Distributed Keyword Management Scheme:
Using Hashes and Routing
for efficient mDNS Global Search

Piyush Harsh
Computer and Information Science and Engineering
University of Florida
Gainesville, Florida 32611
Email: pharsh@cise.ufl.edu

Richard Newman
Computer and Information Science and Engineering
University of Florida
Gainesville, FL, USA
Email: nemo@cise.ufl.edu

Abstract—We proposed mDNS, which is a DNS aware, hierarchal, and scalable multicast session directory architecture that enables multicast sessions registration and makes them discoverable in real time. It supports domain specific as well as global searches for candidate sessions. In this paper we attempt to refine the mDNS global search algorithm and try to address various security and scalability concerns that remained in our earlier proposal. We propose distributing the overall keyword space among designated MSD servers using hash values.

Further we propose IP style prefix routing, but for keyword hashes, to locate appropriate MSD server in order to register / retrieve any globally scoped multicast session, in an efficient and fast manner.

I. INTRODUCTION

Multicast [1] currently operates in both ASM (Any Source Multicast) and SSM (Source Specific Multicast) [2] [3] mode. Many researchers as well as several ISPs favor SSM mode over ASM because it removes much of the network complexity that is part of ASM mode. Also IGMP version 3 [4] is specially suited for SSM. But with SSM, source discovery burden rests with the end users. It then becomes critical to come up with a global yet scalable mechanism to enable fast and convenient session discovery (or rather source) by the end users.

In our earlier work we analyzed several existing and upcoming proposals for maintaining multicast session database in the Internet. We studied in detail possibility of flaws in those proposals. Based on our identification of goals of such a architecture we proposed mDNS [5] which is a DNS aware, hierarchical and globally scalable multicast session directory architecture. It allows multicast session registration and facilitates real time discovery of such sessions by end users. It addressed many of the concerns that we found in competing proposals in the literature. Along with HOMA (Hybrid Overlay Multicast Address allocation) [6] which is a proposal for smart multicast address management, mDNS is capable in ushering native IP multicast into next level of deployment.

The Internet today is massive collection of nodes that operate as part of several independently managed administrative domains. Because of several access restrictions in many such domains, and also keeping in mind the operating scope of many multicast sessions i.e. global or local, mDNS search algorithm enables restricted and selective session discovery. Further since it is tightly integrated and co-located with existing DNS [7] hierarchy, it allows for the possibility of multicast sessions to be assigned URL style strings that end users can bookmark and retrieve later.

mDNS allows for two modes of search, namely domain specific and global search. The domain specific search is restricted to sessions originating from within the target domain. Global search allows for the Internet wide globally scoped multicast sessions to be searched and returned to the requesting end user. We proposed every MSD (Multicast Session Directory) servers hosted in any and every domain to be queried for sessions registered with it, that matched the search criteria, to be returned back directly to the querist.

But the drawback in current approach is, with increasing number of global searches, it puts more and more burden on MSD servers. Also even if a particular MSD server has no matching sessions, it is being activated under current scheme. And there are other security concerns particularly DoS (Denial of service) [8] attacks on any particular end user under current scheme.

In this paper we will try to address these concerns by proposing a smart workload division among MSD servers in various domains using one way hashes [9] and hash distribution techniques. Further we will propose a mechanism on the similar lines of IP unicast prefix based routing [10] technique that will enable fast routing of global sessions search query to appropriate MSD server. We believe such a proposal would greatly reduce the workload on MSD servers due to global searches. We will further modify the existing MSD search pseudocode towards our modified solution. We would also present a workload analysis in best case and worst case scenario among MSD servers for several keyword space division scenarios and end user search patterns. We will end the paper with the current project status and scope of further research in this field.
A. A brief introduction to mDNS

Briefly, in this section, we will describe the essential components of mDNS which will help put the hash distribution scheme we describe in later sections into perspective. mDNS stands for multicast enabled DNS. It makes use of existing DNS service in order to enable URLs for various multicast sessions. It allows sessions to be registered both locally and on global scale. And unlike other approaches, there is no latency in mDNS, i.e. sessions become discoverable as soon as they are registered.

1) mDNS administrative zone components: Figure 1 shows a typical mDNS network components in an administrative domain. It comprises of MSD (one or more) server(s), URS server and a DNS server with additional MCAST record entered to facilitate mDNS URL parsing. MSD servers are responsible for storing actual multicast session registration details for multicast sessions originating in their own domain. Additionally the designated MSD server in any domain (if there are more than one MSD server maintained) also stores session details for globally scoped sessions originating from foreign domains provided the keywords match those that fall under its jurisdiction. We will provide the details (keyword hashing and distribution) in later sections.

The URS (URL Registration Server) sole purpose is to maintain uniqueness among registered keyword within its own domain. Keywords may be duplicated among various domains, but within one domain, URS prevents duplication of keywords among two sessions originating at that domain. This is what guarantees the uniqueness among the several mDNS URLs and enforces one-to-one mapping of mDNS URLs and multicast sessions.

2) mDNS hierarchy: The additional MCAST record entry into DNS servers allow mDNS hierarchy to be constructed. For mDNS scheme to work properly DNS server is assumed to be present in every administrative domain in the Internet, or at least in those domains that wish to be part of mDNS hierarchy. A typical MCAST record looks like this -

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

If multiple MSD servers are maintained in a domain, they all are assigned a common anycast [11][12][13] address and that allows the mDNS URLs to be translated properly. PMCAST and CMCAST are the multicast channel addresses. They are assumed to be taken from GLOP [14] range and therefore are ISP assigned. Port numbers are assumed fixed and well known, possibly IANA assigned. The MSD-designated server joins both PMCAST and CMCAST channels and that allows it to form a hierarchy with MSD-designate servers in its children sub-domains and MSD-designate server at its parent domain and sibling domains under it’s parent domain. Figure 2 shows the typical mDNS hierarchical structure. It is this structure that allows searches to be routed to appropriate MSD servers and enables session discovery by the end users. The complete details of various components of mDNS including session registration, search algorithms and bootstrap process have been described in depth in our earlier work [5]. In the rest of this paper we will present enhancements to the global search algorithm to rectify the issues with the current scheme that we identified earlier.

II. mDNS GLOBAL SESSION SEARCH

In the current mDNS design, global searches are propagated on both CMCAST and PMCAST channels. Thus each search query is bound to reach every designated MSD servers in different domains almost surely. Once a search query reaches
a particular MSD server, it performs a database search and returns the candidate search results directly to the search querist. Clearly current scheme suffers from security and scalability loopholes. Depending on the number of MSD servers online, and using IP spoofing, an attacker can swamp any particular host on the Internet.

Since each MSD server does database search for every global search query that comes to it, and given that these searches could be computationally extremely taxing, MSD servers could not sustain large number of queries. Hence current scheme raises severe scalability concerns. Another argument against current scheme is, most of these MSD servers may not even have any relevant search results to provide, so why activate these seemingly wasteful database searches in the first place?

But nevertheless, the current scheme does prevent session registration data duplication, which has been one of the major design goals for mDNS. Can this goal be relaxed a little bit, such that, global session details can be permitted to be duplicated but somehow this duplication be bound? Furthermore can search algorithm be modified in a manner to direct keyword searches to specific MSD servers in the Internet rather than activating every MSD server in the Internet? In the next few subsections we will present a new approach to conduct global searches in mDNS that tries to answer these questions raised here.

A. Design Goals

mDNS was designed keeping usual Internet design principles in mind namely -

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

In addition to these general design criteria, there were a few service specific goals as well -

- make multicast sessions discoverable in real time,
- prevent multicast sessions’ details from being cached all over the Internet,
- make search mechanism sensitive to multicast scope limitations, and
- prevent enforcement of any restrictive session characterization semantics i.e. give session creators freedom in choosing keywords to define their session rather than compartmentalizing every session into a few number of predefined categories.

We have tried to uphold these design guidelines set for original mDNS proposal in our modified scheme as well.

B. Uniform distribution using Hash schemes

In order to direct global session registration data to a particular MSD server, it is necessary to use a function that would provide a uniform and consistent mapping of keywords into fixed length bit string regardless of input keyword length. The mapping must also appear random so as to allow an even distribution of keywords among existing MSD servers. This second property is especially important to maintain fairness in distribution (at least in the keyword distribution). Ensuring a particular server’s load fairness is hard to administer a priori as that is dependent on a particular keyword’s popularity among end users.

Hashes are a one way function that have the properties well suited for the task. Among the many hash schemes including MD2 [15], MD4 [16], MD5 [17], SHA2 [18] to name a few notable ones, we chose SHA2 for mDNS. SHA-256 does not suffer from collisions and APIs are easily available for various platforms.

The hash function is used to hash the keywords that a session creator provides during session registration. The 256 bit hash value is used to locate the appropriate MSD server in the mDNS hierarchy and the registration details are stored at that particular server. The locally scoped sessions data are stored locally at MSD servers in the originating domain.

C. Keywords Routing using Longest Prefix matching scheme

In order to route registration data to the correct MSD server, routing approach very similar to current IP unicast routing has been proposed. What upstream or downstream multicast channels (PMCAST or CMCAST) to route to depends on the longest matching hash prefix in the routing table maintained by every MSD-designate servers in the mDNS hierarchy. Essentially, the 256 bits hash values take on the role of 32 bits IP-addresses in the IP routing analogy.

In order to maintain some fairness, hash space distribution among participating MSD servers is done proportionately to the total number of participating MSD servers in the system and the number of MSD servers in a given subtree in the mDNS hierarchy. We have assumed that MSD servers will be long lived entities and mDNS hierarchy will be fairly stable and long lived over long periods of time. We believe that this assumption is fairly reasonable. Lets take an example scenario to better understand the routing table construction and routing of keywords to appropriate MSD servers. Figure 3 refers to an
example tree hierarchy formed by the MSD-designate servers in various domains and sub-domains in mDNS. Here node A represents the root DNS server and is the only node in the tree hierarchy that is not a MSD server. Hence it does not store any global session information but does help in the initial hash space division among its children MSD nodes.

1) Hash Space Division: In mDNS hierarchy, all children of a parent node join the PMCAST multicast channel, and similarly the parent node joins its CMCAST multicast channel. Needless to say, the CMCAST channel for the parent node and the PMCAST multicast channel for the children nodes are basically the same. This allows the parent node to push down instructions and status to its children nodes and similarly the children nodes are able to communicate between themselves and their parent node.

One of the pieces of information that is furnished to the parent node from its children is the number of MSD nodes in the subtree. This information in a similar fashion bubbles up to the root node and thus the root node knows how many MSD servers are maintained under each of its children domains. It uses the information to proportionately divide the hash space among its children domains. Figure 4 shows how the MSD count propagates up to the root node. The number next to each node represents the number of MSD-designate servers in the subtree rooted at that node (including that node itself). The only exception is the root node, where the count does not include it, because it does not maintain any session database and hence the hash space division does not include it.

In the example figures 3, 4 the 256 bits SHA-256 generated hash space is divided into 16 parts and the space divided into each child domain proportionately.

2) Routing Table Construction: During the process of hash space distribution, the parent node also constructs the routing table. Let us demonstrate how this is done taking the path of nodes from A to D via nodes B and C. We will use CIDR notation to specify groups of hash addresses in the routing table. The table shown for Node A i.e. Table I is just for clarity.

<table>
<thead>
<tr>
<th>Hash Address</th>
<th>Node ID</th>
<th>Channel Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xxxx...xx/1</td>
<td>Node B</td>
<td>CMCAST</td>
</tr>
<tr>
<td>1xxxx...xx/2</td>
<td>Node G</td>
<td>CMCAST</td>
</tr>
<tr>
<td>11xxxx...xx/3</td>
<td>Node G</td>
<td>CMCAST</td>
</tr>
<tr>
<td>111xxxx...xx/4</td>
<td>Node E</td>
<td>CMCAST</td>
</tr>
<tr>
<td>1111xxxx...xx/4</td>
<td>Node F</td>
<td>CMCAST</td>
</tr>
</tbody>
</table>

Since any hash query will go down via CMCAST channel at the root node, the routing table at root node i.e. A can contain just a single entry * signifying all remaining values (in this case every value) in the Hash Address column and Channel Out set to CMCAST.

Node B similarly divides the hash space allotted to it by the root node into 8 parts, assigns one part to itself and allots the remaining has space to downstream MSD servers. The example routing tables at Node B, C and D is also provided.

<table>
<thead>
<tr>
<th>Hash Address</th>
<th>Node ID</th>
<th>Channel Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000xxxx...xx/4</td>
<td>self</td>
<td>MSD-LOCAL-MCAST</td>
</tr>
<tr>
<td>0001xxxx...xx/4</td>
<td>Node H</td>
<td>CMCAST</td>
</tr>
<tr>
<td>0010xxxx...xx/4</td>
<td>Node I</td>
<td>CMCAST</td>
</tr>
<tr>
<td>0011xxxx...xx/4</td>
<td>Node C</td>
<td>CMCAST</td>
</tr>
<tr>
<td>01xxxx...xx/2</td>
<td>Node C</td>
<td>CMCAST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hash Address</th>
<th>Node ID</th>
<th>Channel Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111xxxx...xx/4</td>
<td>self</td>
<td>MSD-LOCAL-MCAST</td>
</tr>
<tr>
<td>0100xxxx...xx/4</td>
<td>Node M</td>
<td>CMCAST</td>
</tr>
<tr>
<td>0101xxxx...xx/4</td>
<td>Node D</td>
<td>CMCAST</td>
</tr>
<tr>
<td>0110xxxx...xx/4</td>
<td>Node N</td>
<td>CMCAST</td>
</tr>
<tr>
<td>0111xxxx...xx/4</td>
<td>Node O</td>
<td>CMCAST</td>
</tr>
</tbody>
</table>

Thus every MSD-designate server knows about the hash space it is responsible for. It maintains global multicast session registration database for all sessions whose keywords hash values maps to its own subset. It also constructs the routing
table in order to correctly direct the query along the path to appropriate MSD server. mDNS provides for possibility of multiple MSD servers at any domain. If multiple servers exist, they all subscribe to administratively scoped multicast channel MSD-LOCAL-MCAST. This provision provides for fault tolerance and load balancing in mDNS architecture at domain local level.

D. Session Registration and Retrieval

A new session registration request is always initiated on MSD-LOCAL-MCAST channel. The session, regardless of its scope, is always first registered with the domain local MSD servers. If the session is global in scope, he provided keywords are hashed individually using SHA-256 and the generated hash value is used to route the registration details to the appropriate MSD server in the mDNS hierarchy. Each MSD server along the route uses the hashed value and its own routing table to forward the registration request on either PMCAST or CMCAST channel. The routing table can be maintained in main memory for most cases and the hash value check against it can be done extremely fast. Once the registration request reaches the correct MSD server, which is responsible for hash space to which the candidate hash value belongs, the registration details are stored into the sessions database and the registration process finishes.

The session search also proceeds mostly along a similar line. End user initiates multicast session search by providing session keywords. If the search is for administratively scoped sessions only or is domain specific then the search algorithm is essentially same as original mDNS proposal. But if the search is a Internet wide search, then the keywords are hashed individually, and each hash value is used to route that particular search keyword to the target MSD server which stores session details for globally scoped multicast channels for that particular keyword. The relevant sessions are sent across from the remote MSD server directly to the end user / querist.

1) Quick Performance Analysis: How does the above modified proposal enhance the performance and scalability of mDNS architecture? In the original proposal, general Internet wide session searches activated every single MSD server in the global hierarchy. MSD servers performed database searches regardless of whether they had any relevant session stored with them or not. Also many MSD servers which possibly had a valid session detail, responded back to the requesting host, possibly overwhelming it with barrage of responses. This had major security implication, possibly an easy DDoS attack on any node in the Internet.

<table>
<thead>
<tr>
<th>Hash Address</th>
<th>Node ID</th>
<th>Channel Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101 22...25/4</td>
<td>self</td>
<td>MSD-LOCAL-MCAST</td>
</tr>
</tbody>
</table>

Algorithm 1: MSD session registration algorithm

```
begin
// always route based on transitional-next routing table if it exists
Read incoming session registration request
if request arrives on PMCAST channel then
  compare hash value with routing table entry
  if CHANNEL OUT = MSD-LOCAL-MCAST then
    register session
    make entry into local database
  else
    if CHANNEL OUT = PMCAST then
      drop request
    else
      // CHANNEL OUT = CMCAST
      route request on CMCAST channel
    end
  end
else
  if request arrives on CMCAST channel then
    compare hash value with routing table entry
    if CHANNEL OUT = MSD-LOCAL-MCAST then
      register session
      make entry into local database
    else
      if CHANNEL OUT = PMCAST then
        route request on PMCAST channel
      else
        // CHANNEL OUT = CMCAST
        drop request
      end
    end
  else
    // request arrives on MSD-LOCAL-MCAST
    register session
    make entry into local database
    if session scope = global then
      Data: keyword list
      foreach keyword in the list do
        compute SHA-256 hash
        compare hash value with routing table entry
        if CHANNEL OUT = MSD-LOCAL-MCAST then
          do nothing
        else
          route registration request on CHANNEL OUT
        end
      end
    end
end
```
Under the modified search scheme presented (which is very similar to pseudocode 1) here, the security concern is partially alleviated as now only a particular MSD server responds back to the end host instead of every single MSD server in mDNS hierarchy. Each MSD server along the routing path quickly verifies the hash value with the hash space it maintains and prunes the request or forwards it depending on routing table entry. Subtree rooted at parent where pruning takes place is never activated under proposed scheme. Comparing hash entry. Subtree rooted at parent where pruning takes place prunes the request or forwards it depending on routing table entry. Subtree rooted at parent where pruning takes place never becomes stale. The only scenarios where the number of MSD-count could change are when network administrative domains or subdomains are added or deleted. We hypothesize that these situations would be few and far between. Administratively independent network zones (domains / sub-domains) are not created or destroyed frequently. Such decisions have significant monetary implication on any organization or ISP and hence are taken usually as a long term decision. Hence we conjecture that the network domain hierarchy as a quasi-stable structure. And as a result mDNS MSD routing table structure should be generally stable over long time periods.

Still any change in the hierarchy could result in hash space redistribution from root node down to all the leaves and interior nodes. Hence we have decided to incorporate threshold based count propagation as a stabilizing measure.

A. Routing during transition phase

Every session registration gets a lease duration until which the data is maintained at the MSD servers. In order for the same registration data to remain valid, the registrant must renew the lease before the current one expires. There is a tradeoff between communication overhead and lease duration. Without loss of generality let us suppose that a typical lease period is $k$ days. Any registration entry at a given MSD may expire in one of two ways - if the session itself is no longer valid after a given point in time, and second if its lease expires and the session has failed to renew its lease within the expired lease period.

If due to some significant network hierarchy change, the hash space gets redistributed among existing MSD-designate servers, the old routing table is maintained until the next routing table update cycle, which is also a periodic update with periodicity $k$ days. The system makes sure that the time gap between existing hash space distribution scheme and next update is at least 1 period. The assignment is done (if needed) during a particular update cycle but the assignments achieve total validity from the next cycle.

In between these two cycle, the mDNS system goes into transition mode behavior. During the transition period, each MSD server depending on its own network load, transfers the registration records that would no longer belong to its future assigned hash values set to the appropriate MSD servers. As and when the registration records are transferred, they are being deleted as well from the old MSD servers. Once the transfer of records finishes, the MSD node which transferred all the records, marks its routing entry for that space as stale. If every routing entry at a node becomes stale, it sends an explicit prune request to its parent. The parent node once it receives the explicit prune request from one of its child nodes, marks the corresponding routing entry in the current (soon to be stale) table as stale as well and drops future search requests belonging to that child node’s space. This explicit pruning behavior is recursive in nature.

The registration requests always are routed based on the newer hash space distribution if one exists (or transitional-next routing table). The search queries during the transition period are routed to both the assigned MSD server according to newer scheme as well as the existing (soon to be stale) routing table during the transition period. As soon as the transition cycle has expired, the old invalid routing entries are purged and replaced with the routing table that reflects the newer scheme and the system proceeds from the transition mode to stable mode behavior.

Although this scheme allows for sessions to remain discoverable during transition phases, we would like this transition phase to be as few as possible. In order to achieve this goal, we propose to use threshold based algorithm to determine how each MSD server reports the MSD count up to it’s parent node. Algorithm 2 describes the normal mode search process. Transitional mode search is very similar to normal mode search algorithm, except both current as well as transitional-next routing tables are checked. If any current routing table entry is marked STALE, the corresponding action (database search or routing) is not taken in transitional mode.
Algorithm 2: MSD normal mode search algorithm

begin
    Read boolean mode flag
    if mode = NORMAL then
        if search arrives on MSD-LOCAL-MCAST then
            // search came from one of nodes in self domain
            check routing table
            if CHANNEL OUT = MSD-LOCAL-MCAST then
                search database
                return search result
            else if CHANNEL OUT = CMCAST then
                route request on CMCAST
            else
                route request on PMCAST
            end
        else if search arrives on PMCAST then
            // search came from one of the child sub-domains
            check routing table
            if CHANNEL OUT = MSD-LOCAL-MCAST then
                search database
                return search result
            else if CHANNEL OUT = CMCAST then
                // narrow search diameter
                route request on CMCAST
            else
                drop the query
            end
        else
            // mode = TRANSITIONAL
            follow transitional mode search pseudocode
        end
    else
        // mode = TRANSITIONAL
        follow transitional mode search pseudocode
    end
end

B. Threshold scheme and its implications

In order to reduce mDNS architecture transitional phase frequency, any MSD-designate node count reporting behavior is governed by simple threshold based algorithm. Two threshold values, $\alpha$ and $\beta$ are used with $0 < \alpha < \beta < 50\%$.

Algorithm 3: Node count reporting algorithm

begin
    set values for $\alpha$ and $\beta$
    set $COUNT_a = $ current count
    set $COUNT_b = $ current count
    begin periodic behavior
        listen for count reporting from children
        update $COUNT_b$
        if $\frac{COUNT_b - COUNT_a \times 100}{COUNT_a} \leq \alpha$ then
            do nothing
            if mode = TRANSITIONAL then
                set mode = NORMAL
                purge stale routing table
            end
        else if $\alpha < \frac{COUNT_b - COUNT_a \times 100}{COUNT_a} \leq \beta$ then
            do proportional hash space reassignment among children domains and self
            create new routing table
            set mode to TRANSITIONAL
        else
            report $COUNT_b$ to parent node
            set $COUNT_a = COUNT_b$
        end
        if new hash space assignment comes from parent then
            create new routing table
            set mode to TRANSITIONAL
        end
    end
end

Once the initial mDNS hierarchy buildup stage is over, we conjecture that algorithm 3 will result in more stable and fewer mDNS transitions. The values of $\alpha$ and $\beta$ are tunable and more experimentation is needed to decide on an optimal setting.

C. Redundancy

To improve the reliability and availability of mDNS in the face of partial network outages and temporary network split, a simple redundancy scheme has been proposed. When a client initiates session registration, the registration request for globally scoped session is routed according to SHA-256 hash of the keywords, but a duplicate registration request is automatically routed with the generated hash bits inverted. The same is done whenever any registration lease is renewed. In rare cases where the original hash and its inversion lies within the same hash space allotted to a particular MSD server, both requests goes to the same server and there is no redundancy. Generally the duplicate registration request will get routed to a different server under different portion of the mDNS hierarchy.

During global search, if usual keyword hash is routed to a MSD server which is unavailable, the client software does
an additional search with hash bits inverted. We hope with this scheme, unless there is a large scale network outage, global sessions would continue to remain discoverable at minor network outages.

IV. Performance Evaluation

Suppose at the root level, number of child domain is $k$. And the total count of MSD-designate servers is $N$.

$$\sum_{i=1}^{k} n_i = N$$

(1)

Hash space to divide among these servers: $2^{256}$. Let the significant bits used for routing between MSD servers be $m$. Then -

$$2^m \geq N \quad \text{or} \quad m \geq \log_2 N$$

(2)

Each child domain at the root level gets $n_i \times 2^m \times 2^{(256-m)}$ part of the total hash space. Hence the share allotted to domain $i$ becomes -

$$share_i : \frac{n_i}{N} \times 2^{256}$$

(3)

Since now each domain distributes their allotted space among the total MSD-designate count at that node, each MSD server’s share becomes: $share_i \div n_i$. After simplification, each MSD server’s hash-space share comes out to be -

$$share_{MSD} : \frac{n_i}{N} \times 2^{256} \div n_i \quad \text{or} \quad \frac{1}{N} \times 2^{256}$$

(4)

If mDNS hierarchy remains stable, then hash space is shared uniformly among every designated MSD server in the system. As more servers are added, this hash space division could become non-uniform until the next top-down reassignment is done starting at the root node.

A node that appears higher in the mDNS hierarchy, does more query routing compared to a node at lower level. Routing load depends on the MSD count at that particular node. At any node $a$, suppose there are $j$ children domains, then routing load becomes

$$load_a = \frac{1}{N} \times \sum_{i=1}^{j} count_i + 1$$

(5)

where $count_i$ is the MSD count propagated to it from its child $i$.

If every keyword is equally likely to be searched, then workload for any MSD node “a” in the hierarchy over a period of time $t$ becomes -

$$Workload = t \times rate_{query} \times probability_{range} \times \frac{share_{MSD}}{2^{256}}$$

Using equation (4), we get

$$Workload = t \times rate_{query} \times \frac{1}{N} \times 2^{256}$$

(6)

Hence under mDNS, the query workload is also evenly distributed provided each search keyword is equally likely to be queried.
discussions. The authors would like to thank the anonymous reviewers who have reviewed this paper and provided useful feedback. We would also like to thank Dr. Shigang Chen, Dr. Randy Chow and Dr. Arunava Banerjee for their continuous support and engaging us in periodic discussions during various phases of mDNS research.

REFERENCES