MeGaDiP: A Wide-Area Media Gateway Discovery Protocol [⋆]

Dongyan Xu, Klara Nahrstedt, Duangdao Wichadakul [*]

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
USA

Abstract

In wide-area and heterogeneous environments, it is necessary to deploy media gateways at strategic locations within the networks, in order to deliver customized and compose-able multimedia services to individual users. In many cases, media gateways have to be dynamically discovered, because (1) a host may not know a priori all the media gateways it will possibly use, and (2) the change of host location or end-to-end resource condition can make a known media gateway no longer useful. However, current general service location mechanisms are not sufficient to perform media gateway discovery.

In this paper, we present MeGaDiP, a wide-area Media Gateway Discovery Protocol. It is based on the same basic architecture of the general service location mechanisms, and can be seen as a discovery heuristics that will be invoked first when discovering a media gateway. The key properties of MeGaDiP include: (1) awareness of *both* source and destination end hosts: any discovered media gateway will *not* create an excessively long path between the end hosts *via the gateway*; (2) resource-awareness: any discovered media gateway is likely to have sufficient end-to-end resources to perform its service; (3) return of discovery results with low latency and high validity by caching and resource validation; (4) small amount of management traffic, regardless of the number of media gateways and end hosts in the environment. Furthermore, an extension to MeGaDiP using a hierarchical architecture is proposed to improve the discovery success rate. The initial performance results from both our prototype implementation and simulation show the soundness of MeGaDiP.

Key words: Distributed Multimedia, Media Gateway, Service Discovery
PACS:

[⋆] Manuscript number: IS-01-2441-A

[*] Corresponding author: Dongyan Xu (d-xu@acm.org). Address before August

1 Introduction

To provide multimedia services in heterogeneous environments, it is necessary to deploy media gateways (also called media proxies) at strategic locations within the networks. A media gateway intercepts a multimedia stream from a source host, performs certain processing on the media data, and forwards the processed media stream to the destination host(s). The gateway, the source and destination hosts, and their services are all considered at the application level of the communication protocol stack, and they create an *overlay network* on top of the Internet. There are many reasons for using a media gateway between the source and destination hosts, such as quality adaptation, format transformation, error control, caching, and prefetching. In many cases, media gateways have to be dynamically located, because (1) a host may not know a priori all the media gateways it will possibly use, and (2) the change of host location or end-to-end resource condition can make a known media gateway no longer useful.

Most recent research work on media gateways focus on the control and management issues - for example, the initialization and termination of media gateway sessions, and the naming and description of media gateway services[1] [2]. However, the issue of media gateway discovery (also called media gateway location) has so far received less attention.

Meanwhile, mechanisms for general service location have been proposed[3][4] [5]. However, the services they consider are primarily ‘request-reply’ or ‘sink-like’ services. For the rest of this paper, we will call them *RR/SL services*, i.e. a client makes a request to a server for some service, and the server will perform the service and get back to the client. The media gateway discovery problem is different from the RR/SL service location: media gateway discovery is constrained by two end hosts: the source and the destination, instead of by one client as in the RR/SL service location. In addition, current service location mechanisms do not emphasize the performance of the service provider to be located. However, for a multimedia application, the end-to-end quality of service is very sensitive to the quality of the service provided by the media gateway. In section 2.3, we will show why the general service location mechanisms may not perform well in media gateway discovery.

In this paper, we present MeGaDiP, a wide-area Media Gateway Discovery Protocol. Based on the same basic architecture, MeGaDiP is an easy extension to the general service location mechanisms, which are in the process of standardization. In fact, MeGaDiP can be seen as a discovery heuristics that

12, 2001: 3315DCL, Computer Science, 1304 W. Springfield Avenue, Urbana, IL 61801, USA. Address since August 12, 2001: 1398 Computer Science Building, W. Lafayette, IN 47907-1398, USA.

will be first invoked for discovering a media gateway. If MeGaDiP fails to find a qualified media gateway, then the system can fall back to the general service location mechanisms.

The key properties of MeGaDiP are the following: (1) it is aware of *both* end hosts: any discovered media gateway will *not* create an excessively long path between the source and destination end hosts *via the gateway*; (2) it is resource-aware: any discovered media gateway is likely to have sufficient end-to-end resources to perform its service; (3) it returns discovery results with low latency and high validity by caching and resource validation; (4) it only generates small amount of management traffic, regardless of the number of media gateways and end hosts in the environment. Furthermore, an extension to MeGaDiP using a hierarchical architecture is proposed to improve the discovery success rate. We are currently implementing MeGaDiP as part of the *multimedia service configuration* function in the 2K System[6] - a component-based and network-centric distributed operating system which supports flexible configuration and adaptive execution of distributed multimedia services. We target a wide-area environment with client and resource heterogeneity. In such an environment, multimedia services are *dynamically composed* by choosing the appropriate service components to fit the end-to-end resource condition of each individual client. The role of MeGaDiP is to dynamically discover the intermediate service components on media gateways.

The rest of the paper is organized as follows: Section 2 describes the basic architecture of the general service location mechanisms, and discusses their problems in media gateway discovery. Section 3 presents the details of MeGaDiP, and Section 4 suggests a further extension to MeGaDiP. Section 5 presents our experimental results. Section 6 discusses related work. Finally, we conclude this paper in Section 7.

2 A General Service Location Architecture

In this section we first describe the basic architecture common in the proposed general service location mechanisms. Then we will identify some problems of the general service location mechanisms when they perform media gateway discovery.

2.1 Basic Architecture

The basic architecture consists of common entities and functions found in the proposed service location mechanisms[3][5][4]. For notation consistency with

MeGaDiP, the names of the entities in this architecture may not be the same as those in the literatures on general service location mechanisms. Entities in the basic architecture include *end hosts (EHs)*, *service providers*, and *dealers*, as shown in Figure 1.

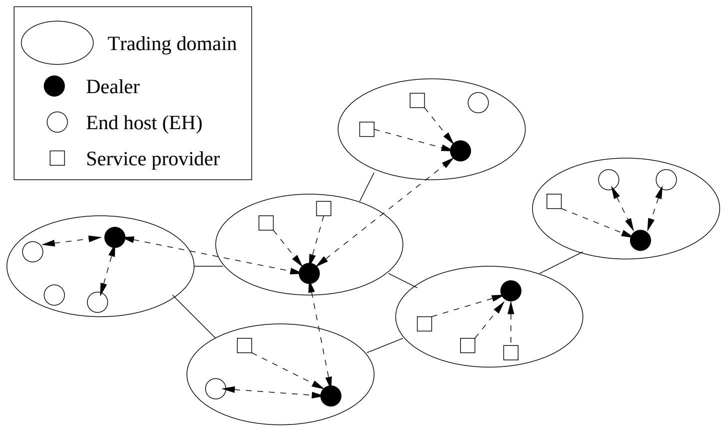

Fig. 1. Entities in the basic service location architecture

- *End hosts* (corresponding to *user agents* in the Service Location Protocol (SLP)[3]): an EH is the requester of a service. In order to find the service provider (defined below) of a requested service, the EH will query its local dealer (defined below) for information about any qualified service provider.
- *Service providers* (corresponding to *service agents* in SLP): a service provider performs and delivers a certain service to the requesting EHs. In order to advertise its service, a service provider sends *service advertisement* to its local dealer. The service advertisement is sent periodically, so that the dealer can maintain the (soft) state of the service provider.
- *Dealers* (corresponding to *directory agents* in SLP): dealers are key entities in the architecture. As the agent between EHs and service providers, a dealer accepts service advertisements from service providers, and returns information about qualified service providers to querying EHs. In the wide area environment, each dealer is associated with a *trading domain*. Hosts (including EHs and service providers) within the trading domain refer to the dealer as their *local dealer*. A service provider sends service advertisements only to its local dealer, and an EH only queries its local dealer. In order to support wide-area service location, a dealer may contact other dealers for the forwarding of service queries, or for the return of query results found in its trading domain. In the next subsection, we will describe the service location mechanisms in wide area environments.

2.2 Push and Pull Based General Service Location

Based on the basic architecture, we now describe two typical approaches in current service location mechanisms: a push-based approach and a pull-based approach:

- In the push-based approach, a dealer further pushes the service advertisements of its local service providers to other dealers. The service advertisement push is typically performed by periodic unicast or multicast. As a result, each dealer will gradually collect service advertisements of service providers outside its own trading domain. When a local EH submits a service query to the dealer, the dealer looks up the service advertisement entries of both local and non-local service providers, in order to find a qualified service provider. In [7], the service providers themselves push their service advertisements to multiple dealers via multicast channels. This is similar to (but less bandwidth efficient than) letting the dealers perform the service advertisement push.
- In the pull-based approach, when an EH queries its local dealer, if the dealer cannot find a qualified service provider based on its local service advertisements, it will call for (pull) service advertisement(s) satisfying the query condition from other dealers. The service advertisement pull from other dealers can be implemented by broadcasting the query. When a dealer with a qualified service advertisement receives the query, it will return the service advertisement to the requesting dealer. In order to save pull bandwidth, a dealer usually uses *increasing-scope multicasts* to gradually expand the searching space. Furthermore, to speed up the discovery process, each dealer will cache the service advertisements from other dealers for future use.

2.3 Problems in Media Gateway Discovery

Current service location mechanisms work well in the location of RR/SL service providers such as a web server or a printing server. However, there will be problems in the discovery of a special class of service providers: *media gateways*. In the discovery of a RR/SL service provider, the searching space can roughly be thought of as a *circle* of trading domains - the center of the circle is the local dealer of the querying EH. However, in media gateway discovery, there are two EHs involved: the source and the destination. Due to the end-to-end path constraint between the two EHs, the searching space is intuitively more like a *stripe* of trading domains, starting from the trading domain of one EH, and ending in the trading domain of the other EH. Consequently, if a general service location mechanism is used for media gateway discovery, the following problems will make it inadequate for such a purpose:

(1) It is unaware of the end-to-end path between the two EHs. In current service location mechanisms, the local dealer of one of the two EHs tries to find a qualified media gateway within a certain circle of trading domains around

the local dealer, *without* checking if the gateway deviates significantly from the path between the source and the destination EHs. For example, in Figure 2, to find a media gateway between a source EH in Denver, Colorado and a destination EH in New York City, the local dealer of the EH in Denver performs the discovery. It is possible that the result will be a media gateway G in Phoenix, Arizona, although there exists a better choice of media gateway G' in Chicago. The former choice is likely to introduce higher end-to-end delay (both in number of hops and in real-time) than the latter.

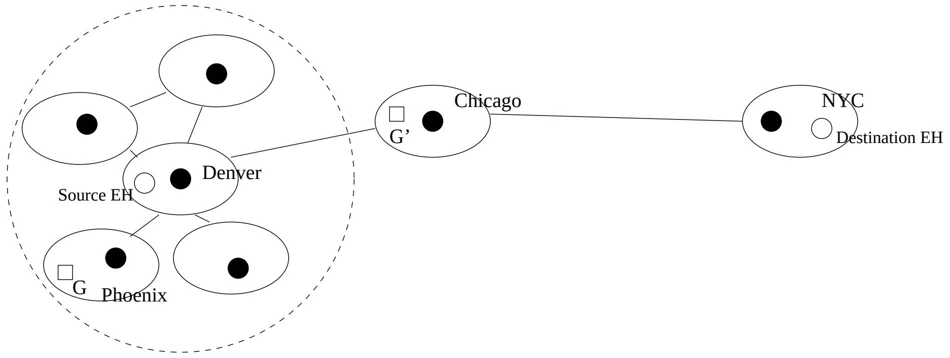

Fig. 2. The unnecessarily long path problem

(2) It is unaware of the intermediate network condition between the two EHs. It is desirable that there be sufficient network bandwidth from the source EH to the media gateway, and from the media gateway to the destination EH. However, current service location mechanisms have little awareness of the intermediate network condition between the two EHs. In Figure 3, for example, the local dealer of EH s tries to discover a media gateway with MPEG-to-H.261 transcoding service, which lowers the data rate of an MPEG video stream before crossing the network bottleneck link l. However, it is possible that the result is gateway G, which is of little use because G is 'behind' the bottleneck link l with respect to the video stream.

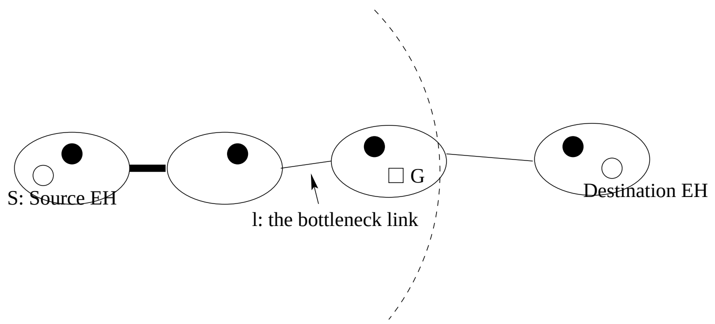

Fig. 3. The bottleneck unawareness problem

(3) It may result in an overly large number of results. For example, in Figure 4, if gateway G is found, then it is likely that many other gateways within the same searching space (as shown in Figure 4) are also returned as search results. With the expansion of searching space, the number of potential search results

will increase rapidly. This will cause confusion at the local dealer D, who does not know which media gateway is the most appropriate one to choose for the querying EH.

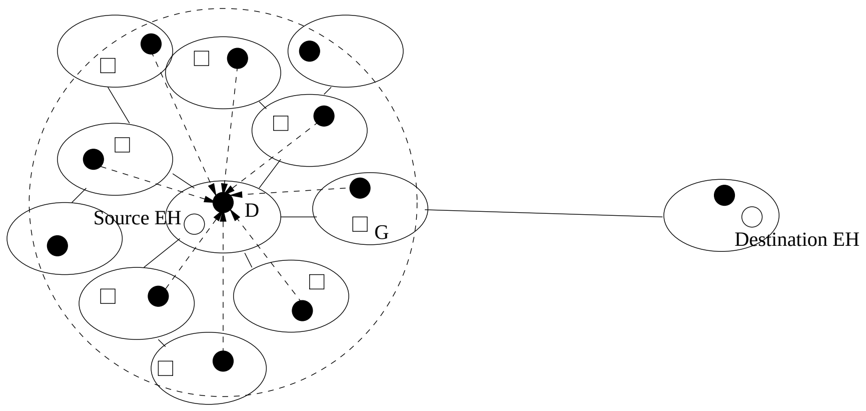

Fig. 4. The results implosion problem

(4) Finally, it may incur excessive push or pull bandwidth. A local dealer discovers qualified gateway(s) outside its trading domain by either the push-based or pull-based approach. As indicated in (3), the overly large number of search results will incur excessive push or pull bandwidth in the meantime.

3 MeGaDiP: Media Gateway Discovery Protocol

To remedy the inadequacies of general service location mechanisms, we propose a *Media Gateway Discovery Protocol*, called MeGaDiP, based on the basic architecture described in Section 2. The problem of media gateway discovery can be stated as follows: given a pair of EHs: one is the source and the other is the destination of a media stream, if it has been determined that a certain service X, residing on a media gateway, will enable the delivery of this media stream from the source EH to the destination EH via the media gateway, how to discover such a media gateway in the networks. Service type X can be determined by our multimedia Service Configuration Protocol. Details may be found in [8].

The basic idea of MeGaDiP is the following: the discovery procedure starts from the local dealer of one of the EHs, and the searching space is the stripe of trading domains along the end-to-end path between the source and destination EHs, as shown in Figure 5. A dealer in this searching space will forward the query to the next dealer, if it can not find a qualified media gateway in its own trading domain. Finally, if no qualified media gateway is found by MeGaDiP, the system can fall back to one of the general service location mechanisms. The idea of giving search priority to the stripe of trading domains along the path between the EHs is based on the heuristics that a media gateway on or

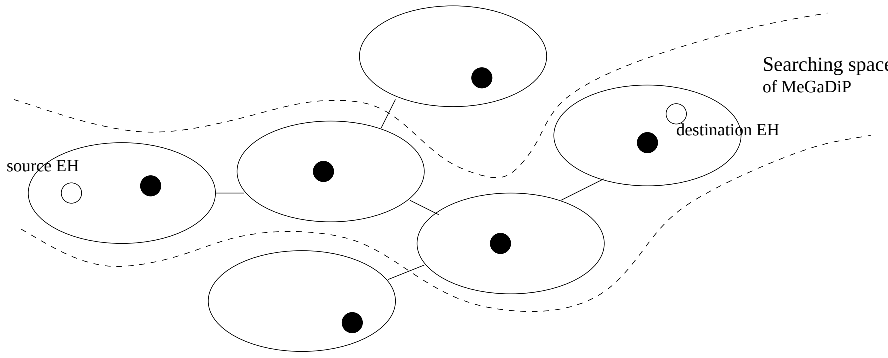

Fig. 5. Searching space of MeGaDiP

close to the end-to-end path is likely to find more bandwidth and/or to incur smaller delay (in term of hop-count and real-time) between the EHs *via the media gateway.*

This heuristics is to some extent supported by the underlying network routing mechanism. It means that network routes from the source to the destination EHs are acquired from the underlying network layer, where the network layer may use either underlying source QoS routing protocols[9], anycast protocols [10], or current Internet source routing protocols to compute the routes. If QoS routing is the underlying routing mechanism, then this heuristics is well supported because a QoS routing protocol finds a route satisfying bandwidth, and/or delay requirements along the path between source and destination EHs. Even for today's Internet routing, based on minimal number of hops, in most cases, the actual path between the EHs will not be drastically different from the optimal route with the highest bandwidth and lowest latency. Using anycast routing, multiple paths may be chosen for fault-tolerance between the source and destination EHs, and the final choice of path will depend on the resource availability along the path[1]. Key design issues of MeGaDiP are discussed in the following subsections.

3.1 Query Forwarding Order

The first issue of MeGaDiP is the order in which the stripe of along-the-path trading domains are searched. More specifically, let s and d be the source and destination EHs, and D_s and D_d be their local dealers, respectively. The question is whether the discovery procedure should (1) start from D_s, and the query be forwarded towards D_d, or (2) start from D_d, and the query be forwarded towards D_s. In MeGaDiP, this is determined by the nature of

[1] The anycast routing protocol[10] may potentially be used for media gateway selection, but not for dynamic source EH or destination EH selection, because in this paper we assume fixed source and destination EHs.

the media gateway to be discovered. We categorize media gateways into two classes, depending on the relation between a gateway's input data rate R_{in} and the output data rate R_{out}.

- Class I - $R_{in} > R_{out}$: a Class I media gateway processes an input media stream, and the corresponding output stream has a lower data rate. For example, the media gateway for transcoding Motion JPEG video to low-bit-rate H.261 video belongs to class I. A class I media gateway should be located closer to the source EH than to the destination EH, in order to save the network traffic of higher data rate (R_{in}) from s to the media gateway. Therefore, to discover a class I media gateway, the discovery will start from D_s;
- Class II - $R_{in} \leq R_{out}$: a class II gateway accepts a media stream at a data rate lower than the corresponding output stream. For example, a video prefetching gateway [11] belongs to Class II. A class II media gateway should be located closer to the destination EH than to the source EH, in order to save the network traffic of higher data rate (R_{out}) from the media gateway to d. Therefore, to discover a class II media gateway, the discovery will begin from D_d.

3.2 Along-the-Path Trading Domains

The next issue is to determine the trading domains along the path between the EHs. In the basic architecture, dealers do not have any 'routing' or 'forwarding' knowledge, i.e. a dealer does not know who is the next dealer to forward a query to, given the path from the source EH to the destination EH. In MeGaDiP, instead of implementing another routing mechanism at the dealer level, we exploit the Domain Name System (DNS) and the underlying network routing mechanism to determine the trading domains along the path.

In MeGaDiP, to find the local dealer of a host, a DNS lookup is performed. We propose a new *Resource Record type*[12] 'LD' in the DNS, which represents the local dealer address of the host. More specifically, the DNS now provides a function *DNS_LD*: given a host name h, the corresponding local dealer of h is:

$$D_h = DNS\_LD(h) \tag{1}$$

The new resource record type can be deployed incrementally by participating DNS servers over the networks.

For two EHs s and d, each of them performs a DNS lookup to find its local dealer D_s or D_d. To determine the dealers of other along-the-path trading domains, D_s first obtains a list of the intermediate routers r_i on the path from D_s to D_d. This is done by calling a simplified version of the *traceroute*[12] routine, and the route will be determined by underlying Internet routing or QoS source routing (if available) mechanism. With the list of routers, the list of intermediate dealers (D_i) between D_s and D_d can be obtained by performing DNS lookups using the LD Resource Record type, i.e. $D_i = DNS\_LD(r_i)$. If there are duplications or non-defined values in $D_1, D_2, D_3, ...$, those values will be discarded.

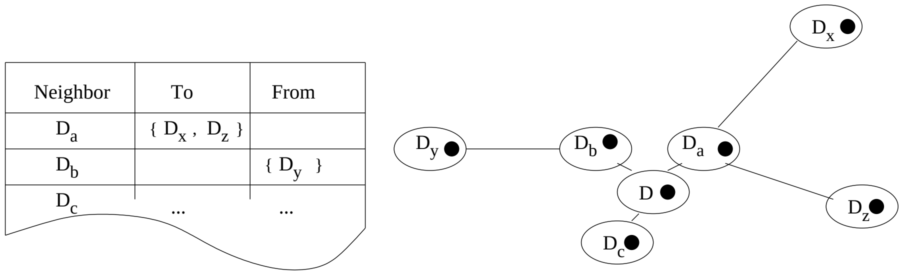

Fig. 6. DNS lookup cache for dealer D

The overhead of performing *traceroute* routines and performing DNS lookups are both non-trivial. To improve efficiency, we propose that each dealer keeps the addresses of its immediate neighbor dealers, and caches the source and destination dealers on the paths that frequently travel across this trading domain. The structure of this cache is shown in Figure 6, which is similar to a routing table. For example, for D, the next dealer towards D_x is D_a; and the previous dealer from D_y is D_b. Using this technique, the traceroute and DNS overhead can be effectively amortized.

3.3 Resource Awareness

One of the key properties of MeGaDiP is *resource awareness*. The resources include the intermediate network bandwidth and the local resources of a media gateway.

First, MeGaDiP should ensure that there is sufficient bandwidth between the source EH and the media gateway discovered, and between this media gateway and the destination EH. More specifically, to avoid the bottleneck unawareness problem shown in Figure 3, a dealer should not always forward a query to its neighbor dealer when it can not find the media gateway in its own trading domain.

In MeGaDiP, to be aware of the network condition in its neighborhood, each dealer periodically measures the bandwidth between itself and its neighbor dealers. MeGaDiP does not specify how the bandwidth is measured. In fact, there exist many tools or services for this purpose, such as Bprobe[13], Pathchar[14], SPAND[15], and IDMaps[16]. The dealer's measured bandwidth coarsely estimates network bandwidth available to EHs and gateways in the domain. This means that if the dealer does not have enough bandwidth to support multimedia flows, then media gateways in this domain will not have sufficient bandwidth to handle inter-domain multimedia traffic.

A dealer will use the measured bandwidth to decide if it should forward a query to its neighbor dealer. For example, in Figure 7, a class I media gateway is to be discovered between EHs s and d. The end-to-end bandwidth from s to d is 150Kbps (this is measured by the EHs), and the expected input and output data rates of the media gateway are 1Mbps and 128Kbps, respectively (this is specified by the EHs). The searching space by MeGaDiP consists of trading domains of dealers D_s, D_1, D_2, D_3, and D_d. Current bandwidth measurements between the dealers are also shown in the Figure. In this example, the discovery procedure should terminate at dealer D_2, because the bandwidth from D_2 to D_3 is 150Kbps, the same as the end-to-end bandwidth from s to d. This indicates a high probability that the bottleneck between s and d is encountered. Therefore, any qualified media gateway discovered by D_s, D_1, or D_2 (in that order) should be returned as query result, while the query should not be forwarded to D_3 if no qualified media gateway is found by D_2.

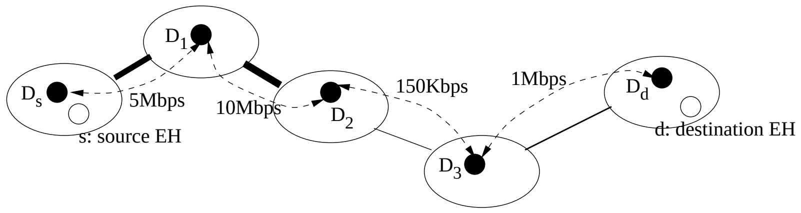

Fig. 7. An example of network condition awareness of MeGaDiP

Second, to be aware of local resources of media gateways, each media gateway in MeGaDiP reports its current resource availability in its periodic service advertisement to the local dealer. A media gateway's local resources include its (1) input bandwidth,(2) output bandwidth, and (3) other critical resources. Different media gateways may have different critical resources, for example, CPU for a video transcoding gateway, and memory and disk for a video prefetching gateway. Note that when a dealer estimates the gateways' local network bandwidth availability, the bandwidth can be either best-effort estimates in the case of best-effort Internet, or soft-guaranteed values in the case of networks with bandwidth reservation capability such as an RSVP-enabled network. On the other hand, when an EH submits a query for a media gateway, the query condition will include its estimated amounts of resources needed by the media gateway.

The dealer may potentially accumulate a large amount of resource availability information. One possible approach to control the amount of information is to apply QoS information *aggregation* as proposed in [17].

3.4 Basic Discovery Procedure

We are now ready to describe the basic discovery procedure of MeGaDiP. In the next subsection, we will propose the caching and validation of discovery results as an important enhancement to improve the performance of MeGaDiP.

(1) To discover a class I media gateway, the source EH submits a query to its local dealer D_s; to discover a class II media gateway, the destination EH sends the query to its local dealer D_d. The condition $Cond$ of the query includes the following: (a) name of service provided by the media gateway, (b) estimated input bandwidth R_{in} and output bandwidth R_{out} needed by the media gateway, (c) estimated amount(s) of other critical resource(s) needed by the media gateway, and (d) other constraints. Examples of other constraints include price range, name of service provision party, or security level etc. The query is in the form of $< D_s, D_d, Cond, gateway\_class >$.

(2.1) The dealer looks up its service advertisement entries. If there exists a service advertisement that satisfies condition *cond*, a qualified media gateway is discovered, and the result is returned to the querying EH. The discovery procedure terminates with a success.

(2.2) If there is no qualified media gateway in its trading domain, the dealer finds the next dealer D_{nxt} to forward the query, as described in Section 3.2.

(2.2.1) For a class I media gateway, D_{nxt} is the current dealer's *downstream* neighbor dealer with respect to the media stream. If the current dealer is already D_d, or the measured bandwidth *from the current dealer to* D_{nxt}, is less than R_{in}, then the current dealer reports failure to the originating dealer D_s. Otherwise, the query is forwarded to D_{nxt}, which will execute from step (2.1).

(2.2.2) Symmetrically, for a class II media gateway, D_{nxt} is the current dealer's *upstream* neighbor dealer with respect to the media stream. If the current dealer is already D_s, or the measured bandwidth *from* D_{nxt} *to the current dealer* is less than R_{out}, the current dealer reports failure to the originating dealer D_d. Otherwise, the query is forwarded to D_{nxt}, which will execute from step (2.1).

3.5 Discovery Results Caching and Validation

The basic discovery procedure in the previous subsection has a performance limitation: it will incur non-trivial latency in the setup of *every* media session, due to its distributed and wide-area nature. To speed up the discovery procedure, we propose the technique of caching and validation of discovery results in MeGaDiP.

After a media gateway is discovered, its service advertisement is returned to the originating dealer. The originating dealer will cache the returned service advertisement for future use. Each cached service advertisement will be tagged with (1) local dealer of this media gateway, and (2) local dealer of the destination EH (for a class I gateway) or source EH (for a class II gateway) involved in the discovery of this gateway. When the originating dealer receives another query about the same type of media gateway, *and* the destination EH (for a class I gateway) or source EH (for a class II gateway) has the same local dealer as (2), the originating dealer will return this cached service advertisement to the querying EH without starting the discovery procedure. Caching of discovery results significantly reduces the latency of MeGaDiP. The cache replacement policy can be the Least-Recently-Used (LRU) policy.

However, a cached discovery result needs to be validated/invalidated regarding the end-to-end resource availability, otherwise the corresponding media gateway may not be able to provide satisfactory service quality to the querying EHs. To validate local resource availability of the media gateway, the caching dealer will periodically contact the local dealer of the media gateway, obtain its current local resource availability information, and update the cached service advertisement. This validation can also be performed on-demand when the dealer receives a query about the same type of gateway.

It is more difficult to validate available end-to-end bandwidth from the source EH to the media gateway, and from the media gateway to the destination EH, in an efficient and scalable manner. Bandwidth measurement between dealers (as described in Section 3.3) only serves as an estimation of the intermediate network condition, and is used to determine when to stop the discovery process. However, it may not be used as a more precise estimation of the end-to-end 'source EH - gateway - destination EH' bandwidth. One possibility is to let every dealer periodically perform end-to-end bandwidth measurement ('dealer of source EH - gateway - dealer of destination EH') for each of the media gateways that the dealer has a cached service advertisement. However, the measurement traffic may (1) generate additional load on the media gateways and dealers, and (2) form a non-trivial amount of network traffic which may deteriorate network congestion. The situation gets worse with the increasing number of service advertisements cached by a dealer. Therefore this is not a scalable solution.

We propose a simple and scalable solution which does not introduce any additional measurement traffic. In MeGaDiP, we use the media streams themselves as the measurement traffic. During a media streaming session, we assume that the media streaming mechanism supports real-time monitoring of data rate, for example by using RTP and RTCP [18]. If the media gateway detects either of the following conditions:

- Its actual input data rate R_{in}^{act} is less than $min(R_{in}, R^{src})$ for more than T amount of time. R_{in} is the input data rate specified in the media gateway query, R^{src} is the data sending rate at the source EH, and T is a pre-defined value;
- The data receiving rate at the destination EH R^{dst} is less than $min(R_{out}, R_{out}^{act})$ for more than T amount of time. R_{out} is the output data rate specified in the media gateway query, R_{out}^{act} is the actual output data rate of the media gateway.

the media gateway will send an *invalidation message* to the local dealer of the source EH (if it is a class I gateway), or to the local dealer of the destination EH (if it is a class II gateway). The dealer, who has a cached service advertisement of this media gateway, will then invalidate and delete the advertisement, and start the discovery procedure to find a *different* qualified media gateway.

4 Hierarchy of Dealers: An Extension to MeGaDiP

In this section we briefly describe a further extension to MeGaDiP and the basic architecture. The goal is to further improve discovery success rate by expanding the searching space of MeGaDiP and to provide scalability in wide-area networks. Our basic architecture assumes a 'flat' topology of dealers, i.e. the dealers are peers to each other. Expanding the searching space will involve a large number of dealers in the discovery procedure. In order to minimize the number of dealers involved, we suggest a hierarchical architecture to organize the dealers, as shown in Figure 8 [2]. Trading domains of 'leaf' dealers are grouped into larger trading domains, with corresponding *parent dealers*. The leaf dealers periodically send service advertisement *summaries* (in order to control the volume of service advertisements sent) to their parent dealer. The number of levels in the hierarchy can be arbitrary (there are two levels in Figure 8). The address of a dealer D's parent dealer can be resolved by performing a DNS lookup of $DNS\_LD(D)$.

The basic media gateway discovery procedure can be extended as shown in Figure 8. Suppose the basic discovery procedure terminates at leaf (level-one) dealer D_3^1 without a result, then D_s^1 will forward the query up to its parent dealer D_s^2, and another discovery procedure begins at level two, and the

[2] In fact, the Internet already deploys strong hierarchical approaches in areas such as domain routing and naming. We believe that the hierarchy of dealers aligns well with the hierarchy of domains as the networks are organized.

discovery procedure will terminate at dealer D_1^2. Suppose a qualified gateway G exists in the trading domain of leaf dealer D_x^1. The level-two dealer D_1^2 will find its service advertisement summary (sent by D_x^1), and D_1^2 will then forward the query down to D_x^1, which will check if the full service advertisement satisfies the condition in the query. If so, the discovery result will be returned to D_s^1 by D_x^1.

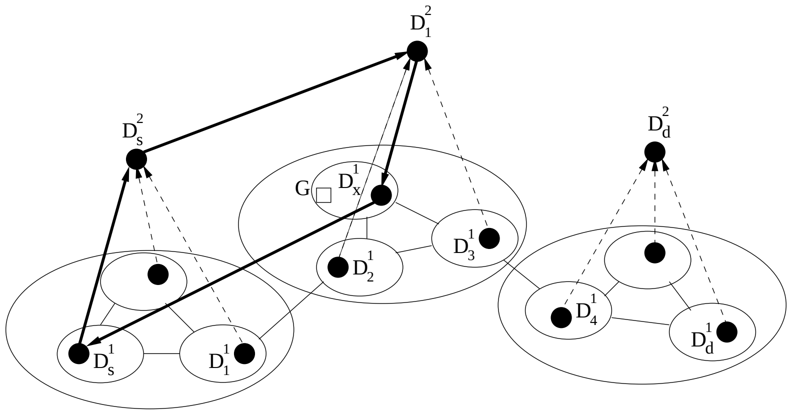

Fig. 8. Extended media gateway discovery procedure in a dealer hierarchy

With the dealer hierarchy architecture, cached service advertisements by leaf dealers may be used to answer more queries. For example, if there is a cached advertisement of a class I media gateway G at leaf dealer D_s^1, and the local leaf dealer of the destination EH is D_d^1. Then G may also satisfy (subject to the invalidation test described in Section 3.5) any query with a destination EH in the level-two trading domain of D_d^2, which is the parent of D_d^1.

5 Experimental Results

5.1 Prototype Results

We are implementing a prototype of MeGaDiP in a local testbed. It has three trading domains I, II, and III as shown in Figure 9. The dealers in these domains are D_1, D_2, D_3, respectively. In domain I, there is a stored-MPEG video server. The MPEG-1 video used in our experiment has a run time of 10 minutes and an average data rate of 800Kbps. In domain III, there are 15 video clients. For experimental purpose, each video client can only use a low-bit-rate player in bitmap format, with a maximum data rate of 128Kbps. In order to stream the video from the server to the clients, a media gateway with MPEG-to-Bitmap transcoding service is needed. We install the MPEG-to-Bitmap service on four media gateways G_1, G_2, G_3, and G_4. Each gateway has the same background workload. Network connection between domains is the 10Mbps

Ethernet. However, we only allow 1.5Mbps (by generating background traffic) between domain II and domain III. The dealers measure current inter-domain bandwidth every 30 seconds, and the bandwidth consumed by the periodic measurements is only 2Kbps. The media gateways send service advertisements to their local dealers every one minute, or when they do not have sufficient CPU resource - whichever occurs earlier.

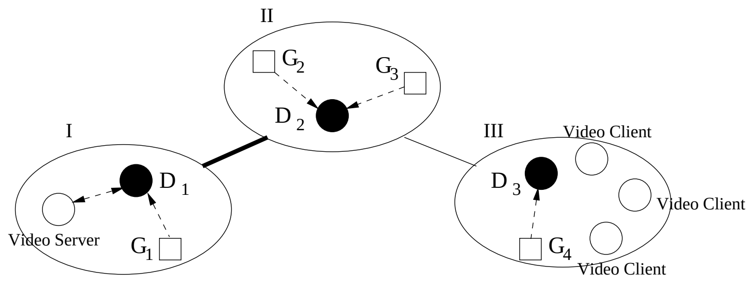

Fig. 9. The MeGaDiP prototype environment

We perform the following simple experiment and obtain some initial results: the video server submits media gateway queries for the 15 video sessions within a period of 5 minutes, and the submission times are uniformly distributed. Table 1 shows: (1) the number of video sessions that result in the discovery of G_i $(i = 1, 2, 3, 4)$ by MeGaDiP, and (2) the corresponding average query response time of MeGaDiP. The values are averaged over 10 trials.

MeGa.	No. of sessions	Response Time
G_1	4.3	20.3ms
G_2	3.5	26.6ms
G_3	3.1	27.2ms
G_4	N/A	N/A

Table 1
Initial results from prototype

The response time for the discovery of G_1 is shorter than that for the discovery of G_2 or G_3, because G_1 is in the same domain as the server, and will be discovered first by MeGaDiP. When G_1 does not have sufficient CPU [3], D_1 will forward the queries to D_2, and G_2 or G_3 will be discovered. However, later when G_2 and G_3 run out of CPU, D_2 will not forward the query to D_3, because D_2 detects that the current bandwidth from domain II to III is not sufficient for a source MPEG stream to get through. Our initial results illustrate MeGaDiP's resource awareness. In addition, we notice that the response time for the discovery of G_2 or G_3 is not significantly longer than that of G_1, due to the caching of service advertisements of G_2 and G_3 in D_1.

[3] Our earlier work of CPU Broker [19] is responsible for detecting this.

The local testbed is not adequate to evaluate the performance of MeGaDiP in a wide-area environment. Before the deployment of MeGaDiP on a larger scale, we study its performance by simulation. The trading domains are shown in Figure 10. For simplicity, we assume that there are three hops between any two hosts in neighboring domains. We will focus on discovery overhead, rather than resource awareness of MeGaDiP. Therefore, in this subsection we assume that the resources are always sufficient for any media gateway discovered.

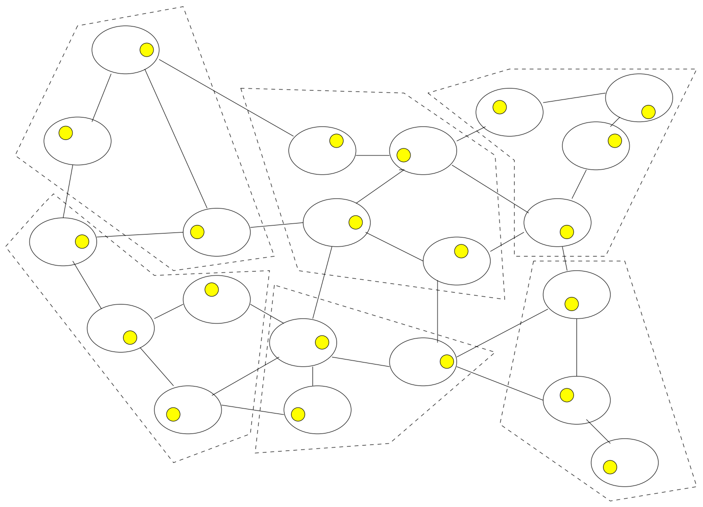

Fig. 10. The MeGaDiP simulation environment

In our first experiment, we deploy 8 class I media gateways with a transcoding service in randomly selected trading domains in the environment. We also randomly generate 200 pairs of <source EH, destination EH> in the environment. The number of hops between each pair of EHs is a multiple of 3. Then we use MeGaDiP, a push-based approach, and a pull-based approach to discover the media gateway between each pair of EHs, respectively. In the push-based or pull-based approach, a dealer first looks for a qualified local media gateway. If none is found, it will *randomly* choose a qualified gateway discovered by the push or pull based method in other domains.

Figure 11 shows the average number of hops along the 'source EH - discovered media gateway - destination EH' path. The media gateway discovered by MeGaDiP incurs the least end-to-end delay in term of hop count.

Also based on the first experiment, Figure 12 shows how many dealers in the worst case are involved in the gateway discovery. For MeGaDiP, the number of dealers involved in the worst case grows linearly with the hop distance between the source and destination EHs. For the push or pull based approach, the number of dealers involved in the worst case only depends on the number of media gateways deployed in the environment.

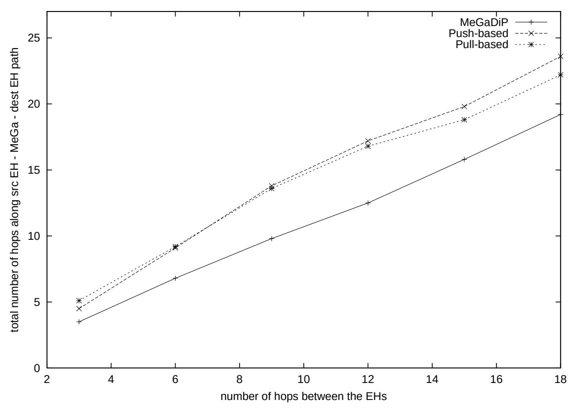

Fig. 11. Average number of hops along the 'source EH - discovered media gateway - destination EH' path

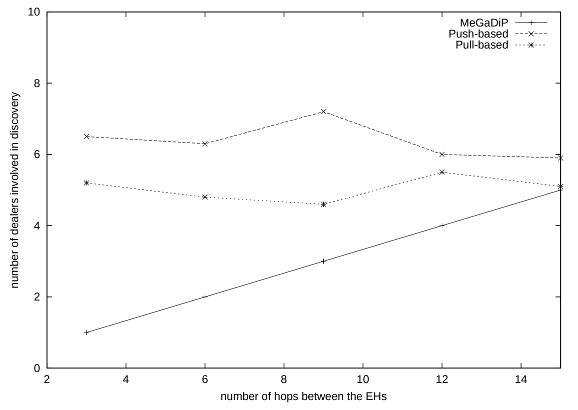

Fig. 12. Worst-case number of dealers involved in gateway discovery

We then evaluate the effect of caching in MeGaDiP. We deploy 10 class I media gateways with the same transcoding service in 10 trading domains. The EHs pairs are now randomly generated at a rate of 100 pairs per minute. We assume that each cached service advertisement will be invalidated within t amount of time, and t is uniformly distributed between 30 and 120 seconds. Figure 13 shows the average number of dealers involved in the discovery for each pair of EHs in each 30-second period. It is obvious that caching effectively reduces the number of dealers involved, thus reducing the discovery latency.

Finally, we examine the effect of dealer hierarchy on the success rate of media gateway discovery procedure. In the previous experiments, we observe that the discovery success rate of MeGaDiP is moderate, due to limited number of qualified media gateways and limited searching space defined by the basic MeGaDiP discovery procedure. We group the trading domains into larger domains, as illustrated by the dotted circles in Figure 10. Figure 14 shows the discovery success rate of basic MeGaDiP and MeGaDiP with dealer hierarchy, under different numbers of media gateways deployed in the environment. The success rate is computed based on 200 randomly generated pairs of EHs.

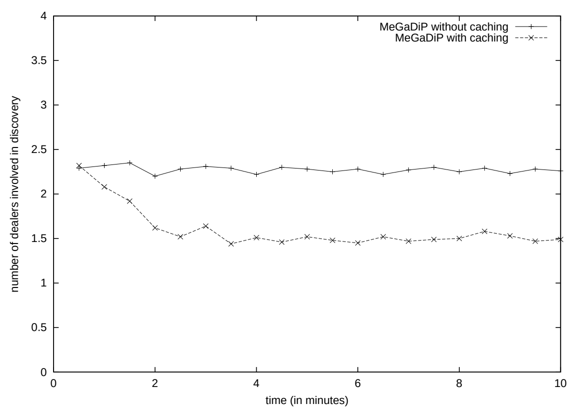

Fig. 13. Average number of dealers involved in gateway discovery: with and without caching

MeGaDiP with dealer hierarchy (as described in Section 4) constantly achieves higher success rate than the basic MeGaDiP, due to the expanded searching space.

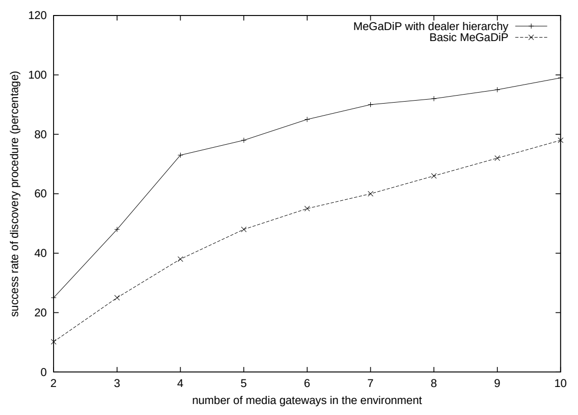

Fig. 14. Gateway discovery success rate: with and without dealer hierarchy

6 Related Work

Service location in networked environments has been an active research topic in recent years. Service Location Protocol (SLP) [3] defines the basic architecture that is also used in several other service location mechanisms (including MeGaDiP). The Directory Agent (DA), User Agent (UA), and Service Agent (SA) in SLP correspond to the dealer, end host (EH), and service provider in our architecture, respectively. However, SLP was originally designed for service location within one administrative domain such as an intranet, rather than in a wide-area environment. There have been wide-area extensions based on SLP[5][4][7]. In [5], two new entities: Advertising Agent and Brokering Agent, are introduced. An Advertising Agent multicasts selected service information from its local domain, and Brokering Agents in other domains listen to the multicast, and report the foreign services to their own local DAs. This approach may cause a global propagation of service information, which appears to be unnecessary in the context of media gateway discovery.

In [7], a framework for Internet Telephony Gateway (ITG) location is proposed. The framework uses Brokered Multicast Advertisements (BMA) to push service advertisements of ITGs to brokers in every domain, and each broker answers queries from clients in its local domain. In this approach, ITGs push their own service advertisements via well-known multicast channels. Since only one endpoint of an ITG session is in the Internet, while the other endpoint is in the Public Switched Telephone Network, the ITG location problem is still an RR/SL service location problem. It does not address (1) the constraint of *two* EHs and (2) two different classes of media gateways as defined in MeGaDiP. In addition, the ITG location framework does not explicitly support end-to-end resource awareness.

The multi-level hierarchy has been proposed as an efficient and scalable architecture for wide-area information distribution and discovery [4][20]. In particular, our dealer hierarchy extension is influenced by the Ninja Service Discovery Service (SDS) [4]. SDS servers (corresponding to dealers in MeGaDiP) arrange themselves into a hierarchy, and each SDS server only propagates summarized service information to its parent(s). The difference between SDS and the extension to MeGaDiP lies in the forwarding of service queries, which is also called query routing in SDS. If an SDS server can not find qualified service information (if it is a leaf SDS server) or service summary (if it is a non-leaf SDS server), the query will be immediately forwarded to the server's parent. However, in MeGaDiP, the query for a media gateway will first be forwarded to a neighbor dealer at *the same level.* Only when the discovery procedure fails at the current level, will the query be forwarded to the parent of the originating dealer of the current level. In fact, SDS also implicitly assumes RR/SL services. It can be shown that if SDS is used in media gateway discovery, the first three problems in Section 2.3 will still exist. SDS only solves the fourth problem by using hierarchical architecture and lossy service information aggregation.

The resource (or performance) awareness issue has been studied in the context of server selection in replicated services[21][22]. In [21], an application-layer anycasting service is proposed to collect servers' performance metrics (for example, server response time), and dynamically allocate servers to clients in order to minimize the response time. In particular, a hybrid push/probe technique is proposed to collect the performance metrics. In MeGaDiP, end-to-end resource awareness is implemented by (1) using 'push' within a trading domain, and (2) using 'probe-on-demand' across trading domains. In [22], a unified framework is presented for request scheduling and performance information collection in dynamic multimedia environments. An active probe technique is used to detect changes in network and server conditions. Most solutions to the server selection problem assume that the locations of candidate servers are already known, and the servers only provide RR/SL services. In contrast, MeGaDiP focuses on the discovery of media gateways with sufficient end-to-end resources. In addition, media gateways are not necessarily replicated in MeGaDiP.

7 Conclusion

In this paper, we present MeGaDiP, a wide-area media gateway discovery protocol. We first describe a basic architecture common in the current general service location mechanisms. We then identify weaknesses of the general service location mechanisms when performing media gateway discovery. Based on the basic architecture, we propose MeGaDiP as a discovery heuristics that will be invoked first when discovering a media gateway. Key properties of MeGaDiP include: (1) it is aware of both source and destination end hosts: any discovered media gateway will *not* create an excessively long path between the end hosts *via the gateway*; (2) it is resource-aware: any discovered media gateway is likely to have sufficient end-to-end resources to perform the requested service; (3) it returns discovery results with low latency and high validity by caching and resource validation; (4) it only introduces small amount of management traffic, regardless of the number of media gateways and end hosts in the environment. Furthermore, an extension to MeGaDiP using a hierarchical architecture is proposed to improve discovery success rate. Our initial experimental results demonstrate the soundness and feasibility of MeGaDiP.

We are currently refining the details of MeGaDiP in the dealer hierarchy architecture. We also plan to deploy MeGaDiP in a wide-area environment to study its performance. Of particular interest is the impact of underlying network topology and routing mechanism on the discovery success rate and the validity of discovery results.

Acknowledgement

We would like to thank the anonymous reviewers for their constructive comments which have helped us improve the final version of this paper. Our work was supported by the National Science Foundation under contract number 9870736, the Air Force Grant under contract number F30602-97-2-0121, National Science Foundation Career Grant under contract number NSF CCR 96-23867, NSF PACI grant under contract number NSF PACI 1-1-13006, NSF CISE Infrastructure grant under contract number NSF EIA 99-72884, NSF

CISE Infrastructure grant under contract number NSF CDA 96-24396, and NASA grant under contract number NASA NAG 2-1250. However, Views and conclusions of this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

References

[1] E. Amir, S. McCanne, R. Katz, An Active Service Framework and its Application to Real-Time Multimedia Transcoding, Proceedings of ACM SIGCOMM '98 (1998) pp. 178–189.

[2] W. Ooi, R. van Renesse, B. Smith, Design and Implementation of Programmable Media Gateways, Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2000) (2000) .

[3] E. Guttman, Service Location Protocol: Automatic Discovery of IP Network Services, IEEE Internet Computing 3 (4) (1999) pp. 71–80.

[4] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, R. Katz, An Architecture for a Secure Service Discovery Service, Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99) (1999) pp. 24–35.

[5] J. Rosenberg, H. Schulzrinne, B. Suter, Wide Area Network Service Location, Proceedings of the Fortieth Internet Engineering Task Force (1997) .

[6] F. Kon, R. Campbell, D. Mickunas, K. Nahrstedt, F. Ballesteros, 2K: a Distributed Operating System for Dynamic Heterogeneous Environments, Proceedings of IEEE International Symposium on High-Performance Distributed Computing (HPDC 2000) (2000) pp. 201–208.

[7] J. Rosenberg, H. Schulzrinne, Internet Telephony Gateway Location, Proceedings of IEEE INFOCOM '98 (1998) pp. 488–496.

[8] D. Xu, D. Wichadakul, K. Nahrstedt, Multimedia Service Configuration and Reservation in Heterogeneous Environments, Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS 2000) (2000) pp. 512–519.

[9] S. Chen, K. Nahrstedt, An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions, IEEE Network Magazine 12 (6) (1998) pp. 64–79.

[10] W. Jia, D. Xuan, W.Zhao, Integrated Routing Algorithms for Anycast Messages, IEEE Communications Magazine 38 (1) (2000) pp. 48–53.

[11] Y. Wang, Z. Zhang, D. Du, D. Su, A Network-Conscious Approach to End-to-End Video Delivery over Wide Area Networks Using Proxy Servers, Proceedings of IEEE INFOCOM '98 (1998) pp. 660–667.

[12] D. Comer, Internetworking with TCP/IP (Volume 1), Prentice Hall .

[13] R. Carter, M. Crovella, Server Selection Using Dynamic Path Characterization in Wide-Area Networks, Proceedings of IEEE INFOCOM '97 (1997) pp. 1014–1021.

[14] A. Downey, Using Pathchar to Estimate Internet Link Characteristics, Proceedings of ACM SIGCOMM '99 (1999) pp. 241–250.

[15] S. Seshan, M. Stemm, R. Katz, SPAND: Shared Passive Network Performance Discovery, Proceedings of USENIX Symposium on Internet Technologies and Systems '97 (1997) .

[16] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. Gryniewicz, Y. Jin, An Architecture for a Global Internet Host Distance Estimation Service, Proceedings of IEEE INFOCOM '99 (1999) pp. 210–217.

[17] K. Lui, K. Nahrstedt, Topology Aggregation and Routing in Bandwidth-Delay Sensitive Networks, Proceedings of IEEE GLOBECOM 2000 (2000) pp. 410–414.

[18] J. Rosenberg, H. Schulzrinne, Timer Reconsideration for Enhanced RTP Scalability, Proceedings of IEEE INFOCOM '98 (1998) pp. 233–241.

[19] K. Nahrstedt, H. Chu, CPU Service Classes for Multimedia Applications, Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS '99) (1999) pp. 296–301.

[20] N. Sturtevant, N. Tang, L. Zhang, The Information Discovery Graph: Towards a Scalable Multimedia Resource Directory, Proceedings of IEEE Workshop on Internet Applications '99 (1999) .

[21] Z. Fei, S. Bhattacharjee, E. Zegura, M. Ammar, A Novel Server Selection Technique for Improving the Response Time of a Replicated Service, Proceedings of IEEE INFOCOM '98 (1998) pp. 783–791.

[22] Z. Fu, N. Venkatasubramanian, Combined Path and Server Selection in Dynamic Multimedia Environments, Proceedings of ACM International Conference on Multimedia (ACM Multimedia'99) (1999) pp. 469–472.