We intend to look more closely at an anycast routing algorithm that takes bandwidth safety into account in the future.

Methods

To study the performance of anycast algorithms under various conditions, we have developed a network simulation. In this section, we discuss the general structure of the simulation, then we describe and elaborate on its main components.

The simulation was written in the C programming language using the CSIM discrete event simulation libraries. CSIM is designed for process-oriented simulation, where events are driven by a number of concurrent processes. In our case, the main processes are requests, and link-state updates form a secondary process. The arrival of requests is modeled as a Poisson process, using a random stream from the CSIM libraries that is based on the exponential distribution. Each request has a number of parameters, including source, destination, and QoS requirements. Upon initiation of a request, a path for the request to follow is found based on the algorithm being used. If a path can not be found, the request fails. If a suitable path is found, an attempt is made to forward the packet along that path. If at any time the attempt to forward the request violates the QoS restrictions, the request fails. Information about each request is written to an output file, and a record is kept of the total number of successes or failures. Performance of an algorithm is measured in bandwidth acceptance ratio, which is defined as the number of bandwidth units successfully reserved, divided by the total number of bandwidth units requested.

Our simulation is composed of four main components: the main simulation, graphs, algorithms, and bandwidth update functions.

The main simulation includes all elements directly related to CSIM. In this component, elements of the network are mapped onto CSIM elements. In CSIM, a facility is an element that can be used or reserved by a process. We define an array of facilities that corresponds to the routers in our network. A CSIM storage is an element that contains some number of units which may be allocated or deallocated by a process. An array of storages in our simulation corresponds to the links of our network. Allocation or deallocation of these storages corresponds to the consumption of bandwidth by a request. The CSIM libraries contain methods that allow the user to generate several types of random streams. We use an exponential random stream to model request arrival as a Poisson process. In our simulation, we separate physical and logical destinations. Physical nodes correspond to the routers of the network and logical nodes correspond to logical destinations. Each logical destination may be mapped to one or more physical destinations, an arrangement which reflects the nature of anycast. A unicast destination - a mapping of one logical node to one physical node - is treated as a special case of an anycast destination. We choose a source from the set of physical nodes with each node having equal probability of being chosen. A destination is randomly chosen from the set of logical nodes.

Our method of mapping logical destinations onto physical nodes gives our simulation a great deal of flexibility in the ratio of anycast to unicast nodes. When every logical destination is mapped to one physical node, the simulation becomes a unicast simulation. When each logical destination is mapped to several nodes, every destination becomes an anycast destination. In contrast, most anycast simulations specify a static number of nodes to act as anycast destinations. Although some of these variations do not reflect the true nature of the Internet, they may provide insight into the workings of algorithms under extreme circumstances. Also defined in the main body of our simulation are requests, which are modeled as CSIM processes. Each request consists of a source, a logical destination, and several QoS requirements. The request process identifies the physical nodes associated with the logical destination and attempts to find a path from the source to one of the possible destinations using the specified algorithm. Upon successful identification of a path, the request reserves routers and allocates bandwidth along the path to simulate completion of the request. Simulation time passes during the request process, then the routers are released and the bandwidth is deallocated.

Graphs

Several functions are defined in our simulation to handle graphs within the program. A graph is defined as a set of vertices, V, and a set of edges E(v_1, v_2), such that two vertices are adjacent if and only if they are connected by an edge. Several graphs are created and used internally in the simulation. One graph represents the initial state of the network, and an array of graphs represents each router's idea of what the network looks like. The initial graph is obtained from an external file. Currently the topology file is generated by the BRITE random topology generator from Boston University. BRITE will be discussed in more detail later. Each graph in our simulation is dynamically allocated and stored as an adjacency matrix.

Algorithms

Although our simulation is designed to test several anycast routing algorithms, we have attempted to make the algorithms as modular as possible so that particular elements may be reused. Dijkstra's algorithm is used to find the set of shortest paths from one node to every other node in a given graph. In addition to the RBH and RBHS algorithms, we include one shortest-

shortest path algorithm with accurate bandwidth information and another with inaccurate bandwidth information.

Inaccurate Link-State Information

To model inaccurate bandwidth information in our simulation, we use an array of graphs representing each router's idea of what the network looks like at a given time. Initially, each router has an accurate perception of the state of the network. As bandwidth is allocated or deallocated, a check is performed to determine whether the change is worth reporting. Whether or not the change is significant depends on the update trigger policy. Our simulation uses a threshold update trigger policy, where a threshold is defined as a percentage of the total capacity of link. When the difference between the new bandwidth and the last reported bandwidth exceeds the threshold, a bandwidth advertisement is flooded through the network using breadth-first traversal. For the RBH and shortest-shortest path algorithms, the actual state of the network is used when making routing decisions. For the RBHS and inaccurate shortest-shortest path algorithms, each router makes a decision based upon its own personal idea of the state of the network.

The networks on which the simulations were run were generated by the BRITE topology generator from Boston University. The number of nodes was varied between 10 and 1000, with connectivity based on the Barabasi model. The Barabasi model is characterized by preferential growth, meaning that nodes that are well-connected have an increased chance of becoming more well connected. Often compared to the phenomenon of "the rich getting richer," this growth model yields a graph with a heavy-tailed degree distribution. In other words, there are a few nodes with very high degree and many nodes with low degree. Studies have shown that the Internet exhibits a heavy-tailed degree distribution.

The simulation was run using several different BRITE output files, each of which had either 10 or 1000 nodes. Six nodes were designated anycast nodes in each graph, corresponding to two possible anycast destinations of three nodes each. The average time interval between requests was varied from 0.1 seconds to 100 seconds and bandwidth acceptance ratio was recorded for each run. Bandwidth acceptance ratio is defined as units of bandwidth successfully allocated divided by units of bandwidth requested. As expected, the performance of the algorithms degraded as the interval between requests approached zero. The bandwidth acceptance ratio increased as the interval between requests increased. After the interval between requests became sufficiently large, however, increasing the interval further had little effect. Also as expected, bandwidth acceptance ratio tended to increase more quickly with request interval size on large graphs then on very small graphs. An explanation for this result is that large graphs provide more possibilities for path choices, which produces greater likelihood of a successful path choice.

Simulation data was recorded for two algorithms: the Random Based on Hops (RBH) algorithm, and the Shortest-shortest Path (SSP) algorithm. In these runs, the effect of the randomization of path selection was being tested, so the only difference between the two algorithms was the method of selecting a next hop. The shortest path algorithm always selected the shortest of all possible paths, while the randomized algorithm used weighted random selection. Both algorithms had accurate knowledge of the state of the network. It was expected that the random algorithm would perform better since it would balance load across the network and less request

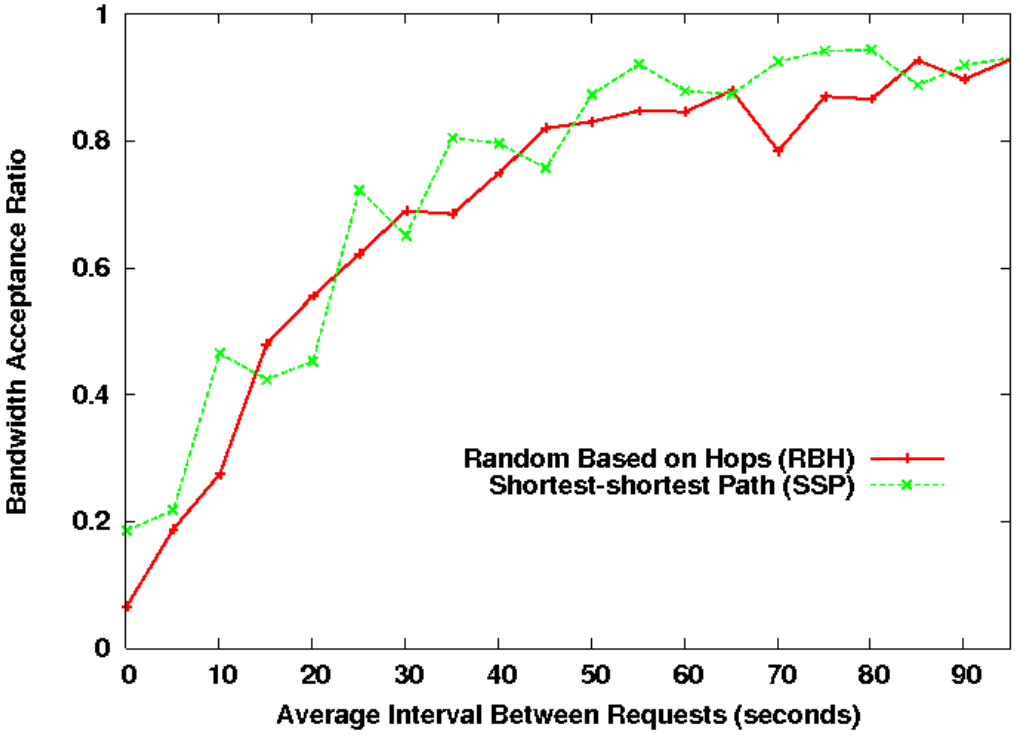
failures would occur due to insufficient bandwidth. Our results show that the performances of RBH and SSP were very similar. One possible reason why the randomized algorithm did not produce a significant improvement in our simulation could be the topology of the randomly generated graphs. If one anycast destination is usually significantly closer to the source than the others, the path chosen by the SSP and RBH algorithms would frequently be the same. One would expect that the most benefit would be gained from a randomized algorithm when there are a number of possible paths of approximately equal length from which to choose. In such a situation, the SSP algorithm would always choose the same path, while the randomized algorithm would distribute traffic across several paths. A method of testing this idea would be to manually construct a graph with anycast destination nodes, then to manually select source nodes that are approximately the same distance from each anycast node. Although this setup seems very contrived, such a study might yield information that would be useful to companies who want to find the most beneficial locations for their replicated servers.

We have not yet tested algorithms with inaccurate link-state information. Our next step will be to compare the performance of the Random Based on Hops and Safety (RBHS) algorithm to the performance of a shortest path algorithm that acts on inaccurate link-state information. Some data resulting from the simulation runs can be seen at the end of this paper.

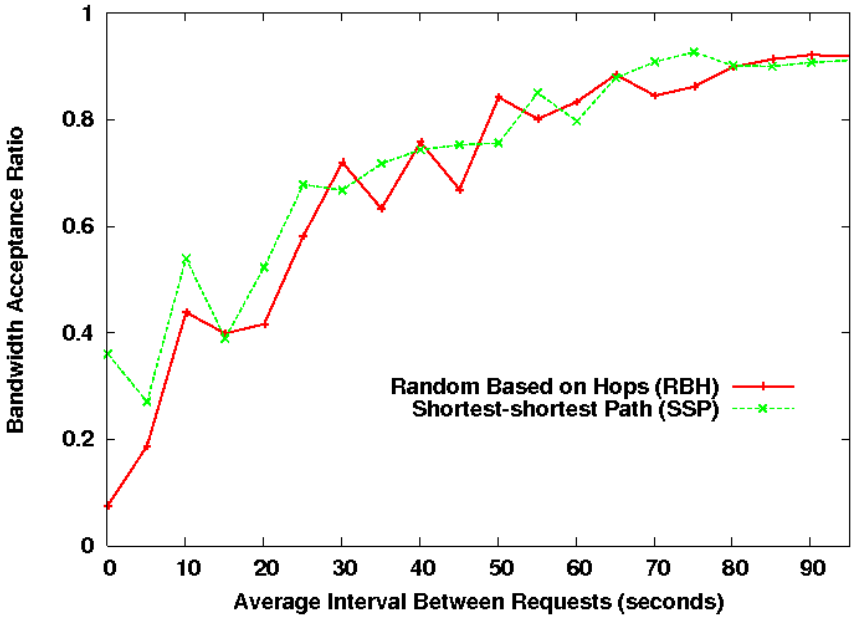
Summary

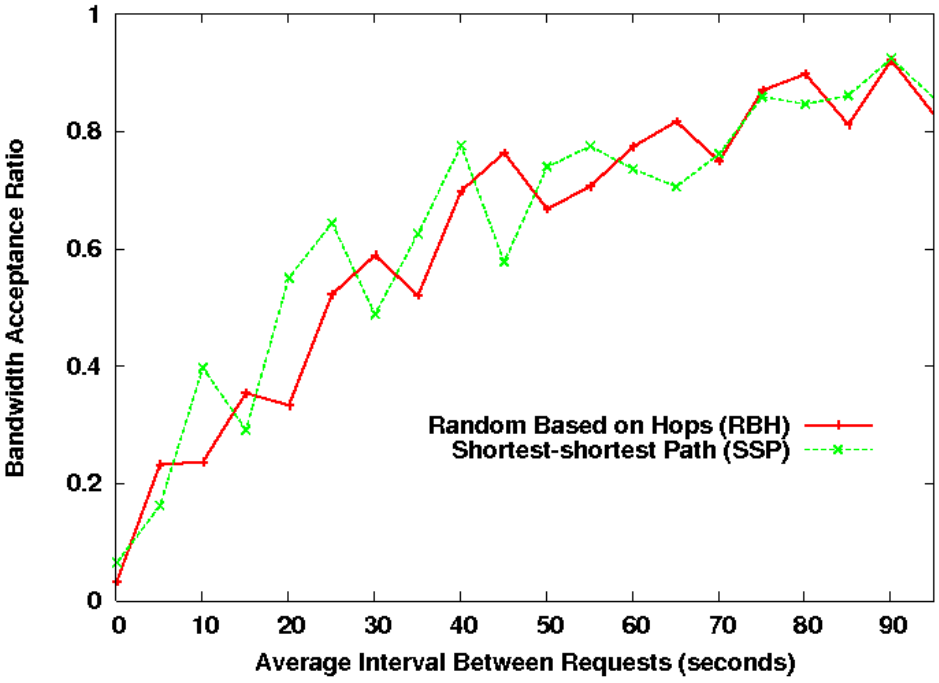
Future research could be done in the area of anycast routing with QoS using application-layer anycasting. Because of the many parameters involved in anycast routing with QoS, the problem might lend itself well to optimization techniques, such as linear programming. The centralization of application-layer anycast also makes it possible to consider other factors that could improve overall performance. For instance, in some cases the resolver might want to direct requests to the anycast destination having a path with the highest bandwidth. If the available bandwidth is sufficiently large, however, seeking even higher bandwidth would not have a significant effect on performance. In such a situation, the anycast resolver could shift priority to a second metric, a technique which would be difficult to accomplish at the network layer.

Effective anycast routing algorithms are becoming more important as the rapid expansion of the Internet leads to increased use of replicated servers and other anycast services. At the same time, the popularization of real-time multimedia applications has led to a need for QoS to maintain desired levels of quality. Our research focuses on randomized anycast algorithms with QoS. We also consider techniques that can improve anycast routing performance under inaccurate link-state information. In our simulation, the performance of our randomized algorithm with QoS was very similar to the performance of a non-randomized shortest-path algorithm. In addition to an algorithm that makes allowances for inaccurate link-state information, our future plans also include application layer anycasting, where optimization techniques can be applied to the anycast QoS problem.

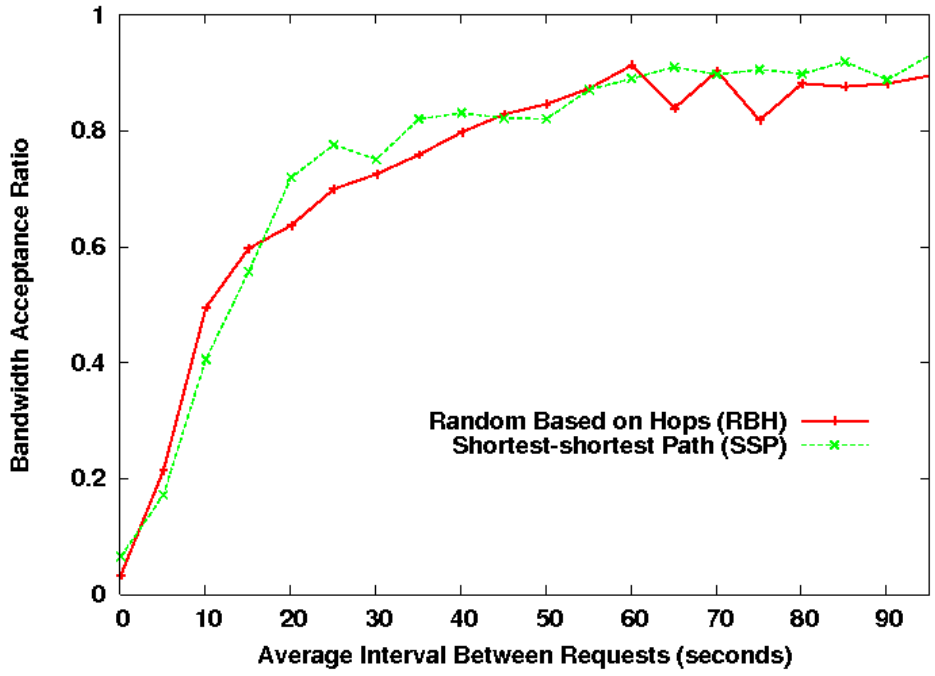


Above and Below: Simulation results for two different BRITE graphs, each with 10 nodes.





Above and below: Runs of the simulation on two different BRITE graphs, each with 1000 nodes.



REFERENCES

[1] Dong Xuan, Weijia Jia, Wei Zhao, Hongwen Zhu, "A Routing Protocol for Anycast Messages", *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, VOL. 11, NO. 6, JUNE 2000.

[2] Weijia Jia, Dong Xuan, Wanqing Tu, Lidong Lin, and Wei Zhao, "Distributed Admission Control for Anycast Flows", *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, VOL. 15, NO. 6, JUNE 2004.

[3] Ellen W. Zegura, Mostafa H. Ammar, Zongming Fei, and Samrat Bhattacharjee, " Application-Layer Anycasting: A Server Selection Architecture and Use in a Replicated Web Service", *IEEE/ACM TRANSACTIONS ON NETWORKING*, VOL. 8, NO. 4, AUGUST 2000.

[4] D. H. Lorenz and A. Orda, "QoS Routing in Networks with Uncertain Parameters," *IEEE/ACM Trans. Networking*, VOL. 6. pp. 768-778, Dec. 1998.

[5] J. X. Wang, W. P. Wang, J. E. Chen and S. Q. Chen, "A Randomized QoS Routing Algorithm On Networks with Inaccurate Link State Information", *Proc. WCC2000,* Aug. 2000.

[6] Jianxin Wang, Jianer Chen, Songqiao Chen, Weiping Wang, "QoS Routing Algorithms for Anycast Services", work supported by China National Science Fund for Oversea Distinguished Young Scholars (No. 69928201), Foundation for University Key Teacher by the Ministry of Education and ChangJiang Scholar Reward Project.