mDNS - A Proposal for Hierarchical Multicast Session Directory Architecture

Piyush Harsh
CISE
University of Florida
Gainesville, Florida, USA

Richard Newman
CISE POB 116120
University of Florida
Gainesville, Florida 32611-6120 USA

Abstract ***Bandwidth in the Internet is constantly increasing. The last mile problem of the Internet has almost been solved. Multimedia has emerged as a dominant type of traffic on the Internet. Multicast is increasingly seen as the delivery vehicle of choice for multimedia streams. What has been the one true stumbling roadblock in widespread use of multicast is the lack of a convenient mechanism for multicast session discovery. This paper examines existing techniques that try to address this issue, highlighting the benefits and drawbacks of such schemes. It then proposes our hierarchical and globally scalable session directory architecture. An analysis of benefits and drawbacks of our scheme follows. The paper concludes with arguments why our scheme might be generally more suitable for global deployment, which may allow end users to enjoy the true power and efficiency of IP multicast.***

Keywords: Multicast, Internet Architecture, DNS, Resource Discovery, Scoping

1 Introduction

IP Multicast [1] is projected to be the vehicle of choice for content delivery for rich multimedia applications in the near future. Even though Internet average bandwidth is increasing and more homes are gaining high speed access to the Internet, the infrastructure is not keeping up with demand for multimedia content; IP unicast simply can not handle a huge subscriber base. With unicast, server and communication bandwidth requirements scale linearly with each subscriber. Maintaining huge server farms can become a costly proposition for organizations, and increasing multimedia traffic can clog the network. Multicast is an efficient answer to these increasing demands.

Even though the time seems appropriate for widespread deployment and use of IP Multicast, this has not happened. In our opinion, the lack of a familiar and usable facility to enable its use is the single most important reason for this. While DNS [2] has many of the desired characteristics, using DNS itself for multicast sessions information dissemination is inappropriate. Information stored in DNS is relatively stable over long periods of time, whereas multicast session information can be very dynamic; with a few exceptions, multicast sessions are not long term entities. There is a need for a multicast directory structure that is usable, scalable, and robust, yet which can handle the dynamic multicast environment.

Until now, the only way users could use Multicast sessions was if they somehow knew the session details beforehand. This has been accomplished via emails, IRC channels, and blog postings. If multicast is to become truly usable, a means must be devised for users to obtain these critical session parameters on the fly. Session parameter retrieval using URL-like strings can bring us a giant step closer to this goal.

The remainder of this paper provides a summary of current and past proposed solutions to the problems raised above. Next it provides the top-level architectural layout of our proposed scheme. The heart of the paper is a general URL construction scheme for multicast sessions, with proposed additional parsing and translation rules, and query handling infrastructure geared towards making multicast more user-friendly, easily accessible, and practical. It then makes cautionary comments regarding possible drawbacks in addition to the potential benefits of our approach. This paper concludes with possible future research directions and interesting questions that still remain.

2 Existing Multicast Session Directory Research

Session Directory tool - 'sdr' [3] - has been very popular among Mbone [4] enthusiasts. It has been used as a session management tool and is primarily based on LBL's Session Directory tool - 'sd'. It makes use of SDP - Session Description Protocol [5] - to announce the critical characteristics of multicast sessions such as channel address, port numbers, timing and resource information needed for remote hosts to join the session. It uses a well-known multicast channel address to propagate this information over the Internet. It also maintains a cache of other multicast sessions advertised elsewhere on the Internet through sdr. Based on the cache information maintained locally by each 'sdr' client, it tries to assign a new channel address to requesting multicast sessions in such a way as to reduce the address collisions among different sessions. It is the general consensus of the research community that even though 'sd'/'sdr' was a great technology demonstrator, it is not globally scalable. All 'sdr' clients essentially maintain a flat hierarchy on the Internet, which makes information dissemination among different 'sdr' clients a challenging prospect, in addition to make search difficult and using much storage.

Andrew Swan and team [6] at Berkeley developed a completely decentralized session directory that they incorporated as part of their Light-Weight multimedia session framework. In this architecture they advertise the multimedia session's bindings at a well-known bootstrap address using Sessions Announcement Protocol [7]. To overcome the high latency that plagues the multicast session directory architecture based on LBL's 'sd' application and SAP announcement bandwidth limitations, they propose a tiered announcement rate approach. The announcement agents under the local scope announce session advertisements at a much higher frequency than traditional SAP clients. They also propose splitting the traditional SAP client into two parts: one persistent server that runs SAP and caches all the network SAP announcements heard over a long period of time, and another ephemeral client that contacts this persistent server for the cached list of available sessions.

In [8] Joaquim and team analyze the use/misuse of SDP as a session directory tool to advertise multimedia sessions. They argue that the session directory information that is embedded inside SDP fields is not standardized. Had it been standardized, these fields such as "media," "repeat time," "time active," etc. might be used for aggregating sessions, which could then be used later to query for sessions. They propose that the user should not be burdened with the task of browsing a flat structure; instead the task could be assigned to a server. This information could be presented to the user using either a well-known multicast channel, or using several multicast channels, or possibly even using some specific server database. The article did not specify to what extent any of these proposed solutions were already implemented or the current status of their work.

Another attempt at making a distributed information discovery system on the Internet was Harvest [9]. This system was built using subsystems such as gatherers, brokers, and replicators. Gatherers are placed at the resource site and manage local resource data. Brokers collect data from gatherers and incorporate this data in their resource index; they further include index/search subsystems optimized for space and/or search time. The system also made use of replicators to copy the data over multiple sites in the Internet and place object caches at critical sites to minimize the communication load on the Internet. Instead of Harvest being truly hierarchical, we believe it is better characterized as a replicated, Internet-wide cache, and is not entirely suitable to solve the multicast session discovery problem. It would be difficult to incorporate scoping and session lifetime requirements within their proposed framework. Moreover, the dynamic nature of most multicast sessions would result in cache instability.

Researchers at UCLA have proposed a scalable multicast information discovery graph (IDG) [10] based on the semantic description of stored as well as real-time multimedia content over the Internet. This work is being done under the Sematic Multicast project [11]. Even though their proposal provides for a hierarchical, semantic directory, it is not truly distributed. They have proposed making use of caching and soft state refreshes to make their system more scalable and robust. In their approach, a new multicast user may have to start searching at a well-known root directory server, which is apt to create a bottleneck as the number of sessions and users grow. This scheme may also be problematic in the face of stale caches and periodic cache refreshes. The semantic hierarchy in their proposal currently is coarsely defined and may require significant rework in order to account for the variety of multimedia sources uploaded online these days. The architecture does allow for much better search time compared to LBL's 'sd' tool, but its bandwidth requirement grows linearly with the number of data

sources; this is clearly a source of concern. Also, multicast scoping requirements are not mentioned in the published work.

In [12], the authors propose building an anycast SDP Proxy in order to give end-users immediate access to session announcements. They propose an architecture along with an HTTP-style protocol format to access session information as well as to create a session entry at the remote SDP Proxy. This approach still suffers from the traditional 'sdr' issues of non-scalability in the face of a large number of sessions. Another issue with this approach is deployability over Source Specific Multicast (SSM [13])-only networks.

In [14] the authors deal specifically with multicast session announcements over SSM networks. They propose a two-tier hierarchy of dedicated session announcement servers (SAS). They propose to reduce the SAP-related delay by increasing the allowed bandwidth limit of 4 kbps to 50 kbps for intra-domain announcements. SAS servers located in the backbone network of various ISP networks (Level 2 SAS servers) act as relays and do not cache any announcements. Simply increasing the intra-domain bandwidth seems to be a nearsighted solution at best. They state that whenever a new SAS Level 2 server is added in the ISP's backbone network, the ISP discovers the address of other Level 2 SAS servers and this information is configured into the new server. However, their paper does not mention how this is achieved, or whether the configuration is manual or automatic.

3 DNS-aware Multicast Session Directory Architecture

This section describes our proposed globally scalable, multicast session directory architecture. It is designed on principles similar to those behind domain name server (DNS), and is intended to satisfy requirements for

- usability,
- robustness,
- scalability, and
- maintainability.

We first provide terminology that we will use in the rest of this section.

- MSD_x^y - Multicast Session Directory (MSD) server 'y' in domain 'x'
- MSD_x^d - Designated MSD Server in domain 'x'
- DNS_x - Domain Name Server for domain 'x'
- URS_x - URL registration server in domain 'x'

We assume that each domain knows its DNS server address, and DNS servers know their parent DNS server's address, which we believe is a reasonable assumption. For global discoverability of multicast sessions, we assume that at least one MSD server coexists with the DNS server at each domain level. Failure to do so may result in disconnected islands of session discovery zones in the global Internet (which actually may be desired in some cases). We propose to make an additional entry into DNS server's record table. We call this entry an MCAST record, and it contains such details as the 'anycast IP address' [15] for MSD servers in that domain, globally scoped multicast channel details for establishing a multicast group with the designated MSD server of a particular domain, addresses of designated MSD servers in the children subnets, the globally scoped multicast channel details for establishing a multicast group with the designated MSD server of that particular domain, and the address of the designated MSD server in the parent's domain. Additionally it may contain the address of a URS server in its domain. An example DNS MCAST record entry is shown below.

```
@MCAST{
    ANYCAST=a.b.c.d
    CMCAST=233.[ASN Byte1].[ASN Byte2].XXX
    PMCAST=233.[ASN Byte1].[ASN Byte2].???
    PORT=pqrs
    URS=x.y.z.w
}
```

Here, ASN denotes the AS (Autonomous System) Number. Note that this example is just for illustrative purposes, and an actual DNS entry must follow the correct DNS entry format. Notice that we have suggested the use of globally scoped addresses from the GLOP [16] address range. These address are assigned by the ISPs, and the domain owners/administrators must apply for these addresses from their ISPs. The system has been designed to co-exist with our HOMA [17] multicast address allocation and management scheme.

3.1 mDNS Hierarchy Construction

This section describes the hierarchy of our scheme through two example partial networks, namely .edu hierarchy network and a general hypothetical ISP network. Our scheme will work with any network organization scheme as long as our initial DNS server and MSD server assumptions remain valid. A typical example is shown in figure 1.

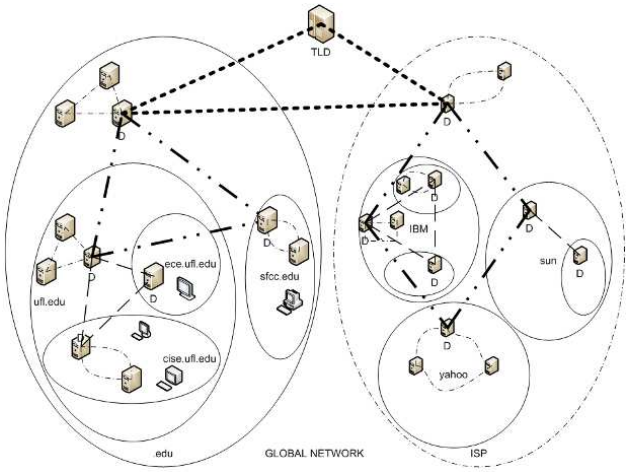

Figure 1: A typical MSD hierarchy

Now we explain the proposed mDNS hierarchical architecture in some detail. First consider the (partial) .edu hierarchy. Under the .edu domain are two university networks. Under the UF network are two sub-domains, namely CISE and ECE; each of these is an independent, administratively-scoped multicast domain. Furthermore, UF is also an administratively-scoped domain. CISE and ECE each maintain their own DNS servers, whose parent DNS server is the DNS_{UF} server. DNS_{UF} server is a child node of the TLD $DNS_{.edu}$ Server. UF maintains multiple MSD_{UF} servers, all of which subscribe to a fixed (possibly IANA assigned) administratively scoped multicast channel. From here on we will refer to this channel as the MSD-LOCAL-MCAST channel. CISE and ECE also maintain their own sets of MSD_{CISE} and MSD_{ECE} servers, and again these subscribe to the MSD-LOCAL-MCAST channel. Since this channel is an administratively-scoped channel, there should be no cross-talk among these channels (assuming the edge routers are properly configured). If there are multiple MSD servers maintained under a domain, a designated MSD server is chosen based on some leader election algorithm [18] [19]. In figure 2, these are marked with the letter 'D' next to them. The designated MSD^d server joins two globally scoped multicast channels, namely those specified by the CMCAST and the PMCAST entries in the MCAST record of the DNS server in their domain. If either of these two entries is NULL, the server does not subscribe to that particular channel. It is important to note that all the MSD servers in any particular domain (excluding child subdomains) are anycasted at the anycast IP address specified in the DNS server's MCAST record.

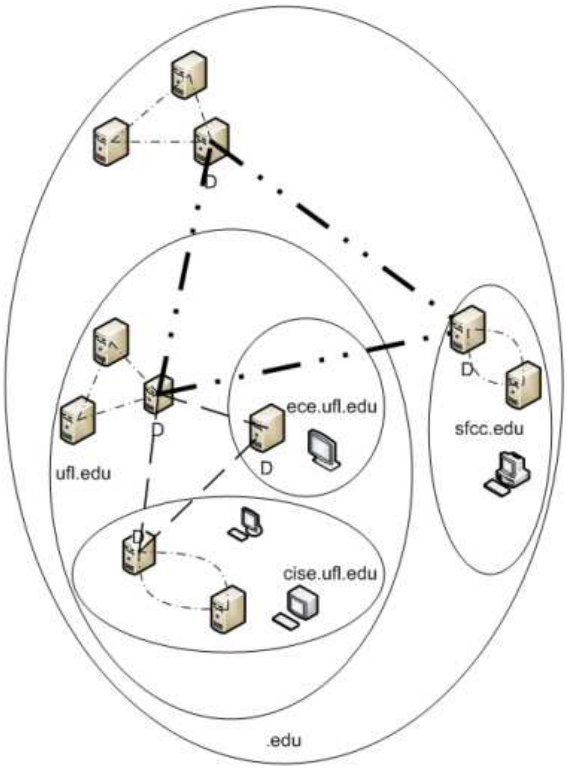

Figure 2: An example .edu hierarchy

MSD Server's Base Algorithm

```
join MSD-LOCAL-MCAST channel
initiate leader election on this chanel
if elected leader then
    query local DNS server for PMCAST and CMCAST
    join PMCAST and CMCAST channels
end if
```

This is how the MSD servers' hierarchical structure is established in the mDNS architecture. It is quite easy to see that the hierarchy will exist as long as the initial two assumptions are satisfied in any network domain hierarchy.

3.2 Session Registration in mDNS

mDNS architecture has been designed to work with with our HOMA [17] proposal for multicast address allocation. We assume that an application in any domain has had an appropriate multicast channel address allotted to it before transmitting multicast traffic (in any scope). While this paper does not provide details on the internal database maintained at each MSD server, let us posit that MSD servers are capable of registering channel keywords along with channel details on behalf of multicast applications. It is the responsibility of the session creator

to provide a limited number of keywords correctly describing the session when registering. These keywords will aid in the session discovery process described later.

An application under the mDNS architecture, after it has acquired a valid channel address, will execute the following pseudocode.

```
Session Registration Pseudocode
contact local DNS Server to find URS address
if URS server exist then
  request URS server to register a channel descriptive 'keyword'
  if keyword registration at URS successful then
    done.
  else
    pick another keyword
    try again
  end if
end if
initiate channel registration request on the MSD-LOCAL-MCAST
register the channel details with MSD server
provide list of keywords (max upto 10)
provide session duration and operating times
provide URS registerd 'keyword' if any
```

The MSD server will correctly identify the scope of the multicast channel based on the channel address with which it is registered.

3.3 mDNS Search Operation

Multicast sessions in mDNS can be searched using session keywords. Sessions can also be accessed directly if the session creator successfully registered a valid 'keyword' with the domain's URS server. We propose a simple URL scheme in order to facilitate multicast channel details access in order for the remote host to sunscribe to that particular channel on the Internet under the mDNS architecture. We believe that having an URL scheme will greatly enhance the usability of IP-multicast and would enable it to reach its fullest potential rapidly.

mDNS URL is constructed using the following syntax.

<protocol>://<domain URL>/<URS Keyword>

In the above URL scheme, 'protocol' is the method the remote user will use to communicate with the MSD server defined in the DNS MCAST record. It could be HTTP or a similar protocol. The domain URL helps resolve the MSD server located in the multicast session creator domain. It must begin with 'mcast' to specify the MSD server. For example, let us assume that under the cise.ufl.edu sub-domain, we have created a globally scoped multicast channel that multicasts information about gators. Further assume we have successfully registered the keyword 'gators' with the URS_{CISE} server located in the CISE domain. Someone else can then directly access the multicast session details to subscribe to the gator channel using this URL:

http://mcast.cise.ufl.edu/gators

The search is done in a very similar fashion. Under mDNS, a user can do a domain-specific search or a general global search. If the user wishes to do a domain-specific search, s/he can specify the specific domain to search for a particular keyword in the same fashion as shown above, but now using the qualifiers 'search' and 'keyword' in the URL. For example, if the user wishes to find sessions with keyword 'gators' under the cise.ufl.edu domain, they can do so by using the following string.

mcast.cise.ufl.edu/search=all&keyword=gators

The end user's multicast search application first resolves the mcast.cise.ufl.edu anycast address and then connects to one of the MSD_CISE servers located in the cise.ufl.edu domain. MSD servers then perform a database search against the keyword 'gators' and every content type. Content type could be audio, video, text, image, whiteboard, etc. Search is performed in a top-down hierarchy starting from the domain specified in the search URL and percolating down to all sub-domains (if any exist). In our present scheme, the search results are returned from MSD servers at each level directly to the requesting client. There are a few other delivery schemes we are considering described in the future research section. The MSD servers are smart in the sense that they recognize whether the query comes from a host inside their domain or outside it. If the querier is located outside the MSD domain, the MSD returns only those search results for globally scoped channels. Administratively scoped sessions are only returned if the querier resides in the same domain.

Additionally any user can choose to perform a global keyword search. In order to do this, the client must contact the local MSD server co-located in the same zone as its DNS server. Global search is propagated by the MSD servers on both PM-CAST and CMCAST channels, as well as its own MSD-LOCAL-MCAST channel, thereby spreading the search to both child domains as well as parent domain. The propagated search request contains a uniquely identifying string that allows the originator MSD to kill the search if the same query id comes again to the same MSD server. Also the search query based on the identifying string is never

re-propagated on the same channel on which it was received. It is easy to see that in the proposed mDNS architecture, this uniquely identifying search id will be generated by the designated MSD server at each level and only when the search query was received on the MSD-LOCAL-MCAST channel.

Pseudocode for search executed at each MSD server is given below.

```
MSD Pseudocode for Search
received search query on subscribed channels
search sessions internal database for search match
if multicast sessions found then
  if querist resides in another domain then
    return only globally scoped session details (if
    any)
  else
    return every result found directly to the
    querist
  end if
end if
if MSD server is a designated MSD server then
  if search unique ID is missing then
    generate unique search ID
  end if
  if query request was not received on CMCAST
  channel then
    propagate search on CMCAST channel
  end if
  if search is global in nature then
    if query request was not received on PM-
    CAST channel then
      propagate search on PMCAST channel
    end if
  end if
  if search query has self generated previous ID
  then
    drop search request
  end if
end if
```

3.4 Search Example

Here we show an example hierarchy to demonstrate how a search operation is performed in mDNS.

Assume a global search is initiated by an end host at the only subnet in the Sun network. The search goes to the only $MSD_{SUN_{subnet}}$ server in the subnet, which is also the designated MSD server. The $MSD^d_{SUN_{subnet}}$ server searches its local session database and returns all the hits to the requesting host's application.

The designated $MSD^d_{SUN_{subnet}}$ server then generates a unique search query ID and propagates the search to both CMCAST and PMCAST channels.

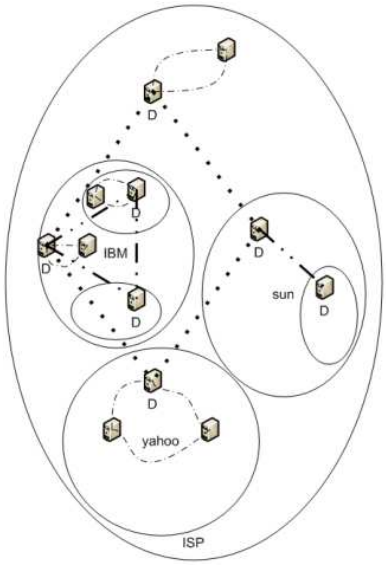

Figure 3: A general mDNS hierarchy

In this example network, the CMCAST channel does not exist for the SUN internal subnet, so the search is propagated only on PMCAST channel (dash dot dot dash line).

When the query reaches the Sun network-wide MSD_{SUN} server, this outer MSD_{SUN} server searches its internal sessions database. Since the search came from a subnet host and not from a co-existing host, it only returns those results that are global in scope. MSD^d_{SUN} propagates the search on the PMCAST channel (dotted line).

Now the search query reaches the MSD servers in the IBM, Yahoo, and the ISP's TLD MSD server (i.e., MSD^d_{IBM}, MSD^d_{Yahoo}, and MSD^d_{ISP}). They all process the query using the same search algorithm and the process continues.

In the example scenario here, if the only SUN subnet has additional sub-subnets under it, the search query would have propagated on the CMCAST channel and thus would have propagated down the hierarchy as well. This is how the query eventually propagates to all MSD servers in the mDNS architecture.

4 Analysis of mDNS

We believe that mDNS has features to take IP-Multicast to the next level of deployment and use in the general consumer network. Together with the HOMA [17] system for multicast address allocation, mDNS has true potential to make multicast session discovery and deployment seamless and consumer friendly. Use of the mDNS URL scheme presented

in section 3 makes bookmarking of popular multicast sessions feasible and as user friendly as webpage bookmarks are currently.

However, mDNS is still in its early development stages, and has certain drawbacks in its current form. It is particularly vulnerable to DoS attacks. Also, each global query has the potential to activate every MSD server deployed under the mDNS scheme. Furthermore, direct query results transmission to the querier from MSD servers creates the possibility of Smurf [20] attack on some remote host on the Internet using IP spoofing.

There are several immediate benefits of the mDNS scheme. First is the database space savings. In LBL's sd and later sdr software implementation, possibly every sdr client may cache session details in the software's local cache. This makes the sdr scheme particularly difficult to scale. In mDNS, only the leaf subnet's/domain's MSD server in the hierarchy maintains the multicast session database. Latency is also improved, as the receiver application does not have to wait 10s of minutes to discover a session because of SDP's global bandwidth usage restriction. We conjecture that session search and discovery in mDNS scheme should be many orders of magnitude faster than current schemes. Benefits of URLs have already been discussed earlier.

5 Future Research

We are also considering a subscription-based approach where each MSD client, in addition to maintaining a locally residing session source database, also maintains keywords subscription level by hosts in its subnet/domain. We are devising a threshold-based system to propagate the keyword subscription up the mDNS hierarchy.

A major concern is activation of every MSD server in the mDNS scheme for every global search. This could prove to be very taxing on MSD servers. We are considering intelligent cache placements schemes along with smart caching techniques in order to reduce the possible workload on MSD servers.

Security remains a major concern in any distributed deployment in the Internet today. mDNS in its current form is vulnerable to various attacks. We are studying threats to mDNS and possible solutions to thwart identified attacks.

This paper provides a very high level description of the mDNS architecture. Design of protocol message formats and database records structure are not yet complete. We are in the process of implementing a prototype system to provide hands-on experience with mDNS. The prototype will also allow us to analyze the real stress levels on MSD servers.

References

[1] S. E. Deering. "Multicast routing in internetworks and extended LANs"; SIGCOMM '88: Symposium on Communications architectures and protocols, 55-64, 1998.

[2] P. Mockapetris and K. J. Dunlap. "Development of the domain name system"; SIGCOMM Comput. Commun. Rev. 18:4, 123-133, 1988.

[3] M. Handley. "The sdr session directory: An Mbone conference scheduling and booking system"; http://www-mice.cs.ucl.ac.uk/multimedia/software/sdr/, April, 1996.

[4] Almeroth, K.C. "The evolution of multicast: from the MBone to interdomain multicast to Internet2 deployment"; Network 14:1, 10-20, Jan/Feb 2000.

[5] M. Handley and V. Jacobson. "SDP: Session Description Protoco"; Internet Draft (IETF), September 1997.

[6] A. Swan, S. McCanne, and L. A. Rowe. "Layered Transmission and Caching for the Multicast Session Directory service"; Multimedia, 119-128, 1998.

[7] M. Handley, C. Perkins, and E. Whelan. "Session Announcement Protocol"; RFC 2974, 2000.

[8] A. Santos, J. Macedo, and V. Freitas. "Towards Multicast Session Directory Services," http://citeseer.ist.psu.edu/264612.html (April 2008).

[9] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. "The Harvest information discovery and access system"; Computer Networks and ISDN Systems 28:12, 119-125, December 1995.

[10] N. R. Sturtevant, N. Tang, and L. Zhang. "The Information Discovery Graph: towards a scalable multimedia resource directory"; IEEE Workshop on Internet Applications, 72-79, August 1999.

[11] "Semantic Multicast project"; Web Resource: http://www.wins.hrl.com/projects/semcast/.

[12] P. Liefooghe and M. Goosens. "The Next Generation IP Multicast Session Directory"; SCI, Orlando FL, July 2003.

[13] H. Holbrook and B. Cain. "Source-Specific Multicast for IP"; Work in progress (IETF), October 2003.

[14] P. Namburi and K. Sarac. "Multicast session announcements on top of SSM"; 2004 IEEE International Conference on Communications, 1446-1450, June 2004.

[15] C. Metz. "IP anycast point-to-(any) point communication"; Internet Computing 6:2, 94-98, March/April 2002.

[16] D. Meyer and P. Lothberg. "GLOP Addressing in 233/8"; RFC 3180, 2001.

[17] P. Harsh and R. Newman. "An Overlay Solution TO IP-Multicast Address Collision Prevention"; IASTED EuroIMSA, March 2008.

[18] M. Mirakhorli, A. A. Sharifloo, and M. Abbaspour. "A Novel Method for Leader Election Algorithm"; CIT '07: 7th IEEE International Conference on Computer and Information Technology 452-456, 2007.

[19] S. Dolev, A. Israeli, and S. Moran. "Uniform Dynamic Self-Stabilizing Leader Election"; IEEE Trans. Parallel Distrib. Syst 8:4, 424-440, 1997.

[20] C. A. Huegen. "The Latest in Denial of Service Attacks: "Smurfing" Description and Information to minimize effects"; http://www.pentics.net/denial-of-service/white-papers/smurf.cgi (May 2008).