Recovering from mDNS domain failures

Piyush Harsh[1], **and Richard Newman**[2]
[1]Computer and Information Science and Engineering, University of Florida, Gainesville, Florida, 32611 USA
[2]Department of CISE, University of Florida, Gainesville, Florida, 32611 USA
Email: [1]pharsh@cise.ufl.edu [contact author], and [2]nemo@cise.ufl.edu
Target Conference: ICOMP

Abstract— *'mDNS' - a hierarchical multicast session directory service architecture, which has been recently submitted to IETF editorial board for publication under Best Current Practice (BCP) track, allows administrative domains to join the global hierarchy incrementally. The global structure dynamically adapts to the changing topology. This paper describes the various failure scenarios in the proposed IETF document and especially the scenario where a participating domain goes down completely. It describes how such a scenario could effect end users' experience and how the system can temporarily recover from such failures until the erring domain can be revived.*

Keywords: failure recovery,multicast session directory,distributed search

1. Introduction

'mDNS' [1] [2] [3] [4] [5] is a hierarchical and globally scalable solution to IP multicast session directory [6] [7] [8] [9] [10] [11] service problem. Its coexistence alongside Domain Name Service (DNS) [12] allows multicast [13] [14] [15] sessions to be assigned a unique URI string similar to domain and web site URLs. Even the most transient multicast sessions become instantly searchable under the proposed architecture. And the supported algorithms ensure the assigned URI string's validity in the face of changing session parameters. 'mDNS' improves multicast content discovery and enhances the usability of the technology. All this is achieved without significant system administrators' involvement in the overall platform management.

'mDNS' architecture is mostly self-managing. The architecture fine tunes the configuration parameters automatically in the face of dynamically changing global topology. It uses a tree based Distributed Hash Table (DHT) structure that enables efficient keyword routing [3] to support fast search lookups. The architecture is multicast mode independent [4] i.e. it can co-exists in domains supporting ASM version of multicast or SSM version of multicast. In fact it can coexist in a domain that does not support any form of multicast.

Essentially, 'mDNS' is an overlay structure and the domains participating in the overlay structure can join-in and leave as they please. We foresee that hierarchy changes will be rare once the overall structure has stabilized. Although domain failures in the participating domains can still happen. In the rare event, a domain may also decide to end its participation in the overall hierarchy. In this paper we will understand what impact such an event will have on correct session search routing. We will analyze failure scenarios other than domain failure that could disrupt the services to end users and would identify the impact and possible way-around these failures.

2. mDNS - a brief overview

An 'mDNS' administrative domain is made up of a DNS server, URL registration server (URS) and one or more Multicast Session Directory (MSD) server(s). If multiple MSD servers are running, one is chosen using some leader election process [16] [17] as designated MSD server referred from hereon as MSD^d.

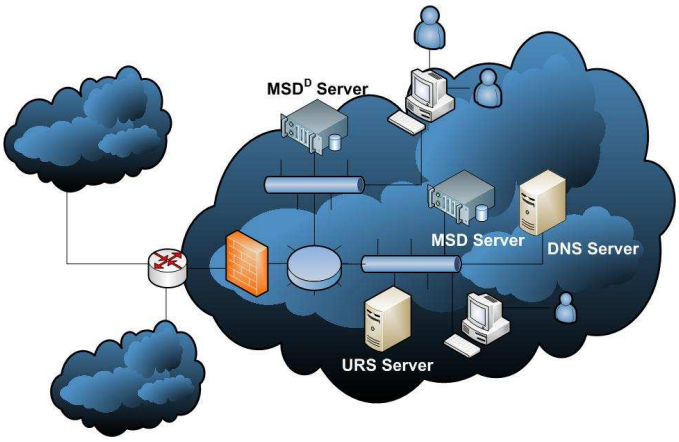

Fig. 1: a typical mDNS domain components

Figure 1 shows a typical 'mDNS' domain along with all the necessary components. At every site, the system administrator is required to enter a one time configuration detail in the URS that allows proper bootstrapping of the domain with rest of the 'mDNS' hierarchy. These settings include -

- PMCAST
- CMCAST
- supported IGMP version
- mDNS parent domain's URL string

The domain's MSD^d retrieve these parameters from the URS and grafts itself in the overall global hierarchy. The hierarchy characteristics depends on the supported multicast environment in the underlying network plane.

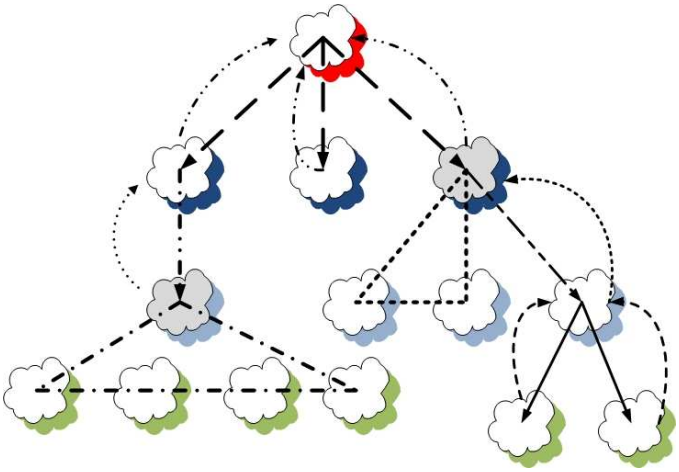

Fig. 2: a typical mDNS domain hierarchy

Figure 2 shows a typical mDNS domain hierarchy. Depending on the underlying network multicast support, some domains are operating in ASM mode while some others are operating in SSM mode. The shaded domains are hybrid domains that support both ASM and SSM multicast and therefore can act as bridge between two different types of mDNS domains.

'mDNS' strives to provide for equitable data sharing responsibilities among participating domains by assigning almost equal hash spaces to domains. Since we recommend using secure hashing schemes such as SHA-2 [18] and MD5 [19], keywords hash values would be ideally uniformly distributed among participating domains. But skew in session keywords popularity may result in unequal storage distribution in the participating domains. Further, the hash space distribution among domains is subject to hierarchy stabilization threshold scheme to ensure global routing stability. 'mDNS' components generally use soft-state [20] approach for maintaining operating configurations and therefore adapt better to changes in working environment or changes in overlay structure.

The supported algorithms such as session registration, search routing, hash space assignments and routing stabilization as well as message protocol formats have been described in details in our earlier publications [1] [2] [3] [4] and in the IETF BCP RFC [5] document. In this paper we will focus on the failure analysis and possible solutions.

3. Failure Scenarios

'mDNS' services depends on the DNS service infrastructure and the MSD servers and URS operating in a domain to provide users with registration and search capabilities. Failure in any one or more of these components can lead to service disruption to a group of end users in the Internet. Let us see each one of these failures in some detail.

3.1 Impact of DNS failures

DNS services are used to resolve the domain portion of the 'mDNS' URI string. The domain portion of the string refers to the URS installed in that domain. The particular session details are retrieved from the URS and thereby allow the end user to start accessing the multicast session of choice that s/he bookmarked earlier. Although DNS infrastructure is distributed with lot of redundancy built at several levels, there still exist few failure scenarios which we will look at next.

3.1.1 Failure at root level DNS servers

Root level DNS servers are highly replicated for load balancing and redundancy reasons and since the TLDs are very frequently visited, it is very likely that the TLD DNS server connectivity details are cached at lower level DNS servers. In most cases in the name resolution process the root servers are entirely bypassed. Failure at root level should not impact 'mDNS' resolution process terribly.

3.1.2 Failure at the TLDs

Failure in the Top Level Domain (TLD) DNS server could cause problems in the name resolution process unless the authoritative DNS details are cached at lower level DNS servers, in which case the TLD can be bypassed. The caching at DNS servers depends on the frequency of visitation of a particular domain. Typically a DNS cache entry is set to expire in 48 hours. Therefore unless a particular domain is visited often, the entry would not be present in the local DNS server and thus 'mDNS' name resolution process will fail.

3.1.3 Failure along the resolution chain

Failure in the DNS server along the resolution path would also disrupt the name resolution process. Typically many domains maintain a primary and a secondary DNS detail so that an alternate resolution path can be used. If the path converges before the failed link in the resolution chain, the overall resolution process will fail too.

3.1.4 Failure of the authoritative name server

This will most likely lead to failure, because the IP mapping of the URS is maintained as an A record at the authoritative DNS server.

The DNS failures are generally very rare. There were some TLD poisoning attacks but they were largely ineffective because of caching and replication of the TLD DNS infrastructure.

3.2 Failure in the URS

'mDNS' URI Registration Server (URS) is critical in the URI string resolution process. The final step in the resolution process is contacting the domain's URS and retrieving the desired session parameters. If the URS is down, then the session parameters can not be retrieved and the end user will not be able to access the multicast stream. Figure 3 shows the 'mDNS' URI string resolution process.

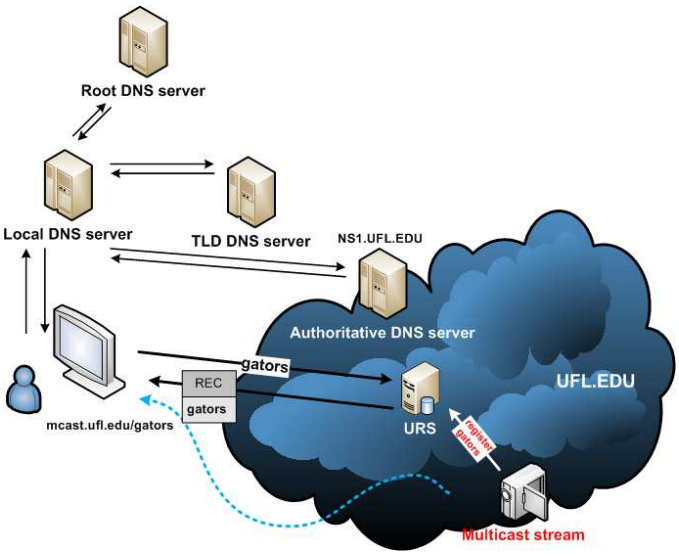

Fig. 3: mDNS URI resolution process

Every multicast session originating at a domain is required to register with the URS server in that domain. Therefore, its failure will result in resolution failure of all the URIs pointing to sessions originating in that particular domain. But that would not effect name resolution process for multicast sessions registered with URS in other domains. Such a problem can be resolved by providing ample redundancy for URS. In fact if multiple replicated URS are present, the authoritative DNS for that domain can use IP rotation feature to achieve load balancing on the URS.

As we mentioned earlier that the URS acts as a bootstrapping device for MSD^d server, it also helps in communication failure scenarios. Since the URS contains the parent domain's URL, it is capable of contacting the parent URS and query it for the IP and port details of the MSD^d server of that domain. In the scenario where the MSD server is unable to get parent MSD communications, it may failover to IP unicast for getting critical hierarchy maintenance information. If URS fails, it may not be able to achieve that.

Another scenario is, when the URS fails first followed by MSD^d server failure. If there are backup MSD servers, one such could take up the responsibilities of the failed MSD server, but it will not be able to retrieve bootstrapping details from the failed URS. Such a scenario could lead to islands of disconnected domains in the overall hierarchy. The MSD^d would soft-state timeout and will eventually try the domain failure recovery algorithm, more details on which is provided later.

3.3 Failure of MSD server

If a domain has multiple MSD servers running, then the failure of the MSD^d server would not result in any major service disruptions. One of the other servers would take over the responsibilities of the failed designated MSD server after winning the leader election algorithm. If there is no backup, then the MSD failure would essentially cut off that domain from rest of the 'mDNS' hierarchy. Any child domain of the failed domain must then initiate domain failure recovery algorithm after a soft-timeout. Any administratively scoped multicast session registered with the failed MSD^d server would become irretrievable by end users in that domain. In fact a global search by users in that failed domain would not be possible. Domain specific search to some other domain might still be possible though.
Let us now see how a participating domain in the 'mDNS' hierarchy recovers from the parent domain's failure.

4. Recovering from domain failure

The hierarchy maintenance information such as hash space assignments, trickle from the root domain in the global hierarchy to the leaf domain. Similarly, the domain count values are aggregated and reported upwards from the leaves to the root domain which in turn helps the root domain in appropriately distributing / re-distributing the overall keyword hash space among all participating domains. Therefore any domain failure results in the disruption in the communication chain from the root to the domains lying below the failed domain in the subtree rooted at the failed domain.

If there are no mechanisms in place to bypass the failed domain(s), it would result in disconnected islands in the global system. The session database will become fragmented and possibly inaccessible among users in different connected islands. The session records stored at the failed domain can still be located by interested users because of the shadow copy of the records getting stored at different site based on the inverted hash bits. Figure 4 shows the shadow copy insertion process. But it becomes necessary to maintain overall connectivity in all live domains by somehow bypassing the failed domain.

As communication percolates from the root node to the leaves in the tree DHT, each domain knows the unicast connectivity information of the root domain and also the hash-space range allotted to the failed domain (parent domain mostly). After a suitable soft-timeout interval, the child nodes of the failed domain sends a [tracer] request to the root node including a hash value from within the failed domain's hash space and its own unicast connection information.

Algorithm 1 shows what steps a failed domain's children take in order to recover from the parent's failure in order to

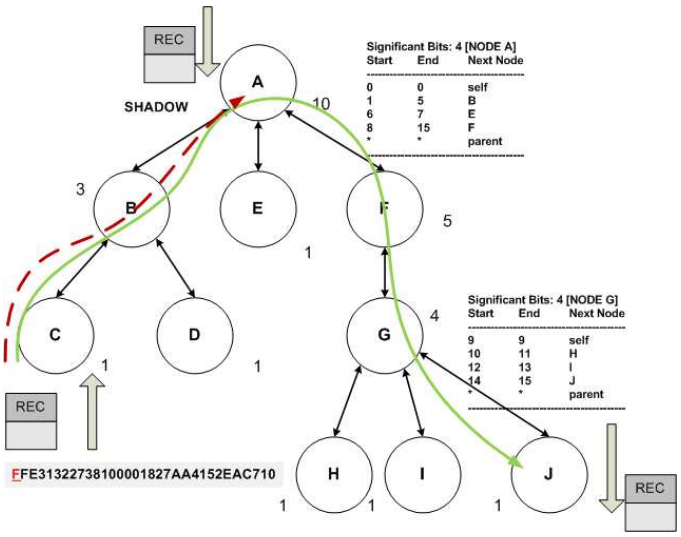

Fig. 4: mDNS session registration and insertion

Algorithm 1: Node failure recovery algorithm

```
1  begin
2      {No parent heartbeat received for 'n' consecutive timeouts }
3      if subscribed to receive parent's communication over PMCAST
       then
4          {send request to parent to receive communication over
           unicast }
5          if connection re-established then
6              {proceed to function normally }
7          end
8          else
9              {proceed to else section of outer 'if' }
10         end
11     end
12     else
13         {send [tracer] request to the root node }
14         {upon receiving graft details, initiate grafting }
15         {periodically keep pinging original parent node }
16         if original parent comes online later then
17             {detach from the temporary graft location }
18             {resume regular operations }
19         end
20     end
21 end
```

maintain the communication connectivity with the root and the rest of the 'mDNS' hierarchy. The domain initiating the recovery algorithm sends one hash value from within the parent's hash space along with its own unicast connectivity details to the root node. The root node initiates a trace route using the hash value as the routing parameter. These trace-route messages are sent to the next level domains using IP unicast only. The last node that is unable to pass on the trace-route message to the next level child node using unicast, has thus identified the failed domain. That node then temporarily assumes the management responsibilities of the hash space allotted to the failed domain. It also sends its own unicast connectivity details to the original domain that initiated the failure recovery algorithm. The algorithm initiator hence knows the graft point in the global tree hierarchy and this graft point is the closest ancestor domain in the DHT tree hierarchy.

The failure recovery algorithm is initiated by all the children domains of the failed domain and they all eventually are grafted at the common ancestor node in the hierarchy thus bypassing the parent. They communicate with the temporary host domain using IP unicast only. In he meantime they keep on probing the liveness of the failed domain periodically. If the failed domain comes back online, the original children domain sends a graft-detach notification to the ancestor node and reattaches to the original parent domain, the parent also notifies its own parent.

The failed domain's hash space is temporarily reserved for that particular domain until a preset timeout interval. During that interval the hash space management is undertaken by the closest live ancestor node. If the failed domain does not recover by that timeout interval, its space is reassigned to other live domains. After that if the domain recovers, the usual hash-space allotment algorithm process takes over depending on the node count reports that reaches the root node.

Figure 5 shows the domain recovery strategy in steps. The shaded domain is the failed domain. Its two children nodes wait for a specified timeout interval before initiating the domain recovery algorithm. They send [trace-route] message to the root node. The root node traces the path down to the failed domain. The immediate parent to the failed domain then sends the graft information to the two children domains after which they bypass the failed domain and re-establish a communication chain from the root domain in the overall global hierarchy.

The domain recovery algorithm is capable of handling multiple domain failures. All the children domains of the failed domains would eventually initiate a failure recovery protocol and would send individual [trace-route] messages to the root domain. The children nodes would get relevant grafting information from appropriate ancestor nodes that would allow them to bypass the failed domain hence re-establishing the communication link with the root domain and rest of the domains in the global 'mDNS' hierarchy.

The root domain plays a pivotal role in the above described algorithm. If the root domain itself fails, all children domains would get disconnected with each other and the entire hierarchy would fall apart. To prevent such scenario, the root domain must provide ample redundancy with multiple backup MSD servers operating at all times with replicated URS as well. Such a requirement can not be expected from every domain. Thus the root domain designation can not be left to chance. The root domain has to be deliberately decided well in advance before the global overlay hierarchy starts taking shape with domains joining in arbitrarily.

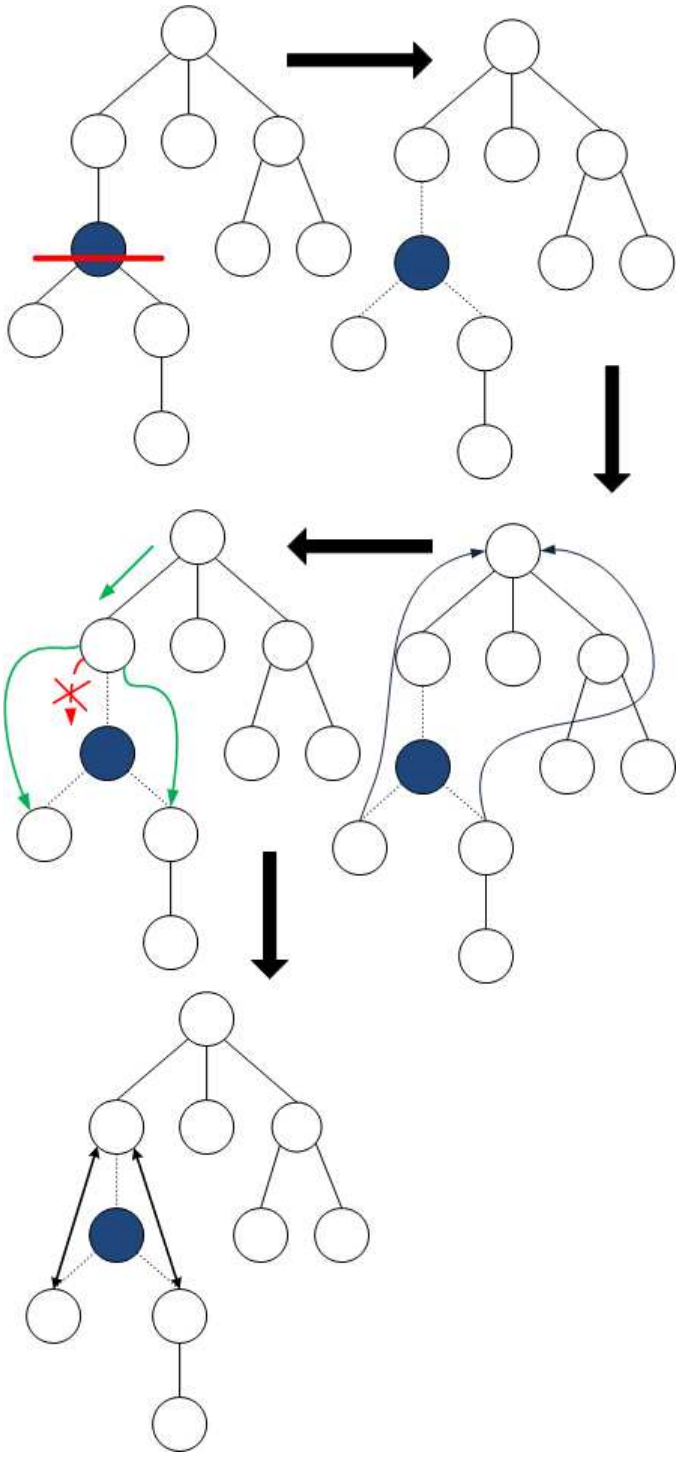

Fig. 5: mDNS domain failure recovery process

5. Impact on end-users

The timeout parameters after which a child domain could initiate the domain failure recovery process must not be set too large as it would adversely effect end users' experience. The time duration between failure and failure recovery would be the period where users in any active domain in the hierarchy could be affected as the portions of hierarchy would become unreachable due to non existence of path through the failed domains. Session databases stored in disconnected islands are rendered inaccessible to the other portions of the hierarchy. The impact of failure would be enormous for end users in the failed domain as they would be totally cut off from the distributed DHT hierarchy. They would be able to perform only domain specific searches.

The services would come to normal once the recovery algorithm has completed. Only hash-space database maintained at the failed domain would become inaccessible. Even then since the session records duplicates are cached as separate remote location using bit-inversion routing, those session records would still be accessible to end users in operational domains albeit somewhat larger search delays.

The QoS perception should return to normal level once the failed domain comes back online and the original hierarchy structure is restored. There would be some records migration from the graft location to the just recovered domain as the graft-location domain was temporarily responsible for managing the failed domain's hash space so any new session registration request would have been processed at that site.

6. Conclusion

In this paper we have analyzed a few 'mDNS' failure scenarios and what effect would those have on the end users' experience with the overall 'mDNS' global service infrastructure. We discussed in some details the effect of a total domain failure on service quality and how does a child domain of the failed node deal with its failed parent. The failure recovery strategy assumes a very high degree of availability on the root domain.

In our past research papers we gave details on the 'mDNS' construction, search routing and session geocoding strategies. Since the order of domain joins are not controlled, the worst case scenario could result in a linear DHT structure which will have adverse worst case search complexity. The next step would be to formulate strategy for self tree restructuring without involving system administrators. How difficult will it be to include options for administrators to specify peering rules during system startups would be an interesting study. We have already floated an IETF RFC document providing protocol structure and other details regarding 'mDNS' operations. These research tasks outlined here would help us make the 'mDNS' overlay more efficient and would help us guarantee better worst case complexities involved in various supported operations.

Acknowledgement

We would like to acknowledge Dr. Randy Chow for his helpful insight and discussions on our work. We would also like to thank CONS lab members for their participation in

numerous discussions on ‘mDNS’. Notable among those are InKwan Yu, Mahendra Kumar, and Panoat Chuchaisri.

References

[1] P. Harsh and R. Newman, “mdns - a proposal for hierarchical multicast session directory architecture,” in *International Conference on Internet Computing*, H. R. Arabnia and V. A. Clincy, Eds. CSREA Press, 2008, pp. 31–37.

[2] ——, “Using geo-spatial session tagging for smart multicast session discovery,” in *SAC*, S. Y. Shin and S. Ossowski, Eds. ACM, 2009, pp. 22–27.

[3] ——, “Efficient distributed search for multicast session keywords,” in *International Conference on Internet Computing*, H. R. Arabnia and V. A. Clincy, Eds. CSREA Press, 2009, pp. 67–73.

[4] ——, “Mode independent session directory service architecture,” Mar 2010, to appear in ACM SAC 2010. [Online]. Available: http://www.cise.ufl.edu/ pharsh/profile/content/SAC2010-NETS.pdf

[5] ——, *A Hierarchical Multicast Session Directory Service Architecture*, Nov 2009, internet Engineering Task Force, ID 19409.

[6] N. Sturtevant, N. Tang, and L. Zhang, “The information discovery graph: towards a scalable multimedia resource directory,” *Internet Applications, 1999. IEEE Workshop on*, pp. 72–79, Aug 1999.

[7] M. Handley, “The sdr session directory: An mbone conference scheduling and booking system,” April 1996. [Online]. Available: http://www-mice.cs.ucl.ac.uk/multimedia/software/sdr/

[8] A. Santos, J. Macedo, and V. Freitas, “Towards multicast session directory services.” [Online]. Available: citeseer.ist.psu.edu/264612.html

[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*, vol. 28, no. 1–2, pp. 119–125, December 1995.

[10] P. Liefooghe and M. Goosens, “The next generation ip multicast session directory,” *SCI, Orlando FL*, July 2003.

[11] P. Namburi and K. Sarac, “Multicast session announcements on top of ssm,” *Communications, 2004 IEEE International Conference on*, vol. 3, pp. 1446–1450 Vol.3, 20-24 June 2004.

[12] P. Mockapetris and K. J. Dunlap, “Development of the domain name system,” *SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 123–133, 1988.

[13] S. E. Deering, “Multicast routing in internetworks and extended lans,” in *SIGCOMM ’88: Symposium proceedings on Communications architectures and protocols*. New York, NY, USA: ACM, 1988, pp. 55–64.

[14] S. Bhattacharyya, *An Overview of Source-Specific Multicast (SSM)*, July 2003, internet Engineering Task Force, RFC 3569.

[15] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, *Internet Group Management Protocol, Version 3*, 2002, internet Engineering Task Force, RFC 3376.

[16] M. Mirakhorli, A. A. Sharifloo, and M. Abbaspour, “A novel method for leader election algorithm,” in *CIT ’07: Proceedings of the 7th IEEE International Conference on Computer and Information Technology*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 452–456.

[17] S. Dolev, A. Israeli, and S. Moran, “Uniform dynamic self-stabilizing leader election,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, no. 4, pp. 424–440, 1997.

[18] W. E. Burr, “Cryptographic hash standards: Where do we go from here?” *IEEE Security and Privacy*, vol. 4, no. 2, pp. 88–91, 2006.

[19] R. Rivest, “The md5 message-digest algorithm,” RFC 1321 (Informational), Apr 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1321.txt

[20] S. Raman and S. McCanne, “A model, analysis, and protocol framework for soft state-based communication,” *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 15–25, 1999.