Development of firmware update mechanism for IoT devices using DNS and IPv6

Atharva Rajendra Karpate

SRCOEM, Ramdeo Tekdi, Katol Rd, Gittikhadan, Nagpur, Maharashtra 440013

Dr. Bheemarjuna Reddy Tamma

Department of Computer Science and Engineering, IIT-Hyderabad, Kandi, Telangana 502285

Table of Contents

Development of firmware update mechanism for IoT devices using DNS and IPv6

Atharva Rajendra Karpate

SRCOEM, Ramdeo Tekdi, Katol Rd, Gittikhadan, Nagpur, Maharashtra 440013

Dr. Bheemarjuna Reddy Tamma

Department of Computer Science and Engineering, IIT-Hyderabad, Kandi, Telangana 502285

Abstract

The Internet of Things (IoT) has revolutionized computing with a multitude of applications being built around various types of sensors. That includes an extraordinary number of objects of all shapes and sizes – from smart microwaves, which automatically cook food, to self-driving cars, whose complex sensors detect objects in their path. A vast amount of activity is seen in IoT based product-lines and this activity is expected to grow in the years to come with projections of up to a Trillion connected devices. With this exponential projected growth in the number of IoT devices, security of these resource constrained and often remotely located devices becomes paramount. Most of the IoT devices come with factory installed firmware. The firmware of IoT devices is not generally updated during their lifetime. So, they are more prone to attacks which may lead to severe problems. Providing timely firmware updates can greatly reduce the security risk for IoT devices. The domain name system is one of the foundations of the internet, which is mainly used to store and retrieve IP addresses from domain names. DNS supports various record types such as A, AAAA,etc. which store different kinds of information about the queried domain name. One such newly proposed record type is the OX record type. The project aims to investigate the suitability of the OX record type with its various available fields to securely update the firmware of IoT devices. As the number of devices on the internet increase, the world is slowly moving from the old IPv4 to the new Ipv6 protocol. Hence, this project is mainly being implemented for Ipv6 based IoT devices.

Keywords: DNS, IPv6, Internet of Things (IoT), OTA update, Object Exchange (OX RR).

Abbreviations

Abbreviations Table

DNS	Domain Name System
IoT	Internet of Things
OX	Object Exchange
RR	Resource Record
OTA	Over the Air
IPv6	Internet Protocol version 6
SHA-2	Secure Hash Algorithm-2
NXDOMAIN	Non-existent Domain Name
DNSSEC	Domain Name System Security Extensions

1 INTRODUCTION

1.1 *Background & Rationale*

From remotely checking in on a smart factory, to conveniently working from home, to relaxing in a driverless car, the Internet of Things (IoT) is paving the way for many exciting new opportunities. It is predicted that the IoT market will double by the year 2021, reaching a staggering $520 bn in market cap[1]. The number of IoT devices is predicted to increase exponentially in the coming decade. A common issue faced by these IoT devices is that of firmware updates. As these devices are often resource constrained and do not possess much computational power. A light-weight update mechanism is required to provide firmware updates for these devices. Timely firmware updates would not only ensure security but can also be used to re-configure the device or add new functionality.

Most of the IoT manufacturers and IoT platform providers such as Bosch IoT [2] and IBM Watson [3] have their own PUSH based server initiated firmware update mechanisms and these vary widely across all devices and manufacturers. Server-initiated firmware update mechanism requires the manufacturer to keep up-to-date and accurate status information of each device. Also, when a firmware download is server-initiated, the status tracker tells the IoT device when to download and install the new firmware. Inaccurate information on the server side could result in unwanted or adverse consequences, especially for health monitoring or life-supporting devices. A client-initiated PULL based firmware update mechanism allows the firmware consumer (client) to take decisions about downloading and installing the firmware update. Therefore, a light-weight client initiated firmware update mechanism is needed to securely update IoT devices.

Domain name system forms the backbone of internet infrastructure and is a light-weight and secure way to retrieve information. The DNS supports numerous resource record types for storing data. One such newly proposed record type is the OX record type. This record type with its various fields can be utilised to provide URI/URL of a firmware update file. This work is implemented on an Ipv6-only network as it is expected to become the de facto standard in future.

1.2 *Objectives of the Research*

1.2.1 Overall objective

The overall objective of the research project is to study and develop a client-initiated PULL based firmware update mechanism for IoT devices by using the OX resource record type of the DNS system over an IPv6 network and carry out performance testing of the firmware update mechanism.

The proposed work consists of the following objectives:

1. Study the DNS system, OX resource record type and DNSpython library.
2. Develop and test different update logic using OX resource record type to facilitate firmware update in IoT Devices.
3. Test update logic for various performance parameters on an IPv6 Network.

1.3 *Scope*

The project activity primarily focuses on using the DNS system for providing firmware updates to resource-constrained IoT devices on an IPv6 network. The update logic and its performance testing are carried out only for a few DNS exceptions and the device count is confined to a few hundred emulated devices. Project activity is limited to providing a proof of concept for the DNS firmware approach and conducting its elementary testing.

2 LITERATURE REVIEW

Firmware updates for IoT or embedded devices have been delivered through many mediums such as via an existing mobile network, as described in [4] Feilong Zhu, 2010 which uses a GPRS module connected on one side to a PC and on the other side to the IoT or embedded device which is to be updated. A similar approach is utilised in [5] Kesab Bhattacharya, 2015 in which instead of GPRS modules, a couple of inexpensive GPRS-enabled phones are used. Paper [6] Taewook Heo, 2015 proposes the use of Bluetooth connectivity in a smartphone to perform firmware updates for embedded devices. But the use of PCs or smartphones is not always practical, especially for devices which are located remotely. Also, these approaches present a scalability problem. [7] Jong-Hyouk Lee, 2016 proposes an ID-based secure delivery mechanism for firmware updates in a home network to which the device is connected to.

The possibility of IoT devices to upgrade running firmware over the Internet, also called FOTA [8] (Firmware Over The Air) is well known and is widely used on very powerful embedded systems devices. But very few systems have been proposed for providing a firmware update mechanism for resource-constrained IoT devices. For example, Jurkovic and Sruk [9] Vlado Sruk, 2014 analyze the use and the specifics of software agents in the process of upgrading the software of the smaller resource-constrained embedded devices without an operating system. The architecture uses a centralized system to distribute adequate firmware updates to a large number of embedded devices.

Amongst the few developed systems for resource-constrained IoT devices, the concentration is mainly on providing a PUSH based server-initiated firmware update mechanism. These techniques majorly employ some kind status trackers for tracking information about the status of each device produced by the manufacturer. Also, in the above manifestations, the server is responsible for the timing of download and installation of firmware updates on each and every device.

3 METHODOLOGY

3.1 *Key Concepts*

3.1.1 IPv6

Internet Protocol version 4 the fourth iteration of the Internet Protocol (IP), is one of the standard internetwork-layer protocol used and the first version of the protocol to be widely deployed on the Internet. IPv4 uses a 32-bit addressing scheme to support 4.3 billion devices, which was anticipated to be sufficient. However, the growth of the internet, personal computers, smartphones and now the Internet of Things devices proves that the world needed more addresses. Its design did not anticipate a number of requirements that turned out to be crucial. Such requirements not only pertained to the proliferation of devices but also the need for additional security, simpler configuration and better prioritization of some services, such as real-time services (often referred to as Quality of Service issues). The deployment of a new architecture becomes a need as well as a necessity.

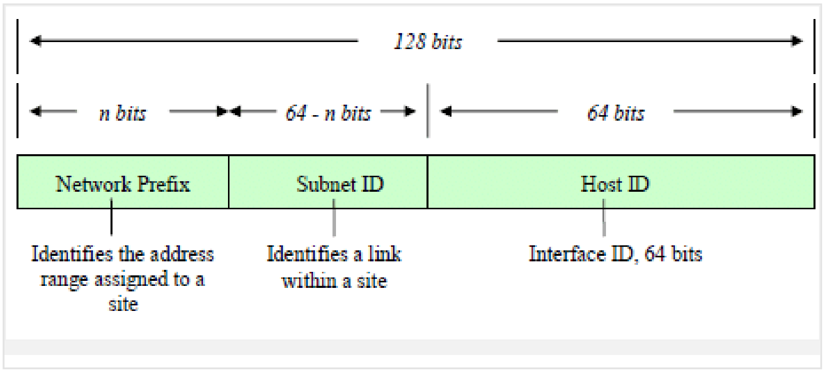


Fig A **IPv6 Address Format**

IPv6 was developed in the mid-1990 by the Internet Engineering Task Force (IETF) [10]. It was primarily engineered to remove the fundamental address space limitation of IPv4. IPv6 uses 128 bits for IP addresses versus 32 bits in IPv4, thus providing a practically unlimited address space that enables any device to have a unique IP address.No doubt, IPv6 improves routing efficiency through better address aggregation resulting in smaller Internet routing tables. It also provides better end-to-end security, improved QoS support, and increased mobility.

Currently, Google [11] estimates the world has 20% to 22% of IPv6 adoption, with India leading in terms of IPv6 adoption at nearly 70% connections [12]. Hence, it can be concluded that IPv6 is going to be the de facto Internet protocol standard for the future. This project activity is thus conducted on an IPv6 only network and assigns a global Unicast address for each emulated IoT device.

3.1.2 Domain Name System

The Domain Name System (DNS) [13] is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols. By providing a worldwide, distributed directory service, the Domain Name System has been an essential component of the functionality of the Internet since 1985.

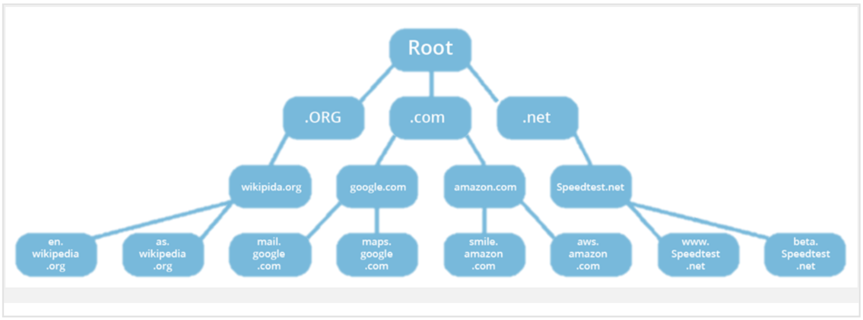


Fig B **Portion of the hierarchy of DNS servers**

DNS Records and Messages

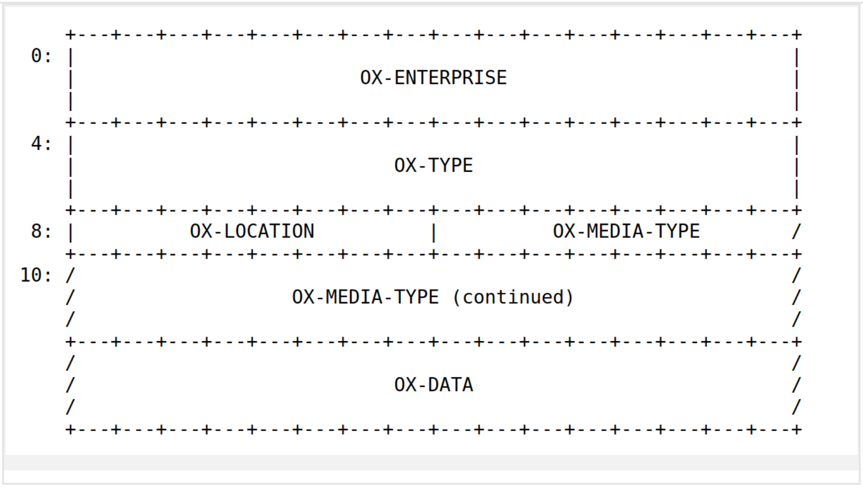
The DNS servers that together implement the DNS distributed database store resource records (RRs), including RRs that provide hostname-to-IP address mappings. Each DNS reply message carries one or more resource records.A resource record is a four-tuple that contains the following fields:

| Name | Value | Type | TTL |

DNS supports many record types such as the Type A record provides the standard hostname-to-IP address mapping. As an example, (relay1.bar.foo.com, 145.37.93.126, A) is a Type A record. Another record type is AAAA which is used to store the hostname-to-IPv6 address mapping. Various other record types (NS, MX, CNAME..etc) are available in the domain name system and each serves different use cases.

Object Exchange Resource Record Type

The Object Exchange(OX) record type as described in Internet draft [14] is a new resource record type designed to implement an architecture for the exchange of digital objects using identifiers stored within the DNS. Each OX RR contains an object type that might be opaque and private to the producer and the consumer of the data and either the data (if small enough to fit in the RR) or a pointer on how to retrieve the actual data.



```
     +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  0: |                                                               |
     |                         OX-ENTERPRISE                         |
     |                                                               |
     +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  4: |                                                               |
     |                            OX-TYPE                            |
     |                                                               |
     +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  8: |           OX-LOCATION         |          OX-MEDIA-TYPE        /
     +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 10: /                                                               /
     /                    OX-MEDIA-TYPE (continued)                  /
     /                                                               /
     +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
     /                                                               /
     /                            OX-DATA                            /
     /                                                               /
     +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

OX RDATA Wire Format

- OX-ENTERPRISE: a 32-bit unsigned integer in network order.
- OX-TYPE: a 32-bit unsigned integer in network order.
- OX-LOCATION: an 8-bit unsigned integer.
- OX-MEDIA-TYPE: A character-string
- OX-DATA: A variable length blob of binary data.

The ENTERPRISE and TYPE fields are combined to indicate the semantic type of the OX record being represented by the RR. That semantic is private to the producer of data hosted on an authoritative DNS server. The LOCATION field signals how the DATA field should be interpreted such as value 1, 2, 3 represent Local, URI, HDL respectively. The MEDIA-TYPE field contains the Internet media type [15] for the object represented by this record. The DATA field contains either the object's data or some form of reference specifying from where the data can be obtained, per the LOCATION field.

The project activity utilizes OX RR type to store the URI/URL and the hash of the new firmware file. This URI/URL and hash is retrieved by the IoT device.

3.1.3 Hash function

A hash function is any function that can be used to map data of arbitrary size onto data of a fixed size. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. This work uses SHA-2(224 bits) [16]for obtaining hash values of the firmware file.SHA-224 is utilised as the domain name cannot contain more than 63 characters. The hash function is used to calculate the hash value of firmware files. The client queries with the hash value of the current firmware file and receives in response the URL/URI and hash value of the new firmware file.

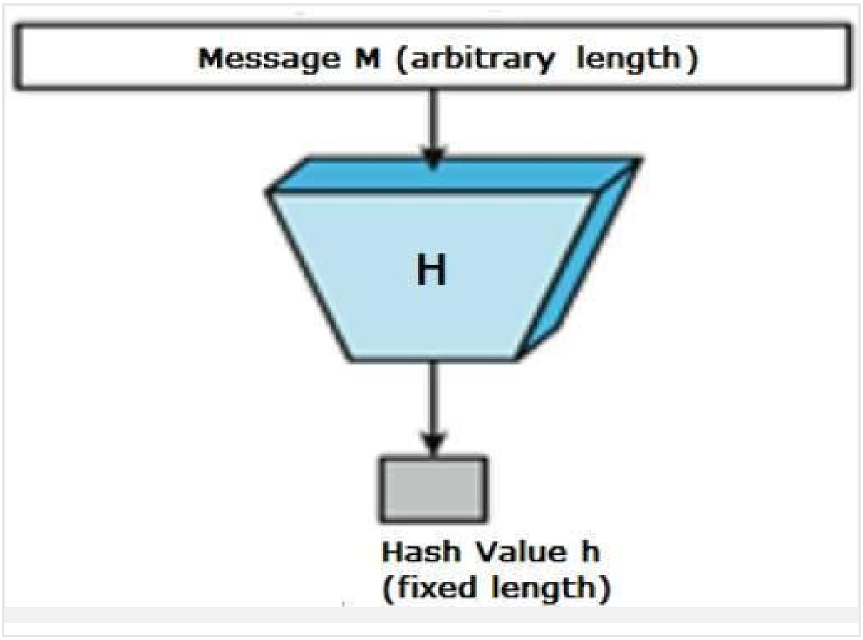


Fig C **Hash Function**

The use of hash value instead of the version number in DNS query and response has the following advantages:

1. The use of hash values in the DNS response field allows the client IoT device to verify the downloaded firmware file from the URL.

2. While querying the client first generates the hash value of its currently running firmware, if the current firmware is corrupted the client will generate a wrong hash and can always fall back to a default version as stated below.
3. Version numbers are predictable, whereas the hash values are totally random and practically unique which adds to the security.

3.2 *Firmware Update Logic*

In this proposed Firmware update logic, the client IoT device queries the authoritative DNS name server for the new firmware update. The query contains the hash value of the currently installed firmware file. The hash value is generated through the aforementioned SHA-2 Algorithm. The authoritative name server replies with a DNS response, which contains the URI/URL and the hash value of the new firmware update file. The client then fetches the firmware file from firmware server of the IoT OEM. The downloaded file can then be verified against the hash value received from the DNS response. After the downloaded firmware file is verified the client IoT device can schedule the installation of the new firmware update. After installing the latest version, when the device queries again for the new update it will receive the same hash value as it queried with, which will indicate the device that it has installed the latest firmware.

If the client is running a partially corrupted firmware, the client shall generate and query with an erroneous hash value, for which the DNS server shall respond with default safe version and its URL so the client can revert back to a safe firmware version.

3.2.1 Example case

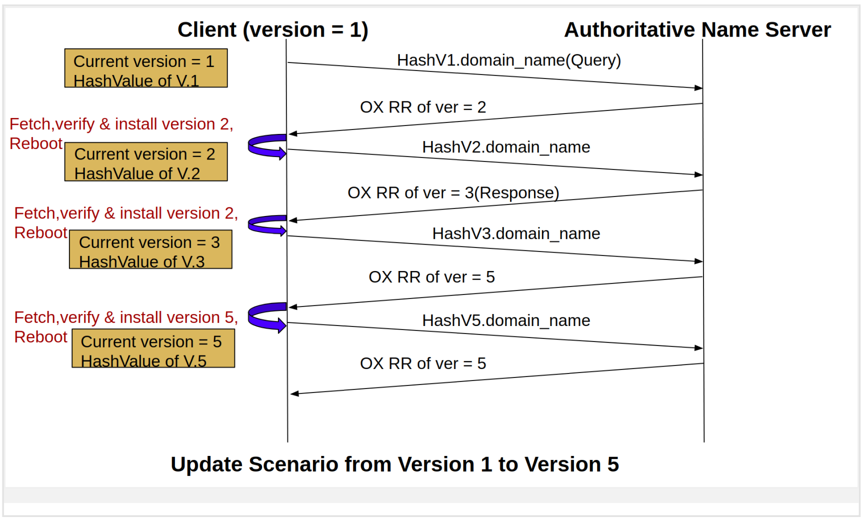


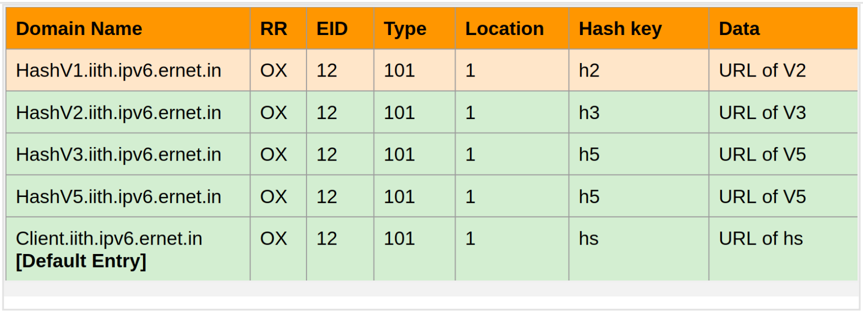
Fig D **Update Logic**

In this example case above, the client IoT device is currently running version=1 and the latest available firmware update is that of version=5. The firmware inter-dependency is as follows:

Ver.1 -> Ver.2 -> Ver.3 -> Ver.5,

Ver.4 -> Ver.5

i.e the device can go from the first version to second, then third and can directly update to version five. The update to version four is optional and can be ignored. If any client IoT device is currently running version four, it can update to version five directly. The client calculates the firmware hash value of version 1 and queries with this hash value. It receives the response, which contains the URL and the hash value of the firmware file of version 2. It fetches, verifies and installs version 2. The same process repeats until the latest version 5 is installed. After installing version 5, the client queries for new firmware and receives the same hash value as the version installed. This indicates the device is running the latest version.



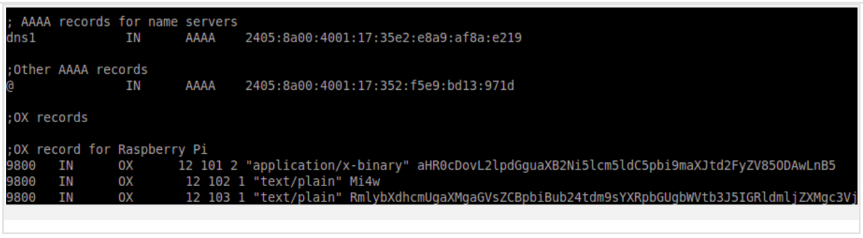
Domain Name	RR	EID	Type	Location	Hash key	Data
HashV1.iith.ipv6.ernet.in	OX	12	101	1	h2	URL of V2
HashV2.iith.ipv6.ernet.in	OX	12	101	1	h3	URL of V3
HashV3.iith.ipv6.ernet.in	OX	12	101	1	h5	URL of V5
HashV5.iith.ipv6.ernet.in	OX	12	101	1	h5	URL of V5
Client.iith.ipv6.ernet.in **[Default Entry]**	OX	12	101	1	hs	URL of hs

Fig E **OX record at the name server.**

3.3 *Tools*

3.3.1 BIND

BIND (Berkely Internet Name Domain)[17] is a popular open source implementation for DNS. It performs both of the main DNS server roles – acting as an authoritative name server for one or more specific domains, and acting as a recursive resolver for the DNS system generally. For the purpose of this project BIND, 9 is utilised as it has in-built support for IPv6. BIND 9 is installed at one server which acts as an authoritative name server for resolving IPv6 based domain name used in this work. In BIND version-9.12.3 the DOA resource record type is modified to OX resource record. The zone files are maintained in the name server for AAAA and OX records.



```
; AAAA records for name servers
dns1            IN      AAAA    2405:8a00:4001:17:35e2:e8a9:af8a:e219

;Other AAAA records
@               IN      AAAA    2405:8a00:4001:17:352:f5e9:bd13:971d

;OX records

;OX record for Raspberry Pi
9800   IN      OX      12 101 2 "application/x-binary" aHR0cDovL2lpdGguaXB2Ni5lcm5ldC5pbi9maXJtd2FyZV850DAwLnB5
9800   IN      OX       12 102 1 "text/plain" Mi4w
9800   IN      OX       12 103 1 "text/plain" RmlybXdhcmUgaXMgaGVsZCBpbiBub24tdm9sYXRpbGUgbWVtb3J5IGRldmljZXMgc3Vj
```

Fig F **Sample Zone File**

3.3.2 DNSpython

DNSpython[18] is a DNS toolkit for Python. It supports almost all record types. It can be used for queries, zone transfers, and dynamic updates. DNSpython provides both high and low-level access to DNS. The high-level classes perform queries for data of a given name, type, and class, and return an answer set. The low-level classes allow direct manipulation of DNS zones, messages, names, and records. DNSpython library is used to query the name server and retrieve the URL and hash for the client IoT device. As DNSpython library does not have the support for OX-RR type, it has been modified to provide support for the same.

4 **RESULTS AND DISCUSSION**

4.1 *Purpose*

The proposed update logic was tested for different kinds of DNS exceptions using the DNSpython library. The purpose is to check for the average duration taken by the IoT device to update to the latest version, under varying Exception loss percentages. The study has been carried out for two DNS response fragment sizes. The Client IoT device is randomly assigned a version number and its respective hash value using the random() function. The client then starts querying DNS name server to fetch the latest firmware file. After fetching the latest firmware file the client generates the hash of the new file and verifies it against hash value received from the DNS response. If it matches correctly, the client reboots and installs the new firmware.

4.1.1 Exceptions

1. Timeout exception: The timeout exception occurs when the query response is not received within the stipulated timeout value. In this case, the client retries again after some time, the wait time until the next query is calculated randomly. The timeout exception is implemented using the 'tc' command:

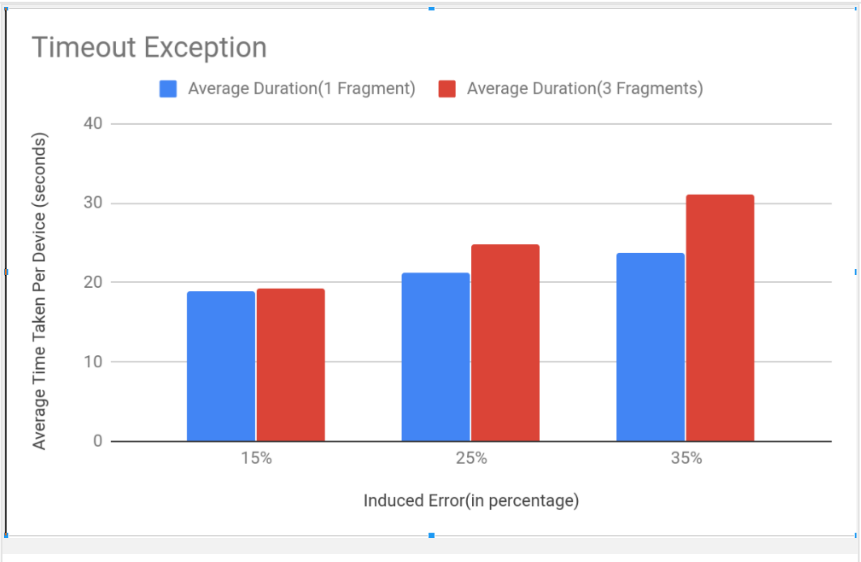
```
sudo tc qdisc replace dev <eth0> root netem loss <In percent>  delay <In ms>
```

2. NXDOMAIN exception: A NXDOMAIN exception suggests that the RR is not present for the given query in the name server. This replicates the situation where the client has installed a corrupted firmware and can fall back to a safe default version. This is implemented by passing a wrong hash value to the name server.
3. NoNameServer Exception: A NoNameServer exception arises when the name server is not reachable. In this case, the client IoT device is expected to wait for some time and try again. The Client device uses an exponential back-off technique to calculate the wait time. This exception is implemented by querying for the incorrect name server.

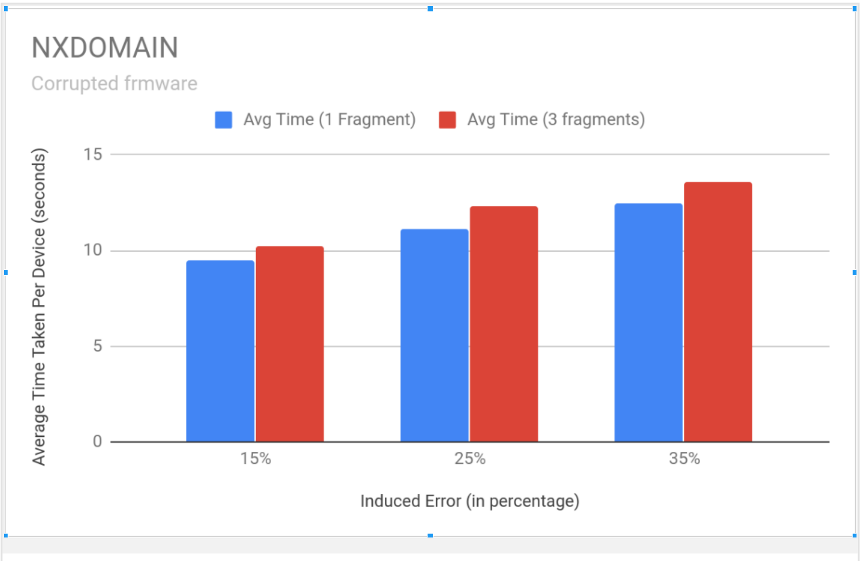
The following parameters are set for the tests:

- No. of emulated devices : 500
- DNS Exceptions: Timeout, NXDOMAIN , NoNameServer
- Varying Exception Loss percentage: 15%, 25% ,35%
- No. of Response Fragments: 1 , 3
- No. of firmware versions: 5 , Inter-dependency: 1 -> 2 -> 3 -> 5, 4 -> 5
- Timeout value: 5s
- Inter-query time:1s
- RTT : 0.236 ms

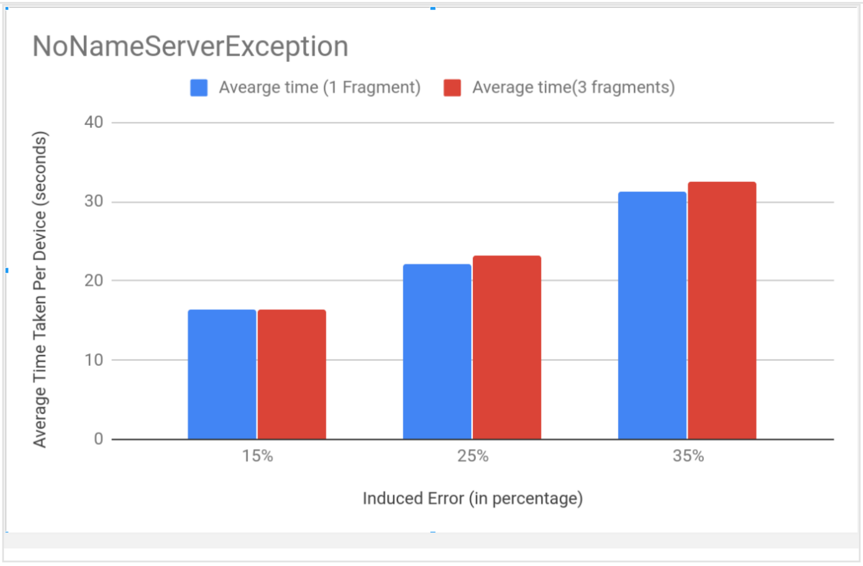
4.2 *Results Obtained*



Graph 1 **Timeout Exception**



Graph 2 **NXDOMAIN Exception**



Graph 3 **NoNameServer Exception**

The above graphs are plotted for the average time taken by the device to update to the latest version vs Induced Error percentages for various exceptions. As inferred from the results above there is an increase in the average time taken by the device to update as the exception loss percentage increases. The average time taken to update to the latest firmware version is lowest for NXDOMAIN exception, as there is no waiting time involved. The average duration is relatively higher for NoNameServer Exception, this is due to the fact that the client has to exponentially back-off before querying again to the name server.

There is also a sharp increase in the average time taken by the client IoT device when the DNS response fragment size is increased from 1 fragment to 3 fragments. This is expected as the communication time & cost for 3 DNS fragments will be significantly higher than that for a single fragment. This is more pronounced for the Timeout Exception.

5 CONCLUSION AND RECOMMENDATIONS

Providing timely firmware updates to IoT devices, especially to the resource-constrained ones is critical to the overall success of IoT. This work attempted to provide a light-weight PULL based solution for IoT firmware updates using the OX-RR type of the DNS. The project activity realized a primitive update logic using the OX-RR type on an IPv6-only network. Elementary testing was also carried out for single and multiple DNS fragment sizes for different DNS exception cases.

Future work can be carried out by scaling the number of devices and by conducting exhaustive testing, considering all different DNS exception cases. Various other DNS security options could be integrated with this work like DNSSEC, DNS over TLS, DNS cookies etc,.

ACKNOWLEDGEMENTS

With great pleasure, I express my sincere gratitude to my project guide **Dr Bheemarjuna Reddy Tamma**, Associate Professor, IIT-Hyderabad for his excellent scientific guidance and constant encouragement. I would also like to thank **Dr Antony Franklin**, Associate Professor, IIT-Hyderabad for his critical supervision and support.

I express my heartfelt thanks to my mentors, **Ms Madhura A**, and **Ms Tulika Agarwal** for their indispensable support, motivation and guidance. I would like to extend my thanks to all the members of the NeWS Lab, IIT-Hyderabad for their constant encouragement.

I would also like to thank **Prof. Rushikesh A Deshmukh**, RCOEM,Nagpur for providing me with the Letter of Recommendation.

REFERENCES

[1]https://www.forbes.com/sites/louiscolumbus/2018/08/16/iot-market-predicted-to-double-by-2021-reaching-520b/#62020d4b1f94

[2]https://www.bosch-iot-suite.com/service/rollouts/

[3]https://www.ibm.com/developerworks/community/blogs/Informix%20Bit%20&%20Pieces/entry/Device_Management_using_IBM_Watson_IoT_Platform?lang=en

[4] YANG, M. and F. ZHU. The Design of Remote Update System Based on GPRS Technology. In: 2010 International Conference on Management and Service Science. Wuhan: IEEE, 2010, pp. 1–4. ISBN 978-1-4244-5325-2. DOI: 10.1109/ICMSS.2010.5575699.

[5]DALAI, S., B. CHATTERJEE, D. DEY, S. CHAKRAVORTI and K. BHATTACHARYA. Microcontroller based remote updating system using voice channel of cellular network.

In: 2015 IEEE Power, Communication and Information Technology Conference (PC-ITC). Bhubaneswar: IEEE, 2015, pp. 11–16. ISBN 978-1-4799-7455-9. DOI: 10.1109/ PCITC.2015.7438154.

[6]HONG, S. G., N. S. KIM and T. HEO. A smartphone connected software updating framework for IoT devices. In: 2015 International Symposium on Consumer Electronics (ISCE). Madrid: IEEE, 2015, pp. 1–2. ISBN 978-1-4673-7365-4. DOI: 10.1109/ISCE.2015.7177805.

[7]CHOI, B. C., S. H. LEE, J. C. NA and J. H. LEE Secure firmware validation and update for consumer devices in home networking. IEEE Transactions on Consumer Electronics. 2016, vol. 62, iss. 1, pp. 39–44. ISSN 0098- 3063. DOI: 10.1109/TCE.2016.7448561.

[8] http://en.wikipedia.org/wiki/Over-theair_programming/

[9]JURKOVIC, G. and V. SRUK. Remote firmware update for constrained embedded systems. In: 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Opatija: IEEE, 2014, pp. 1019–1023. ISBN 978-953-233-077-9. DOI: 10.1109/MIPRO.2014.6859718.

[10]https://tools.ietf.org/html/rfc1883

[11]https://www.google.com/intl/en/ipv6/statistics.html

[12]https://www.akamai.com/uk/en/resources/our-thinking/state-of-the-internet-report/state-of-th

e-internet-ipv6-adoption-visualization.jsp

[13]https://www.ietf.org/rfc/rfc1035.txt

[14] https://tools.ietf.org/html/draft-durand-object-exchange-00

[15]https://tools.ietf.org/html/rfc6838

[16]https://en.wikipedia.org/wiki/SHA-2

[17] https://www.isc.org/bind

[18] http://www.dnspython.org/docs/1.16.0/dns-module.html