QoS Routing for Anycast Communications: Motivation and an Architecture for DiffServ Networks*

Fang Hao
Bell Laboratories
Holmdel, NJ 07733
fangh@research.bell-labs.com

Ellen W. Zegura *Mostafa H. Ammar*
Networking and Telecommunications Group
College of Computing, Georgia Tech, Atlanta, GA 30332
{ewz,ammar}@cc.gatech.edu

Abstract—
Much research in QoS routing has focused on the unicast communication paradigm. However, another communication paradigm — *anycasting* — has grown in importance in recent years. In the anycast paradigm, one endpoint is specified and the other is selected from a set of "equivalent" endpoints. An anycast service in a QoS-based network requires selection of both an endpoint and a path that satisfies a QoS constraint. This paper provides background material on anycasting in both best-effort networks and QoS-based networks. We then focus on the differentiated services QoS model, and present the design of a QoS-based anycasting architecture consistent with that model. Via simulations, we explore the design options implied by the architecture, including signalling protocols, server selection/sorting algorithms and resource reservation granularity. We conclude by highlighting open problems in the emerging area of QoS-based anycasting.

Keywords—**anycast, QoS routing, server selection, scalability**

I. Introduction

Much research in QoS routing has focused on the unicast communication paradigm, where the route is chosen between a specified pair of endpoints. This is understandable: unicast routing is relatively well understood in best-effort networks and is the dominant form of communication by current applications. However, another communication paradigm — *anycasting* — has grown in importance in recent years. In the anycast paradigm, one endpoint is specified and the other is selected from a set of "equivalent" endpoints. This paradigm is appropriate, for example, when sending a request to any one of a set of replicated web servers with equivalent content. Indeed, the terms anycasting and server selection are often used interchangeably.

To achieve scalability, Internet services are increasingly using replication, in which multiple instances of a single service are distributed either locally (in a server farm) or globally (using wide-area replication). The growth of replicated services has motivated considerable research in best-effort anycasting/server selection [6], [14], [3], [12], [15], as well as application-layer deployment, usually via the DNS resolution process (e.g., in the Akamai content distribution network). Extrapolating from the trends in best effort networks, we expect anycasting to be an important paradigm in future QoS-based networks, necessitating increased research and deployment efforts.

An anycast service in a QoS-based network can be defined as follows:

> *Given a client request for service from a set of equivalent servers, select a server and server-client path that satisfies the client QoS constraint. The client request will consist of (at least) a name for the desired service that identifies the equivalent servers and a QoS requirement. The server will be selected from among the set of servers that offer the desired service.*

Best-effort anycasting can leverage the "best-effort" nature of the service to provide a scalable solution that usually (but generally not always) provides good performance (e.g., [6], [14], [3], [12], [15]). QoS-based anycasting faces additional challenges, since the service inherently makes a guarantee regarding performance. Unlike in best-effort networks, offering an anycasting mechanism in QoS networks may require explicit support from the network layer since we have to consider the path QoS constraints. Hence we believe it is important to take the anycasting mechanism into consideration in the design and implementation of QoS architectures, to ensure efficient support. Furthermore, there is always a scalability concern associated with QoS networks. Any new mechanisms proposed for QoS-based anycasting should not incur much additional overhead. For instance, it is generally not desirable to make server selection and resource reservation on a per-flow basis.

This paper provides background material on anycasting in best-effort networks, as well as a review of the limited work on QoS-based anycasting that has been part of ATM PNNI standardization [1]. An understanding of this background, as provided in Section 2, forms the foundation for research in QoS-based anycasting. We then turn in Section III to the differentiated services (diff-serv) QoS environment. Our interest in diff-serv networks derives from their potential to achieve scalability by dealing with aggregates of flows and offering a small set of different service categories. We review the diff-serv model with emphasis on the features relevant to anycasting.

With background on anycasting and diff-serv, we then present in Section IV the design of a QoS-based anycasting architecture. The architecture is scalable in the sense that server selection and resource reservation are done in an aggregated fashion and operate in the background, rather than being driven by individual client demand. At the same time, the architecture offers fast response time to the client requests for server selection. We explore the design space implied by the architecture and evaluate various design options including signalling protocols, server selection/sorting algorithms and resource reservation granularity via

* This work was supported by the DARPA ActiveCast project F30602-99-1-0514, NSF grant ANI-9973115, Sprint, and BellSouth research funding. The work was done during the first author's graduate study at Georgia Tech.

simulations. Our results are summarized in Section V. We find, for example, that "backward" signalling, from clients towards servers, is much more efficient than "forward" signalling, from servers to clients. However, backward signalling requires additional support from bandwidth brokers placed in the network, and hence adds complexity to the bandwidth broker signalling protocols.

We conclude the paper with a list of open problems in the emerging area of QoS-based anycasting.

II. Background: Anycasting

Most research in anycasting and server selection has been done in the context of best-effort networks such as the current Internet. Very little attention has been paid to anycasting in QoS networks, with the exception of the ATM PNNI specification [1]. In this section, we first review the research of anycasting in best-effort networks, and then explain the anycasting mechanism defined in PNNI protocol.

A. Anycasting in best-effort networks

The simplest server selection approach is to assign servers statically, i.e., each client uses a *pre-determined* server. For instance, Domain Name Service (DNS) servers and Usenet news servers for clients are statically configured. DNS can also be enhanced to support mapping a service to a server inside a domain. It can approximately balance the load among servers by making selections randomly or in a round robin fashion. Such simple approaches typically try to locate any server that can provide the service, regardless of server performance.

More sophisticated dynamic server selection techniques have also been proposed with the goal of selecting the "best" server based on performance metrics such as response time or server load [6], [14], [3], [12]. Such techniques typically select servers based on server and/or path characteristics that are dynamically measured.

Server selection can be supported at either the network layer or the application layer. RFC 1546 proposes a network layer anycast service that can be used for service location and auto-configuration. Recently, Katabi and Wroclawski present a framework for Global IP-Anycast (GIA) [9], intended to provide global and scalable support for network layer anycast. Network layer anycasting typically does not consider server characteristics such as the server load.

Zegura et al. have proposed a detailed application-layer anycasting architecture, including a novel server selection mechanism that can be used for Web servers [15]. Their architecture considers both server and path metrics. Designated "resolvers" are used to collect both path and server information and to select an appropriate server for clients. We will use the resolver-based architecture as a starting point for our architecture.

All the above studies have been conducted in the context of best-effort networks. The major goal is to select a server (together with the path) with the best "performance" (e.g., with shortest expected response time). Our study differs fundamentally from the earlier work due to the focus on QoS networks. Our primary goal is to select a server and path that satisfy the client QoS constraint and optimize network resource usage. Furthermore, scalability is a more challenging issue in our study because of the overhead in QoS network operations.

B. PNNI anycasting

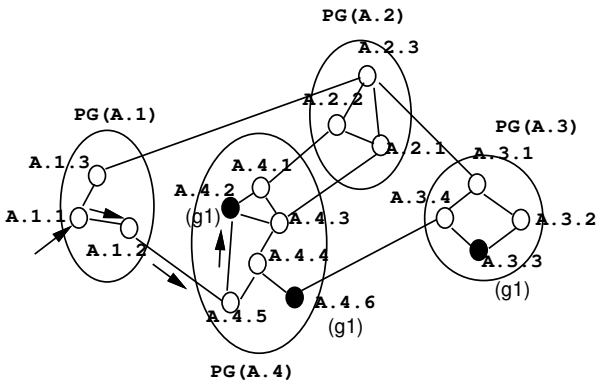


Fig. 1. PNNI anycasting

PNNI is a QoS-based routing and signalling protocol in ATM networks [1], used to set up a virtual circuit that meets the user end-to-end QoS requirements. We describe the anycast service provided in PNNI in detail here, given its close relevance to our study. PNNI defines anycast service at the network/link layer, as part of its routing architecture. Each server that provides the same service participates in the same anycast group by declaring that it can *reach* the anycast group address. The *reachability information*, which indicates the locations where the end-system and group addresses can be reached, is flooded throughout the network. When a user needs to request a service, it simply sends out its request across the user-network interface (UNI) to a directly connected switch, indicating its QoS requirements and the anycast group address. The switch then selects a server based on reachability information. The rest of the operations are similar to unicast requests: the switch first selects a path to the server based on its routing database and the user QoS requirement, then sends out a connection setup request along the path. The connection is successfully set up if every network component on the path and the server can indeed satisfy the QoS requirement. Note that PNNI does not specify how to select a server or path.

PNNI adopts hierarchical routing. Switches are grouped into *peer groups*, and peer groups are further grouped into higher level peer groups. A peer group at a higher level covers a larger range of the network than a lower level. Detailed routing information is advertised within the peer group; and only aggregated information is advertised outside.

Figure 1 illustrates the PNNI anycast signalling process. Switches, shown as small circles, are grouped into four peer groups PG(A.1), PG(A.2), PG(A.3), and PG(A.4), shown as large circles. End-systems are attached to switches and are not shown. Assume end-systems connected to switches A.4.2, A.4.6, and A.3.3 claim to be members of anycast group g1; and hence these switches advertise that they can reach g1. Consequently, PG(A.3) and PG(A.4) also advertise to other peer groups that they can reach g1. If switch A.1.1 receives a connection request for g1, it will select a peer group that can reach g1, say, PG(A.4), and send the setup message along the path. Similar to

PNNI unicast routing, each entry switch makes routing decisions inside a peer group. For instance, A.4.5 may select A.4.2 as the destination and set up a path to A.4.2.

The major constraints of PNNI anycasting are as follows: First, PNNI supports routing and anycasting based on flooding of network state and reachability information, which consumes significant network resources. Furthermore, the operations are supported on a per-flow (or connection) basis, and hence the overhead increases with the number of flows. Both reasons lead to scalability concerns with the protocol. Second, PNNI does not consider server QoS information. A switch can only select a server based on reachability information, without knowing whether the server can satisfy the client QoS constraints or not. Hence the selected server may not sustain the user QoS requirement, especially when server capacity is less adequate than network capacity.

Our work considers server selection in a diff-serv-capable Internet, which does not particularly lend itself to the ATM-style mechanisms used in PNNI anycasting. Furthermore, we consider both path and server QoS information.

III. Differentiated Service Networks

To achieve QoS, additions to best-effort IP service in the form of *integrated* [13] and *differentiated services* [2] are under development. Integrated services (int-serv) can potentially offer guaranteed end-to-end QoS on a per-flow basis, at the cost of complexity in protocol processing and state requirements. Differentiated services (diff-serv) achieve scalability by dealing with aggregates of flows, and seem to be a more promising QoS technology in large scale networks. This section provides a brief overview of differentiated services, and highlights the most important assumptions with respect to our work.

Diff-serv networks distinguish between *edge* routers and *core* routers of a domain. Edge routers are connected to end hosts or routers of other domains. A core router is solely connected to routers in its own domain. Since core routers generally see many more flows than edge routers, a diff-serv network implements only simple operations in the core routers and pushes more complex operations to the edge routers. When data packets enter a diff-serv domain, they are classified, marked, shaped, and policed in the edge routers, typically on a per-user-flow basis. The packets that pass through the edge routers are marked as certain flow aggregates, each corresponding to a *per-hop behavior* (PHB). The PHB defines how the packets should be forwarded. The core routers treat the packets on a PHB basis, i.e., the packets sharing the same PHBs are treated in the same way. Only a small number of PHBs are to be supported in the core routers.

QoS can be guaranteed in a diff-serv network domain if the resources are appropriately provisioned. The resources are managed based on contracts between the neighboring diff-serv domains, called *service level agreements* (SLAs). The diff-serv domains configure their edge and core routers based on the SLAs to make sure enough resources are provisioned for the diff-serv traffic. End-to-end QoS can be ensured by concatenation of the SLAs between all neighboring domains on the path.

Many details regarding the realization of a diff-serv network remain open and are the subject of debate within the IETF and elsewhere. Therefore, we make a set of assumptions about the basic operation, which are consistent with the current proposals for diff-serv operation. Specifically, we assume:

(1) The resources in each diff-serv domain are managed by an entity called a *bandwidth broker* (BB) [10]. The BB is responsible for negotiation with BBs in the neighboring domains, and resource management in its own domain. BBs in different domains typically do not share resource availability information, but they can communicate with each other to reserve resources by using signalling. One example of such a signalling protocol is the SIBBS protocol defined by the Internet2 QBone Bandwidth Broker Advisory Council[1].

(2) SLAs define the maximum amount of resources that can be allocated by one domain on behalf of another domain. SLAs are relatively static, i.e., they change on a long time-scale (weeks or months). A particular portion of the resources within the range of the SLA must be explicitly allocated before any data can actually use the service. Such resource allocation can occur dynamically, e.g., on a time-scale of several minutes or hours.

(3) Resource allocation is typically done end-to-end between all diff-serv domains along the data path to insure end-to-end QoS for the aggregate flow between two end domains.

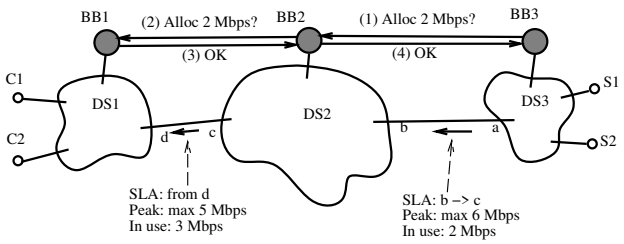


Fig. 2. Resource allocation in diff-serv networks

Figure 2 shows an example of the resource allocation process. Suppose diff-serv domains DS3 and DS2 have the following SLA: DS2 provides a virtual wire (VW) service[2] for DS3 from ingress point b to egress point c with peak rate of 6 Mbps. Also assume that 2 Mbps has already been allocated. Similarly, assume DS1 provides VW for DS2 from ingress point d to the hosts in DS1 with peak rate of 5 Mbps; assume 3 Mbps has already been allocated. Suppose that it is determined that some collection of hosts in DS3 need to send data to hosts in DS1 at a rate of 2 Mbps. The bandwidth brokers in each domain are responsible for allocating the resources, as indicated by the ordered message exchange between BBs. BB3 will send a request message along the data path to BB2, and then to BB1. If resources are available in each domain, a confirmation will be sent back, and the corresponding resources will be allocated; otherwise, a rejection message will be returned to BB3. Note the allocated resources can be adjusted by a similar signalling process if the user demand changes in the future.

[1] http://qbone.internet2.edu/bb/

[2] VW [11] intends to emulate a conventional leased line service in diff-serv networks, by using the expedited forwarding (EF) PHB [8]. It guarantees to deliver data with a specified peak rate, low loss, low latency, and low jitter. Here we use VW as an example, but our anycast architecture design does not depend on any particular service.

IV. Architecture Design

A. Overview

Our goal is to design an architecture to support QoS-based server selection in diff-serv networks. We intend to provide end-to-end QoS guarantees to anycast clients. Hence we need to discover the available resources on the paths from the servers to the clients and make a reservation accordingly. However, the BBs in different domains typically do not share resource availability information; and no entity in the network maintains global network state information. This implies that we have to rely on signalling between BBs for resource discovery and reservation. However, it is not desirable to send reservation messages to BBs on per-flow basis, since that approach will dramatically increase the signalling overhead and thus compromise the scalability of the diff-serv network. Instead, we propose to reserve resources in an aggregated fashion. More details of the architecture are described next. Note that we focus on bandwidth requirement in this design; but it should be possible to extend the architecture to also accommodate other QoS metrics.

Our design is based on two types of entities: *resolvers* and *server agents*. There is a resolver in each domain that has anycast clients, following a similar approach to the best-effort service architecture of Zegura et al. [15]. An entire anycast group shares a single server agent; one server agent may act on behalf of many anycast groups. The server selection process is fairly simple from a client's perspective: When a client needs to access the service, it contacts the resolver in its own domain and asks for a server address. The resolver replies to the client with a server address that can meet the client QoS requirements. Then the client sends a request to the server, and the server starts delivering data.

To enable the resolver to make appropriate server selection decision, additional background operations are necessary. In the background, the resolver predicts the client demand in the domain, and contacts the server agent and/or BBs to make server selection decisions and resource reservation. The server agent collects dynamic QoS information from all individual servers. This can be done by letting the servers "push" the resource update information to the server agent whenever there is a significant change in the server load. Based on the server resource information and the resolver demand, the server agent can select a list of candidate server domains that have servers with available capacity, and then assist the resolver in server selection and resource reservation. Although the resolver selects one server for each individual client request, it may reserve resources from multiple servers (potentially in multiple server domains) for all its clients in the anycast group, so that different clients may be assigned to different servers. Multiple server domain reservation is more flexible than single domain reservation, but is also more expensive in terms of signalling cost, as we indicate in Section V.

In summary, the server selection is accomplished in two steps: first, the server agent selects servers that have available capacity; and second, among the selected servers, the server agent, resolver, and BBs work jointly to select the servers with enough resources on their paths to the client[3].

[3]In general, both server capacity and network resources can dynamically change and become a bottleneck, thus both steps are necessary.

After the reservation process, the resolver has a list of selected servers, including both server addresses and the amount of resources that are available from each server. The resolver monitors the utilization of the reserved resources and adjusts the reservation accordingly by sending new signalling messages.

There are four major issues in this architecture design: (1) client demand prediction; (2) signalling protocol between the resolver, server agent, and BB; (3) server selection / sorting algorithms; and (4) resource reservation granularity.

We assume the resolver predicts the aggregated demand of the clients in its domain. Prediction algorithms that have been used in the context of resource provisioning, such as the local maximum or local Gaussian predictor discussed in [5], are also applicable here. The prediction algorithms are relatively orthogonal to the other three design issues in the architecture, and we focus on the other issues in the rest of this study.

B. Signalling protocol

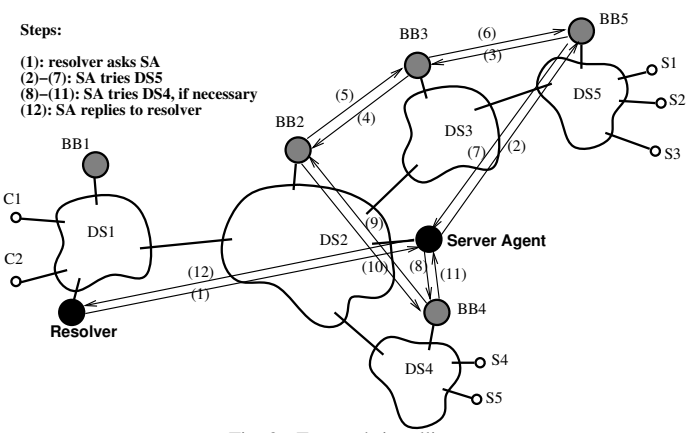


Fig. 3. Forward signalling

Based on its demand prediction algorithm, the resolver is triggered to reserve resources from the server domains to the client domain. Resource reservation is done by signalling between resolver, server agent and BBs. We consider two alternative signalling directions: *forward* (from the server domains to the client domain) and *backward* (from the client domain to the server domains).

In *forward signalling*, the resolver asks the server agent to select server domains and reserve resources. The server agent first selects a list of candidate server domains where servers have resources available, and then tries each domain in the list sequentially by contacting the BB in each server domain.

Figure 3 shows an example of forward signalling. Suppose the server agent selects both DS5 and DS4 as candidate server domains. The server agent first contacts BB5, which triggers signalling messages between neighboring BBs on the path to the client domain (i.e., BB5-BB3-BB2). If all the domains have available resources, a confirmation message will come back to the server agent, and resources will be allocated along the path. Otherwise, the server agent can contact BB4, and try to reserve resources from DS4. Similarly, signalling occurs down the data path towards the client domain (i.e., BB4-BB2). The server agent replies back to the resolver after either resources are reserved, or all candidate server domains are tried. If reservation is successful, the reply message will contain the list of selected

server domains and corresponding resources.

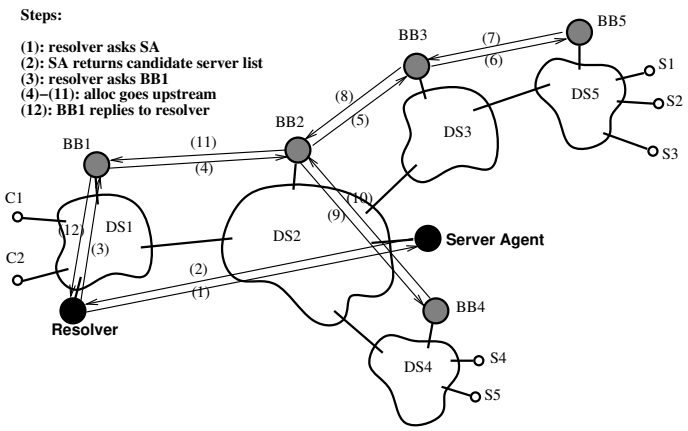


Fig. 4. Backward signalling

In *backward signalling*, the resolver asks the server agent for a list of candidate server domains where servers have resources available. The resolver then asks the BB in its own domain to reserve resources from one or more of the candidate server domains.

Figure 4 shows an example of backward signalling. Suppose the resolver receives the candidate server domain list (DS4, DS5) from the server agent resolver then contacts BB1 to reserve resources from DS4 or DS5, which triggers signalling messages to the neighboring bandwidth brokers in the upstream (BB1-BB2-BB3-BB5). Note that the BBs on the path may participate in server selection. For instance, BB2 may decide which direction (to BB3 or BB4) to try first. BB2 may also decide to try another direction if the previous reservation attempt fails.

We can view the paths from all the candidate server domains to a particular client domain as a tree structure, where the client domain is the root and the server domains are leaves (Figure 5(a)). Forward signalling starts from a leaf and traverses to the root, as shown in Figure 5(b). Multiple leaves may need to be tried before a successful reservation can be made, since some of the domains on the path may not have enough resources available. Note that the server agent makes the decision on which server domain to try, and the BBs determine if resources are available. The BBs do not have to distinguish between anycast and unicast signalling; hence no significant change in the current BB signalling proposals (e.g., SIBBS) is required.

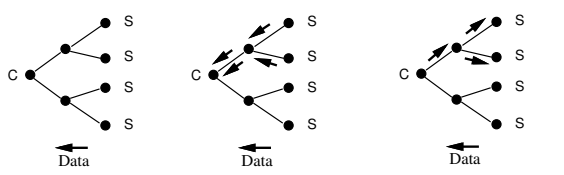


Fig. 5. Comparison of forward and backward signalling

On the other hand, backward signalling starts from the client domain (root), and traverses to the server domains (leaves), as shown in Figure 5(c). Unlike forward signalling, each BB on the path may decide which directions (or branches) to try. Backward signalling is more efficient than forward signalling for two reasons. First, the signalling messages are combined towards the root. At most one message needs to be sent on each link. Hence the total number of signalling messages is reduced. Second, backward signalling can prune infeasible links (i.e., the links with insufficient resources) faster than forward signalling. In forward signalling, pruning one infeasible link leads to pruning one path and leaf. For instance, in Figure 3, if we find link DS5-DS3 is not feasible after step (2), we do not need to do steps (3)-(6), i.e., we can prune path DS5-DS3-DS2-DS1 together with leaf DS5. But in backward signalling, pruning one infeasible link causes the whole subtree associated with the link to be pruned. For instance, in Figure 4, if we find link DS2-DS1 is not feasible after step (3), all steps (4)-(11) can be avoided, i.e., both branches DS2-DS3-DS5 and DS2-DS4 can be pruned. Faster pruning can narrow down the searching space more quickly, and hence reduce the time to reach a successful reservation.

However, backward signalling requires additional support from the BBs. The BBs need to signal upstream instead of downstream. Since the path from a sender to a receiver is, in general, different from that from the receiver to the sender (i.e., the paths are asymmetric), we also need to discover the upstream paths so that the BBs know where to send signalling messages. Hence backward signalling is somewhat more complex.

C. Server pre-selection and sorting algorithms

Server selection is accomplished in two steps. First, the server agent selects a list of candidate server domains based on the server information. Second, signalling is initiated either in the forward or the backward direction to reserve resources from one or more server domains to the client domain.

We refer to the first step as *pre-selection*. Pre-selection can be based on metrics such as available server capacity and the domain-level distance (i.e., number of domains on the path). Server capacity information is "pushed" from each individual server to the server agent. A server domain is *feasible* if it has at least one server with available capacity to support the resolver request. We study two pre-selection algorithms: (1) *unrestricted selection*, which selects all feasible server domains; and (2) *closest-only selection*, which selects all feasible server domains that are closest (i.e., with minimum domain-level hop-count) to the client domain.

In the second step, we need to decide the order in which the candidate server domains should be tried. In forward signalling, the decision is made by the server agent. In backward signalling, the BBs on the path need to decide which direction (i.e., ingress point) to try first. Note that several server domains may correspond to the same ingress point. We study the following three sorting algorithms to determine order:

- *Random*: sort the server domains or ingress points in random order.
- *Widest-first*: sort the ingress points according to the amount of resources available from the ingress point to the egress point; ingress points with more resources go first.
- *Probabilistic balancing*: sort the ingress points based on the resources available from the ingress point to the egress point. However, the order is randomized such that the ingress points with more resources are more likely to be

attempted earlier.

Random sorting is applicable for both forward and backward signalling; the other two can only be used in backward signalling.

D. Reservation granularity

In the simple case, resources are reserved from a single selected server domain to the client domain. However, for large demands, it may be better to allow the reservation to be divided between several server domains, so that the anycast clients in a domain can get service from several server domains. We define *reservation granularity* as the minimum amount of bandwidth that can be reserved from each server domain. Intuitively, a decrease in the reservation granularity makes resource reservation more flexible since the server agent or BBs have more freedom in deciding how to split the reservation among different server domains. Consequently, it may lead to better reservation results. On the downside, splitting the reservation across multiple server domains may potentially cause extra signalling overhead.

E. Further discussion

- *One versus multiple server agents.* One server agent should be sufficient to handle server load updates for a single anycast group, since the number of servers is typically less than a few thousand. An additional server agent can be used as a backup to offer more reliability. It should also be easy to extend the architecture to accommodate more servers by using a hierarchical structure of multiple server agents.
- *Discovery of reserve-path and domain-level distance.* Backward signalling requires the resolver to be aware of the domain-level path between the server and client domains. One possible way to discover the upstream paths is as follows. The BBs access the BGP routing information in their own domains. The server agent contacts the BBs in the server domains to obtain the path information from the server domains to the client domain, and then relays the path information to the resolver in the client domain. The resolver may then use the discovered upstream path information during the backward signalling process. Similarly, closest-only pre-selection algorithm IV-C requires that the server agent knows the domain-level distance between the candidate servers. This information can also be obtained through the BBs in the server domains. Note that more signalling messages are involved in both cases.
- *Client and server interfaces.* This architecture requires each client to get the server address from the resolver before accessing any server; it also requires each server to periodically report its load information to the server agent. But it is not necessary to modify the client and server application interfaces. Instead, we propose to incorporate the modification into the DNS server interface at the client side, and the layer 4-7 switch at the server farms. At the client side, the client application queries the DNS server in the traditional way; the DNS server in turn queries the resolver to obtain the server address. At the server side, since a layer 4-7 switch is typically placed in front of servers for load balancing and reliability, the switch can be enhanced to report load information to a server agent.

V. Performance Evaluation Summary

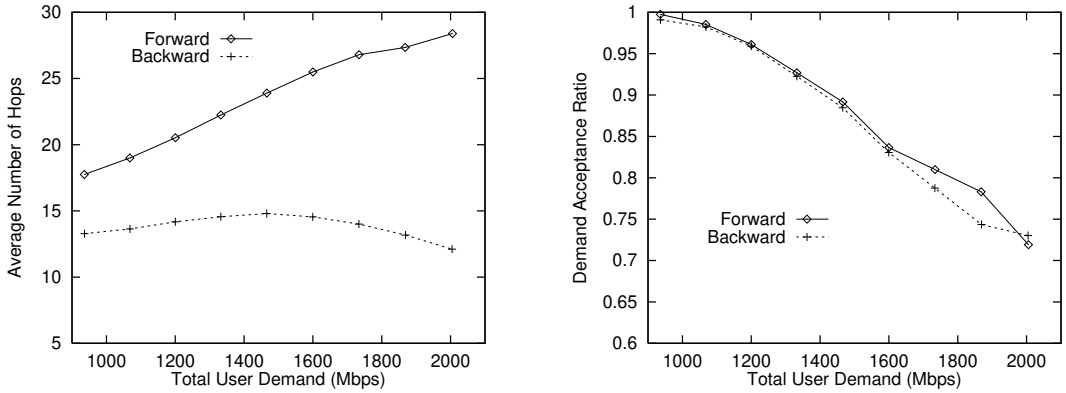
We have evaluated both the signalling overhead and reservation performance of the design options implied by the server selection architecture by using simulation. Signalling overhead indicates the amount of resources consumed by signalling messages. We measure signalling overhead by the total average number of hops that each signalling message traverses. Reservation performance is measured by the demand acceptance ratio, i.e., the total rate of the accommodated reservation requests over the total rate of all reservation requests. The simulation uses a 100-node random topology generated by GT-ITM[4], where each node represents a domain. The topology consists of 25 transit nodes and 75 stub nodes. Five stub nodes are chosen as server domain nodes; and 50 are chosen as client domain nodes. Both unicast and anycast traffic are generated between the stub nodes. In this section, we summarize the evaluation results across all experiments, and then show the results for signalling direction evaluation as a sample. More details of the simulation configuration and the performance results can be found in [7]. Our results show:

- *Signalling direction.* Backward signalling is much more efficient than forward signalling. The advantage of backward signalling is more significant under higher traffic load. However, backward signalling requires additional support from the bandwidth brokers, and hence adds complexity to the bandwidth broker signalling protocols.
- *Reservation granularity.* Smaller reservation granularity allows resources to be reserved from multiple servers, and hence leads to more flexible reservation. But smaller granularity also increases the signalling overhead. Hence it is possible to trade-off between anycast request acceptance ratio and bandwidth broker signalling overhead by adjusting the reservation granularity.
- *Pre-selection algorithms.* Unrestricted pre-selection can admit more anycast requests than closest-only pre-selection, because it tries to maximize the chance of making reservations for anycast requests. Unrestricted pre-selection also incurs more signalling overhead than closest-only; this is the price that it pays for better reservation performance.
- *Sorting algorithms.* The results from our experiments show that the difference between random, widest-first, and probabilistic balancing sorting algorithms is small. However, in general, it is not yet clear whether one can improve reservation performance by taking the advantage of the local resource availability information in each domain.

A. Sample results: evaluation of signalling directions

Figure 6 (a) shows signalling overhead under both forward and backward signalling. Unrestricted server pre-selection and random sorting are used in both cases. We experiment with different user demand levels by changing the user bandwidth demand size while keeping the ratio of unicast and anycast demands the same (about 1.5). Total user demand (horizontal axis) refers to the total user demand (including both anycast and unicast demand) averaged over time. We observe that backward signalling has

[4]http://www.cc.gatech.edu/projects/gtitm/



(a) Signalling overhead (b) Reservation performance

Fig. 6. Forward vs. backward signalling

much less overhead, especially as total demand increases. The main reason is that backward signalling tends to combine the signalling messages towards the client domain (the root of the tree) and hence reduces the total number of signalling messages, as we explained in Section IV-B.

We also find that the difference between forward and backward signalling becomes larger as the total user demand increases. With the increase in user demand, forward signalling overhead monotonically increases, but backward signalling overhead first slightly increases and then decreases. This is because two factors contribute to the change in signalling overhead. On one hand, fewer network resources are available when the user demand increases, and hence the server agent or BBs need to try more times before they can find a feasible path, causing larger overhead. On the other hand, when fewer resources are available, we can prune the infeasible paths faster since fewer links are likely to have sufficient resources, which reduces signalling overhead. Whether the signalling overhead actually increases or decreases depends on which factor is more significant. As explained in Section IV-B, the pruning effect is much more significant for backward signalling, especially when the user demand is relatively higher and fewer resources are available.

Figure 6 (b) shows the reservation performance for anycast requests. We find that forward and backward signalling have similar performance. Almost the same amount of user requests can be accommodated regardless of the signalling direction. This is because the same pre-selection and sorting algorithms are used; it does not matter whether we try from the server side or from the client side. Note that backward signalling is more flexible in the sense that more sorting algorithms can be used.

VI. Open Problems

In this paper, we have presented the background and motivation for the QoS-based anycast problem, and then shown a QoS-based anycast architecture to support server selection in differentiated service networks. Resources for the clients in a domain that request the same anycast service are aggregated and reserved together. Resources are discovered and reserved by using signalling between the bandwidth brokers of the domains on the path. We have evaluated several design issues implied by the architecture, including signalling direction, server pre-selection and sorting algorithms, and resource reservation granularity.

There are still many open issues for future research:

- *Pre-selection and sorting algorithms.* We need a better understanding of the pre-selection and server sorting algorithms. The pre-selection and sorting algorithms are in some sense similar to conventional QoS routing algorithms [4], if we assume all the servers are connected by virtual links that have infinite resources and zero cost. One major difference is that the pre-selection and sorting algorithms are not based on global information, while most conventional QoS routing algorithms are based on routing tables that essentially reflect the global network state. It will be interesting to see whether some QoS routing algorithms can be adapted to the server selection problem.
- *Interaction of path and server constraints.* Although our architecture can accommodate both server and path QoS characteristics in server selection, we have been focusing on the impact of path resource constraints in the performance evaluation. Further experiments are needed to understand the interaction between path and server resource constraints, and their impact on reservation performance.
- *Interaction with path selection methods.* There are many ways that a path can be selected in a domain, such as OSPF, policy-based configuration, QoS routing, or traffic engineering mechanisms. Our architecture does not depend on path selection methods. However, we need a better understanding of the impact of path selection methods on server selection performance.
- *Measurement-based approach.* Our architecture is based on signalling between bandwidth brokers. In doing so, we can make explicit reservations for anycast applications and make stringent end-to-end QoS guarantees. But the signalling overhead is still a concern for the scalability of the approach. Another possibility is to take a measurement-based approach without using signalling. For instance, it might be possible to measure the traffic patterns of each domain and predict the domain resource availability. Server selections can be done based on such predictions. However, the downside of this approach is that we will not be able to make hard QoS guarantees since we do not reserve resources.
- *Network-layer anycasting.* Our architecture offers an

application-layer anycasting solution. It may also be worth exploring the possibility of implementing a network-layer (IP level) QoS-based anycast mechanism, in which server and path selection is made by routers instead of application-layer entities such as gateways or resolvers. One major challenge is that path information is at the network level, but server information is at the application level. It is not immediately clear how to access application level information from the network level without impairing the integrity of the protocol layering structure.

ACKNOWLEDGEMENT

The authors thank Bethany Zegura for being nice and cooperative in the very early stage of her life.

REFERENCES

[1] ATM Forum. Private Network-Network Interface specification version 1.0. Technical Report af-pnni-0055.000, ATM Forum, March 1996.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. *RFC 2475*, December 1998.

[3] R. L. Carter and M. E. Crovella. Server selection using dynamic path characterization in wide-area networks. In *Proceedings of IEEE INFOCOM*, 1997.

[4] S. Chen and K. Nahrstedt. An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions. *IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, November/December 1998.

[5] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe. A flexible model for resource management in virtual private networks. In *Proceedings of ACM SIGCOMM*, September 1999.

[6] J. Guyton and M. Schwartz. Locating nearby copies of replicated Internet servers. In *Proceedings of ACM SIGCOMM*, 1995.

[7] F. Hao, E. W. Zegura, and M. H. Ammar. QoS-based based server selection in differentiated service networks. In *Proceedings of IEEE INFOCOM*, April 2001.

[8] V. Jacobson, K. Nichols, and K. Poduri. An expedited forwarding PHB. *RFC 2598*, June 1999.

[9] D. Katabi and J. Wroclawski. A framework for Global IP-Anycast (GIA). In *Proceedings of ACM SIGCOMM*, 2000.

[10] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the Internet. *RFC 2638*, July 1999.

[11] K. Nichols and K. Poduri. The 'virtual wire' behavior aggregate. *IETF Internet draft* <draft-ietf-diffserv-ba-vw-00.txt>, March 2000.

[12] J. Rosenberg and H. Schulzrinne. Internet telephony gateway location. In *Proceedings of IEEE INFOCOM*, 1998.

[13] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. *RFC 2212*, September 1997.

[14] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. Service location protocol. *RFC 2165*, June 1997.

[15] E. Zegura, M. Ammar, Z. Fei, and S. Bhattacharjee. Application-layer anycasting: A server selection architecture and use in a replicated web service. *IEEE/ACM Transactions on Networking*, August 2000.

BIOGRAPHIES

Fang Hao received his B.E. degree in computer science from Tsinghua University, China in 1993, and Ph.D. in computer science from Georgia Tech in 2000. He is currently a member of technical staff in the Networking Research Lab of Bell Labs, Lucent Technologies. His research interests include scalability and stability of routing protocols, content delivery networks, QoS networks, and performance evaluation. He is a member of IEEE and ACM.

Ellen W. Zegura (M 1993 / ACM 1990) received the B.S. degrees in computer science and electrical engineering (1987), the M.S. degree in computer science (1990) and the D.Sc. degree in computer science (1993) all from Washington University, St. Louis, Missouri. She is currently an Associate Professor and Assistant Dean in the College of Computing at Georgia Institute of Technology. Her research interests include active networking, server selection, anycast and multicast routing, and modeling large-scale internetworks. Dr. Zegura is currently an editor for IEEE/ACM Transactions on Networking.

Mostafa Ammar is a Professor with the College of Computing at Georgia Tech. He received the S.B. and S.M. degrees from the Massachusetts Institute of Technology in 1978 and 1980, respectively and the Ph.D. in Electrical Engineering from the University of Waterloo, Ontario, Canada in 1985. For the years 1980-82 he worked at Bell-Northern Research. His research interests are computer network architectures and protocols, distributed computing systems, and performance evaluation. He currently serves as the Editor-in-Chief of the IEEE/ACM Transactions on Networking. He is the co-author of the textbook "Fundamentals of Telecommunication Networks," published by John Wiley and Sons and was the co-guest editor of April 1997 issue of the IEEE JSAC on "Network Support for Multipoint Communication." He also served as the Technical Program Co-Chair for the 1997 IEEE International Conference on Network Protocols. He is a Fellow of the IEEE, a member of the ACM.