Handling Identity in Peer-to-Peer Systems*

Manfred Hauswirth, Anwitaman Datta, Karl Aberer
Distributed Information Systems Laboratory
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{Manfred.Hauswirth, Anwitaman.Datta, Karl.Aberer}@epfl.ch

Abstract

Due to the limited number of available IP addresses most computers on the Internet use dynamic IP addresses which causes problems for applications that have to maintain routing tables, for example, peer-to-peer systems. To overcome this we propose unique peer identifiers in the routing tables and apply the peer-to-peer system itself to maintain consistent id-to-IP mappings to be used in the routing process. While this may sound like a recursive hen-egg problem we show that it is in fact possible to devise such a mapping service for realistic scenarios. Our approach is completely decentralized, self-maintaining, and light-weight. It takes into account security to provide sufficient security guarantees for the mappings. We also assume that the service operates in an environment with low online probability of the peers constituting the service.

1. Introduction

The necessity to use dynamic IP addresses due to the limited number of available addresses causes problems for applications that have to maintain routing tables for their operation. For example, in advanced P2P systems ad-hoc connections to peers have to be established in the query routing process which is only possible if the receiving peer has a permanent IP address. This problem would be solved if Mobile IP [12] or IPv6 [13] were in place and available at a large scale because they take into account mobility (dynamicity) and offer a much larger address space. However, this requires considerable changes in the networking infrastructure of the complete Internet and it cannot be foreseen at the moment when this will happen. Thus other solutions like the approach presented in this paper are required to overcome the problem in the meanwhile.

As most P2P systems our P-Grid P2P lookup system [2] suffers from the dynamic IP address problem because this causes routing tables to become inconsistent. Only very few systems such as Gnutella and FastTrack do not suffer from this problem but pay this with other drawbacks such as considerable network bandwidth consumption or limitations in their structure and applicability. In this paper we propose to use the underlying P2P system itself (here P-Grid) to keep track of IP address changes by mapping unique peer identifications consistently onto peers' IP addresses in the P2P system. The routing tables would then hold unique peer identifications that would be mapped onto the up-to-date IP address of the peer to be contacted. Mappings would be cached and the service would only be queried if a stale mapping was detected. This may indeed sound like a hen-egg problem: The system depending on mappings for its operation is applied to store the mappings. However, we show that this is possible under real-world assumptions regarding IP address change rates and online availability of the peers. We describe the protocol and query processing strategies especially taking into account security since the mappings are stored in a completely decentralized system and evaluate the efficiency of our approach. The security concept of our approach is a combination of PGP-like public key distribution and a quorum-based query scheme.

The paper is organized as follows: We first describe the problems of dynamic IP addresses in more detail in Section 2. Then we provide the essential characteristics of P-Grid in Section 3 which we use both as the implementing and applying system as a proof-of-concept. Section 4 then defines our proposed protocol and Section 5 illustrates the querying strategy of P-Grid modified according to the protocol so that dynamic IP addresses are accounted for. Section 6 then gives some analytical results of our evaluations. Section7 discusses related approaches and we conclude in Section 8 by giving out conclusions.

2. Dynamic IP addresses

The limited number of IP addresses is currently addressed by two concepts: Dynamic Host Configuration Protocol (DHCP) and Network Address Translation (NAT). In DHCP [6] a DHCP server maintains available IP addresses which can be requested by clients for a certain time *lease time* to be used. In NAT [9] a NAT router maps non Internet-routable IP addresses onto routable IP addresses back and forth. The most frequently used configuration is that the NAT router has an official IP address and all the comput-

*The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

ers in the local network have non-routable ones. Many ISPs combine both technologies. In the following we focus on the problem of dynamic routable addresses. NAT is a general problem of P2P systems because it only supports uni-directional connection establishment and is beyond the scope of this paper.

The problems to address are: (1) How can universally unique identifiers be mapped onto physical addresses in a secure, decentralized, and efficient way? (2) With the possibility of changes of physical addresses a peer must be able to detect whether it is still talking to the same entity it intends to talk with. Gnutella is not affected by the first problem since peers actively announce their availability but at the cost of high bandwidth consumption because of its constrained broadcast approach. However, any peer-to-peer system actually should address the second problem for security reasons to avoid rather simple denial-of-service (DOS) attacks.

3. P-Grid in a nutshell

Since our approach is based on our P-Grid P2P lookup system we will briefly introduce its basic concepts. it. P-Grid [2] is based on a virtual distributed search tree: Each peer only holds part of the overall tree, which comes into existence only through the cooperation of individual peers. Searching in P-Grid is efficient and fast even for unbalanced trees [1] ($O(\log(n))$, where n is the number of leaves). Unlike many other peer-to-peer systems P-Grid is a truly decentralized system which does not require central coordination or knowledge. It is based purely on randomized algorithms and interactions. Also we assume peers to fail frequently and be online with a very low probability. Figure 1 shows a simple P-Grid.

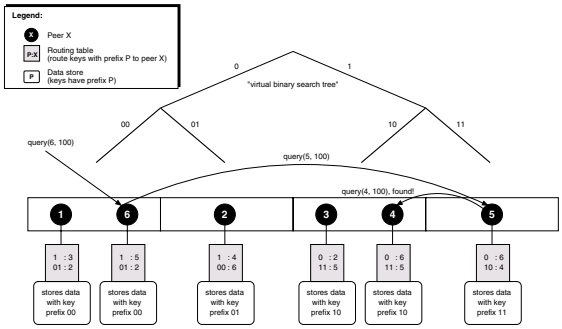

Figure 1. Example P-Grid

Every participating peer's position is determined by its path, that is, the binary bit string representing the subset of the tree's overall information that the peer is responsible for. For example, the path of Peer 4 in Figure 1 is 10, so it stores all data items whose keys begin with 10. For fault-tolerance multiple peers can be responsible for the same path, for example, Peer 1 and Peer 6. P-Grid's query routing approach is simple but efficient: For each bit in its path, a peer stores a reference to at least one other peer that is responsible for the other side of the binary tree at that level. Thus, if a peer receives a binary query string it cannot satisfy, it must forward the query to a peer that is "closer" to the result. In Figure 1, Peer 1 forwards queries starting with 1 to Peer 3, which is in Peer 1's routing table and whose path starts with 1. Peer 3 can either satisfy the query or forward it to another peer, depending on the next bits of the query. If Peer 1 gets a query starting with 0, and the next bit of the query is also 0, it is responsible for the query. If the next bit is 1, however, Peer 1 will check its routing table and forward the query to Peer 2, whose path starts with 01. The P-Grid construction algorithm [2] guarantees that peer routing tables always provide at least one path from any peer receiving a request to one of the peers holding a replica so that any query can be satisfied regardless of the peer queried.

4. Protocol

This section defines the protocol for maintaining id-to-IP mappings in P-Grid. In short the protocol works as follows: Peers generate universally unique identifications locally and store them along with their public key, their current IP address and a cryptographic signature in P-Grid, i.e., on a certain number of peers since P-Grid replicates the stored information. Mapping an id onto an IP address then is done by querying P-Grid using the receiver's id as the key. If a certain quorum of identical answers is returned the mapping is considered trustworthy and the peer is contacted. If contacting the peer fails then the peer is either offline or has changed its IP address. The requester can now either assume that the peer is offline and give up or, in the latter case, submit a new query to determine the new IP address. If contacting the peer succeeds in either case, its public key is used to determine whether the contacted peer really is the one identified by the mapping or whether a different peer reuses the address or a malicious peer tries an impersonation attack. Our security concept is a combination of PGP-like public key distribution and a quorum-based query scheme.

In detail the algorithm works as follows: Each peer p is uniquely identified by a universally unique identifier (UUID) Id_p. This identifier is generated once at installation time by applying a cryptographically secure hash function to the concatenated values of the current date and time, the current IP address $addr_p$ and a large random number. Routing tables only hold these identifiers. Each peer p additionally has a cache of mappings $(Id_i, addr_i, TS_i)$ (TS_i denotes a timestamp) that it already knows. At bootstrap each peer p also generates a private/public key pair D_p/E_p once. Then the algorithm for handling dynamic IP addresses works as follows (inserts and updates are done according to the algorithm presented in [5]):

Bootstrap

1. p generates Id_p, D_p/E_p.
2. At startup p determines its current IP address.
3. $(Id_p, addr_p, E_p, TS_p, D_p(Id_p, addr_p, E_p, TS_p))$ (for brevity denoted as $tuple$ in the following is inserted by p into P-Grid using Id_p as the key (TS_p prevents

replay attacks). Inserting in P-Grid means that the request is routed to a peer $R_i \in \Re_p$. $\Re_p$ is the set of replicas responsible for the binary path using Id_p as the key value ($path(Id_p)$). If Id_p already exists in the P-Grid (though this is very unlikely) p is notified. If so, p generates a new Id_p and repeats this step.

4. The previous step is repeated R_{min} times and p waits for confirmation messages from R_{min} distinct peers to prevent a malicious peer in $\Re_p$ from distributing false data to the other replicas in $\Re_p$.
5. As a result of the previous two steps the mapping will be physically stored at peers in $\Re_p$. Based on the randomized algorithms that P-Grid uses we can assume that the individual replicas $R_i \in \Re_p$ are independent and they collude or behave Byzantine only to a degree that can be handled by existing algorithms.

Peer startup

1. p starts up and checks whether its $addr_p$ has changed. If not the algorithm terminates. Otherwise the following steps are taken.
2. A new mapping and a signature for this mapping $(Id_p, addr_p, TS_p, D_p(Id_p, addr_p, TS_p))$ is sent as an update message to the P-Grid by p.
3. Upon receiving the update request the R_i check the signature by verifying that $(E_p(D_p(Id_p, addr_p, TS_p)).Id_p = Id_p$ (thus only p can update its mapping) and $TS_{R_i} < TS_p$ (to prevent replay attacks). If yes, the new mapping is stored, otherwise an error message is returned.

Operation phase

p is up and running, has registered an up-to-date mapping $(Id_p, addr_p, TS_p)$ and is ready to process requests.

1. p receives a request Q from a peer q.
2. In case p can satisfy Q the result is returned to q. Otherwise p finds out which peers p_f to forward the query to according to P-Grid's routing strategy. Then it checks its routing table and retrieves $(Id_{p_f}, addr_{p_f}, E_{p_f}, TS_{p_f})$ which had been entered during the construction of P-Grid.
3. p generates a random number r, contacts p_f and sends $E_{p_f}(r)$. As an answer p_f must send $(D_{p_f}(E_{p_f}(r)))$ and q can check whether $D_{p_f}(E_{p_f}(r)) = r$. If yes, p_f is correctly identified, i.e., p really talks to the peer it intends to, and Q is forwarded to p_f.
4. If not, then p_f has a new IP address (the case that somebody tries to impersonate p_f is covered implicitly by the signature check above) and p sends a query to P-Grid to retrieve the current $addr_{p_f}$ using Id_{p_f} as the key.
5. p collects all answers $t_i = (Id_{p_f}, E_{p_f}, addr_{p_f}, TS_{p_f}, D_{p_f}(Id_{p_f}, E_{p_f}, addr_{p_f}, TS_{p_f}))$ it receives from the $R_j \in \Re_{p_f}$ (if extended security is required then the R_j should sign their answers, i.e., send $(t_i, D_{R_j}(t_i))$). p has to collect at least R_{min} answers to detect misinformed or malicious peers, i.e., checks whether a certain quorum of the answers is identical (R_{min} is defined by each individual p according to its local requirements for trustworthiness of the reply). Otherwise the query is repeated a certain number of times before aborting.
   - As an optimization the quorum can be avoided under certain circumstances. If p already knows E_{p_f}, e.g., from the construction of the P-Grid or because it has already done a certain number of (quorum-based) queries for E_{p_f} that have resulted in identical answers, so that it can assume that its E_{p_f}, then it can immediately check the validity of the answer by $E_{p_f}(D_{p_f}(Id_{p_f}, E_{p_f}, addr_{p_f}, TS_{p_f})).Id_{p_f} = t_i.D_{p_f}$.
   - The scheme can be further optimized (and made more robust and secure) by having all peers store the E_p's that they receive.
6. Now p can proceed with step 3 and if this is successful p enters $(Id_{p_f}, addr_{p_f}, E_{p_f}, TS_{p_f}))$ in its local cache.

5. Processing queries

This section illustrates how the query processing will work with the protocol of Section 4 in place. Figure 2 shows a typical state of a P-Grid after some processing.

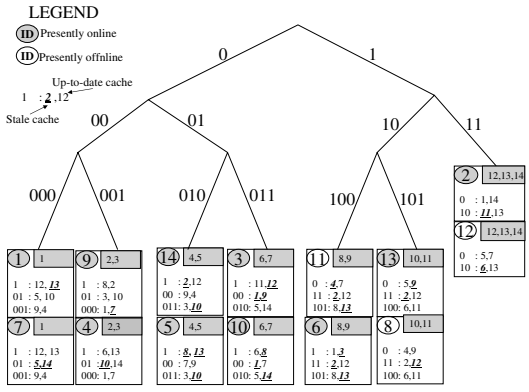

Figure 2. P-Grid before Query(01*) at P_7

Peer P_i is denoted by i inside an oval. Online peers are indicated by shaded ovals, offline peers by unshaded ovals. We will use a query $Q(01*)$ at P_7 for our example. P_7 holds the public key and latest physical address mapping about P_1 (updated by P_1). These are shown in the shaded rectangle in the upper-right corner. In this example, we follow the convention that the peers are represented by paths of a length of 4 bits. For example, information about P_1 (e.g, its public key or latest id-to-IP mapping) can be obtained by $Q(P_1)$, i.e., $Q(0001)$. Under this convention, since P_7 is responsible for the search path 000, it stores information relevant to 000, and routing references for paths starting with 1, 01 and 001, so that queries with these prefixes may be forwarded to respective peers for further processing. These references to other side of the P-Grid subtrees at all depths form P_7's

routing table. The cached physical address of these references may be up-to-date (for example P_{12}) or stale (staled entries are underlined, for example, $\underline{P_5}$).

A peer P_q decides that it has failed to contact peer P_s if one of the following happens: (1) No peer is available at the cached address. In this case, P_q trivially determines that P_s is unavailable. (2) The contacted peer fails in the authentication: If any peer $P_{s'}$ is present at the physical address as cached by P_q for P_s, P_q will use P_s's public key to verify whether $P_{s'}$ is indeed P_s. If $P_{s'}$ fails the identity test, P_q concludes that it has failed to contact P_s. Since only P_s knows its private key, only P_s can pass the identity test.

In either case an up-to-date mapping must be obtained by querying the P-Grid. We have investigated two querying strategies:

Isolated-Query: Upon receiving a query a peer checks whether it can answer the requests or else forwards the query to at least one of the peers in its routing table according to P-Grid's routing algorithm. If none of these peers can be contacted, the query is abandoned and fails.

Recursive-Query: If a peer fails to contact any of the peers in its routing table, it initiates a new query to discover the latest "identity-to-address" mapping of the peers in the routing table, and if such a peer can be located, the query is then continued (forwarded).

While the P-Grid is in the state as shown in Figure 2, assume that P_7 receives a query $Q(01*)$. The query may be for searching any information in P-Grid, either information about some participating peer or any other information. In this example situation, P_7 fails to forward the query to P_5 and P_{14} since the cache entries are stale. The Isolated-Query algorithm fails immediately.

In the recursive query version, a peer that has failed to contact any of the peers to which it could forward the query, first tries to discover the latest addresses for those routing table entries. In our example, P_7 initiates Recursive-Query(P_5), i.e., $Q(0101)$, which needs to be forwarded to either P_5 or P_{14}. This fails again. P_7 may then initiate Recursive-Query(P_{14}), i.e., ($Q(1110)$), which needs to be forwarded to P_{12} and (or) P_{13}. P_{12} is off-line, so irrespective of the cache being stale or up-to-date, $Q(P_{14})$ fails to be forwarded to P_{12}. P_{13} is online, and the cached physical address of P_{13} at P_7 is up-to-date, so $Q(P_{14})$ is successfully forwarded to P_{13}. P_{13} needs to forward $Q(P_{14})$ either to P_2 or P_{12}. It fails to forward it to P_{12}. Further, P_{13} fails to forward it to P_2 because its cached entry for P_2 is stale. P_{13} thus initiates another sub-query, namely Recursive-Query(P_2), i.e., ($Q(0010)$). It may also initiate Recursive-Query(P_{12}) as well. From P_{13}, $Q(P_2)$ is forwarded to P_5. From P_5, $Q(P_2)$ is forwarded to one of P_7 and P_9. Assume P_9 replies (though in parallel it may be forwarded from P_7 to P_4, P_9 and then eventually be answered). Thus P_{13} learns P_2's location and updates it in its cache. P_{13} also starts processing and forwards the parent Recursive-Query(P_{14}) to P_2. P_2 provides P_{14}'s up-to-date address, and P_7 updates it in its cache (directly or via P_{13}, depending on the implementation).

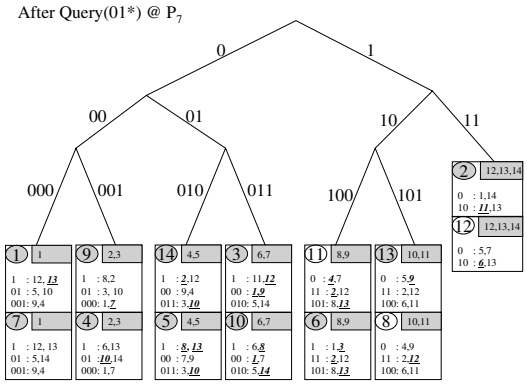

Figure 3. P-Grid after Query(01*) at P_7

Having learned P_{14}'s current physical address, P_7 now forwards the original query $Q(01*)$ to P_{14}. In this case, not only is the original query satisfied, but also P_7 has an opportunity to learn and update P_5's physical address, since P_{14} is responsible for P_5's latest physical address. Thus, apart from successfully replying to the original query, P_7 updates the physical address for P_{14}, and possibly of P_5. Further, because of the initiated child queries, P_{13} updates its cached address for P_2. The final state with several caches updated after the end of $Q(01*)$ at P_7, is shown in Figure 3.

6. Analytical results

Due to the limited space available we will only provide some results from our analysis. The complete analysis and results can be found in [11]. We investigate the performance of queries without and with recursion, and study the improvement in the success rate and the additional effort incurred, as the system parameters change. We give results for the case where the P-grid tree has $n = 2^7$ leaves, and all peers have 4 references cached at any depth.

Figure 4 shows the failure probability of queries under the assumption that peers are online with probability $p_{on} = 0.8$ while varying the probability of local cache entries being stale (p_{dyn}).

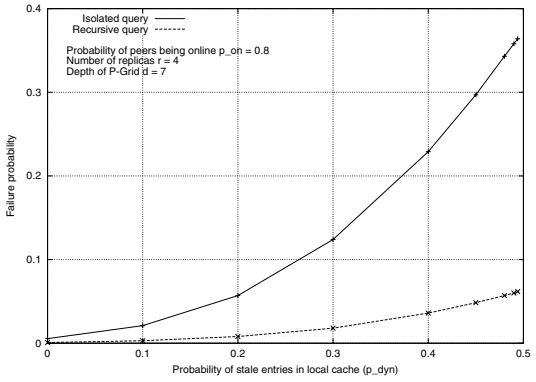

Figure 4. Probability of query failure

The failure probability of non-recursive queries increases rapidly with an increase in p_{dyn}. With recursion,

however, the failure probability is significantly lower, since intermediate failures trigger recursive queries, leading to self-healing effects, and thus to the eventual success of the original query. The benefits of the recursion, are dual, since apart from reducing the probability of failure it induces self-healing, i.e., rectifying stale cache entries.

Figure 5 shows the the increased effort by a factor in terms of the messages in the recursive version of the query (Y-axis) with variation of p_{dyn} for $p_{on} = 0.6$ and $p_{on} = 0.8$ The expected effort for the isolated-query in these cases is fixed, and equals 14 messages.

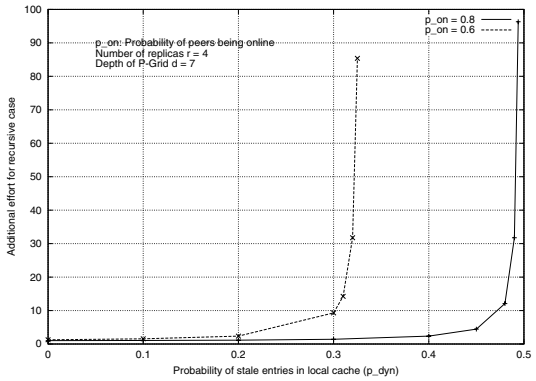

Figure 5. Additional effort for recursion

The results are intuitive: With higher p_{on} and lower p_{dyn}, failure rates are low, and the additional effort in using recursion is marginal. Detailed results on further failure probabilities and self-healing properties of the algorithm and are provided in [11].

7 Related Work

Freenet [4] originally proposed address resolution keys stored inside Freenet itself to cope with dynamic IP addresses. However, this approach has been given up in favor of a dynamic DNS service such as DynDNS.org. DynDNS.org [7] is a service that allows a user to map a dynamic address to a static hostname in some domains and to update this mapping via a HTTP interaction protected by a username/password scheme. Besides DynDNS many similar services exist [14]. In theory even DNS [3] itself could now be used for maintaining dynamic IP addresses. [16] added support for dynamically updating a nameserver's database to the original design of DNS. Access to the database is based on the IP address of the requester. [15] added transactional signatures (TSIG) based on symmetric cryptography to make updates more secure and [8] finally introduced a fully-flexed infrastructure for secure DNS updates based on public-key cryptography. However, all these schemes require access to name servers and incur a considerable configuration and management overhead. In order to handle dynamic physical addresses securely we introduced a self-organizing public key infrastructure. PGP [10] is comparable to our concepts because it offers a similar, decentralized approach.

8. Conclusions

This paper described a decentralized, self-maintaining, light-weight, and secure name service that applies the same P2P system that uses the name service for maintaining the service's database. We have demonstrated that such an self-referencing approach is possible and that our algorithm is robust and applicable in unreliable environments and operates well even if we assume low online probabilities. The service is based on the P-Grid P2P system and applied in P-Grid itself to remedy the problem of dynamic peers addresses in routing tables. We have defined a generic protocol that can also be applied and other systems, described the query processing strategy to support the protocol and provided analytical results on the performance of the algorithm. The service offers a sufficient level of security by combining a PGP-like approach for circulating public keys with a quorum-based query scheme that provides robustness against cheating peers.

References

[1] K. Aberer. Scalable Data Access in P2P Systems Using Unbalanced Search Trees. In *WDAS*, 2002.

[2] K. Aberer, M. Hauswirth, M. Punceva, and R. Schmidt. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6(1), 2002.

[3] P. Albitz and C. Liu. *DNS and BIND*. O'Reilly & Associates, 2001.

[4] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, number 2009 in LNCS, 2001.

[5] A. Datta, M. Hauswirth, and K. Aberer. Updates in Highly Unreliable, Replicated Peer-to-Peer Systems. In *ICDCS*, 2003.

[6] R. Droms. *Dynamic Host Configuration Protocol*. RFC2131. Network Working Group, IETF, 1997.

[7] Dynamic DNS Network Services, 2003. http://www.dyndns.org/.

[8] D. Eastlake. *Domain Name System Security Extensions*. RFC2535. Network Working Group, IETF, 1999.

[9] K. Egevang and P. Francis. *The IP Network Address Translator (NAT)*. RFC1631. Network Working Group, IETF, 1994.

[10] S. Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly & Associates, 1994.

[11] M. Hauswirth, A. Datta, and K. Aberer. Handling Identity in Peer-to-Peer Systems. Technical Report IC/2002/67, Ecole Polytechnique Fédérale de Lausanne (EPFL), 2002.

[12] C. E. Perkins, B. Woolf, and S. R. Alpert. *Mobile IP Design Principles and Practices*. Prentice Hall PTR, 1998.

[13] P. H. Salus, editor. *Big Book of IPv6 Addressing RFCs (Big Book)*. Morgan Kaufmann, 2000.

[14] D. E. Smith. Dynamic DNS, May 2002. http://www.technopagan.org/dynamic/.

[15] P. Vixie, O. Gudmundsson, D. E. 3rd, and B. Wellington. *Secret Key Transaction Authentication for DNS (TSIG)*. RFC2846. Network Working Group, IETF, 2000.

[16] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. *Dynamic Updates in the Domain Name System (DNS UPDATE)*. RFC2136. Network Working Group, IETF, 1997.