INTERNET DRAFT
draft-shyam-hipi-00.txt
Intended status: Experimental
Expires: July 26, 2020

S. Bandyopadhyay
January 26, 2020

Host Identification with Provider Independent Address
draft-shyam-hipi-00.txt

Abstract

This is a protocol to identify a host with a provider independent address. It is useful to identify a host uniquely in a multihomed environment where each host gets associated with more than one provider assigned addresses. By means of associating a host with a provider independent address, customers/customer networks will be able to retain their number even after changing their service provider(s).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 26, 2020.

1. Introduction

Provider independent (PI) addressing can be conceived as naming a host with a number. It can be used by customer networks who would like to retain their number even after changing their service provider; also it is useful to designate a host uniquely if the customer network is multihomed. Just like in name services, as an address needs to be resolved corresponding to a name to initiate communication, the same is required for PI addressing. Each globally unique PI address will be associated to at least one global unicast provider assigned (PA) address. For a host with single interface, this number will be same as the number of service providers the customer network is associated with. This protocol resolves PA addresses associated with a PI address with the approach of DNS. Out of the entire internet protocol addresses, it expects same size of address space to be allocated to PI addresses as the address space allocated for provider assigned unicast addresses. As specified in section 3.2.1 of the architectural specification[1], it assumes address space with prefix ``00" will be assigned for PA addresses and address space with prefix ``01" will be assigned for PI addresses.

2. PI address resolution

This section tries to come up with a solution for PI address resolution with the approach of DNS[2] with necessary differences. Just like name space in DNS, entire address range with prefix 01 will be the address space used by PI addresses. Servers that will hold the information of mapping between PI addresses and corresponding PA addresses will be called as PIMapServers and the programs that will be used to resolve addresses will be called as PIMapResolvers.

In case of DNS where name is used in hierarchical format to resolve the addresses, PI address resolution will be based on the prefix of the PI address used for resolution. The prefix is determined based on the architectural model used for the internet. Based on the prefix information addresses of a list of servers can be found out that will act as regional servers which will be used to resolve mapped PA addresses corresponding to that PI address. A prefix will serve a fixed address space within entire PI address space. Address space belonging to a prefix will be distributed within customer networks of heterogeneous sizes. Address space allocation and the mapping of associated PA address(es) will be assigned by a regional authority. The regional authority will be fully responsible for the operation of regional servers in that region.

Like DNS, there are some root servers which will have some fixed addresses, under which there are some prefixes which will act as top-level-domains. In case of CIDR based hierarchy, these prefixes may be

of different prefix lengths which are selected based on the requirements. Each prefix in a top level domain can further be split into number of prefixes with the approach of CIDR. This tree structured hierarchy will be kept on growing till we get prefixes associated with regional servers. Each prefix associated with a regional server will be distributed amongst customer networks of various sizes as well as prefixes that will again be associated with some regional servers with the approach of CIDR. These regional servers can be considered as equivalent to the authoritative name servers of DNS which are associated with zones. As stated earlier, prefixes starting with "00" will be assigned for provider assigned addresses and prefix starting with "01" will be assigned for provider independent addresses where as prefix starting with "1" will be assigned for addresses of all other types.

As inherent hierarchy is involved in "Mesh structured hierarchy", this hierarchy goes up to two levels. As usual, there will be some root servers with fixed assigned addresses. Each root server will have prefixes with "01.A" that will act like top level domain. Under each top level domain, there will be entries with prefixes "01.A.B". Within a region "A.B", every global PA address is represented as "00.A.B.C.user-id". In order to support customer networks of heterogeneous sizes with the approach of VLSM, the "user-id" portion is further divided as "subnet-id.user-id". So, the effective network prefix of a customer network in PA address space is "00.A.B.C.pa-subnet-id". Within an "A.B", entire PI address space with prefix "01.A.B" will be distributed within customer networks of heterogeneous sizes. So, effective network prefix of a customer network with PI address will be "01.A.B.pi-subnet-id". A particular prefix "01.A.B.pi-subnet-id" will be mapped to at least one provider assigned prefix of same prefix length. For a multihomed customer network within "A.B" that receives services from two service providers will have prefixes "00.A.B.C1.pa-subnet-id1" and "00.A.B.C2.pa-subnet-id2". A PI address prefix "01.A.B.pi-subnet-id" of same length will be mapped to both these prefixes of PA address space. Every region "A.B" will have regional server and backup server(s) with a maximum limit (say 4) with net addresses "00.A.B.server1", "00.A.B.server2", "00.A.B.server3" and "00.A.B.server4".

Each PIMapServer will have a database of records that will have information to resolve PI addresses. In memory copy of a region will have an array of records where each record will have the following format:

NetAddress	NetMask	Type	TTL	NAddr	Addr(1-4)

First two fields "NetAddress/NetMask" represents the PI address range of a network. "Type" will be either Domain/Referral/Individual/ SingleEntry/Default based on which a query and rest of the fields of a record have to be processed. A PI address can have maximum four mapped PA addresses. "Addr1", "Addr2", "Addr3", "Addr4" will hold the corresponding PA addresses and "NAddr" will hold the number of such addresses. The field "TTL" is a 32 bit integer measured in seconds which will hold same meaning and approach as defined in the specification of DNS[2]. When a server receives a query for an address "X", it extracts the record of the network based on "NetAddress/NetMask" and "X" from its database. If no matching record is found, a negative response is sent. Based on the "Type" of the record, the query is processed in the following manner.

Type=Domain:

This is the most common type. If a customer network would not like to maintain a map server opts for this option. In this case there will be one to one mapping between a PI address and corresponding PA addresses. The fields "Addr1"/"Addr2"/"Addr3"/"Addr4" will hold the PA Net Addresses corresponding to the PI address of the network. Server will send the matching record to the resolver with Type=Domain. Resolver will extract the user-id portion of "X" and find the corresponding mapped PA addresses based on "Addr1"/"Addr2"/...etc.

Theoretically, "A.B" portion of a PI address need not match with the "A.B" portion of the corresponding PA addresses. Consider a large corporate that has its corporate office and a branch office within the same region of a particular "A.B" and some other offices with different values of "A.B". The corporate can maintain a contiguous range of PI addresses for the ease of its operation. It needs to split entire PI address range based on its offices and assign the corresponding PA addresses. In order to minimize the path of a query it is desirable that "A.B" of a PI address and its corresponding mapped PA addresses belong to the same region.

Type=Referral:

This is used when an address within the domain "NetAddress"/"NetMask" has to be processed by another map server. The map server may itself be another regional server or a server within a customer network.

When a customer network would like to have a direct control for the mapping of its addresses it needs to opt for this option. "Addr1"/"Addr2"/"Addr3"/"Addr4" of the database entry will hold the pointer to the information associated to each map server. "NAddr" will hold the number of map servers that can be referred. Information

of each server will hold the following values: PI address of the map server + Number of PA addresses to reach the map server + PA addresses of the map server. Any one of these map servers need to be queried for further processing. A server may act either in recursive mode or in iterative mode based on its implementation just like in DNS. A large corporate may have different offices and each (or some of them) may maintain a map server based on their policies.

When a server needs to handle a particular address separately, it needs to set "NetAddress" with that particular address and all the bits of "NetMask" will be set to "1". The "Type" field has to be set as "SingleEntry"(which is similar to the Type Address(A) in terms of DNS). If some of its addresses need to be handled separately but for the rest common rule may apply (like Type=Domain), records of the individual entries should be processed first and then for the rest. In these cases "Type" has to be set as "Default". So, a server of a customer network may have database entries with Type=Domain/Referral /SingleEntry/Default. It makes sense for a server (or a master file) to have entries with Type=Default, but from the point of a resolver, it does not make any sense. So a server needs to extract the PA addresses and form a record with Type=SingleEntry and send it back to the resolver.

For a host having multiple interfaces, each interface may be assigned PA addresses supplied by all the service providers, but it is desirable that PI address gets mapped to only one of them (preferably for a CE router, the interface which will have the shortest path will be mapped PI address with the PA address associated with that CE router).

Type=Individual:

This is meant for the individual users opting for services like telephonic services that need to maintain PI address. With this option a mobile user may maintain its PI address after changing its service provider. A map server needs to maintain some networks with a range of PI addresses in its database. When a query for an address "X" is received, server needs to get the corresponding record where "Addr1" will hold the pointer to a open file descriptor (or pointer to the in memory copy) of a separate data file where there will be one to one mapping between PI address and its corresponding PA address of all the assigned PI addresses. These networks and assignment of individual PI addresses have to be done by the regional authority.

As with Type=Default, Type=Individual does not make any sense to a resolver. So, server needs to extract PA address and form a record with Type=SingleEntry and send it back to the resolver.

As stated above, this solution is based on the approach of DNS. For the ease of implementation and to make use of the existing source code related to DNS (e.g. BIND) most of the features have been taken from DNS. DNS supports multiple entry output, but they appear in a sequential manner. In order to make processing easier, they are arranged in a structured manner in this document.

IANA has assigned a port <IANA_TBD1> for its UDP/TCP based implementation.

2.1. Record Format

Each record (the way they will appear in a master file or will be used for communication) will have the following format:

NetAddress/NetMask + Type (8 bit unsigned int) + <TTL> + RDATA (Type specific information)

Record types are primarily the types of records as described above along with three other types: SOA (Start of a zone of authority), MPS (host with Type=SingleEntry that acts as a Map server for this zone) and DFL (Data File). These types are mainly useful in the context of processing AXFR/IXFR/NOTIFY/DFAXFR/DFIXFR messages.

Types are defined as follows:

Types	values	comments
SEN (SingleEntry)	1	same as type A(address) in DNS
MPS (MapServer)	2	Map server
DMN (Domain)	3	
DEF (Default)	4	
REF (Referral)	5	
SOA (Start of a zone)	6	
IND (Individual)	7	
DFL (Data File)	8	

RDATA of different types will appear as follows:

Type=SOA:
PI address of server+SERIAL+REFRESH+RETRY+EXPIRE+MINIMUM (meaning and values of SERIAL/REFRESH/RETRY/EXPIRE/MINIMUM are same as they were defined in section 3.3.13 of RFC 1035[3])

Type=(SEN/MPS):
NAddr(Number of addresses) + corresponding PA addresses

Type=(DMN/DEF):
NAddr(Number of addresses) + corresponding Net addresses

Type=REF:
NAddr(Number of map server) + for each map server (PI address of map server + NAddr(Number of addresses of map server) + corresponding PA addresses))

Type=IND:
NAddr(=1) + full path name of the data file

Type=DFL:
Data file name + SERIAL + Number of records in the data file(32 bit unsigned int)

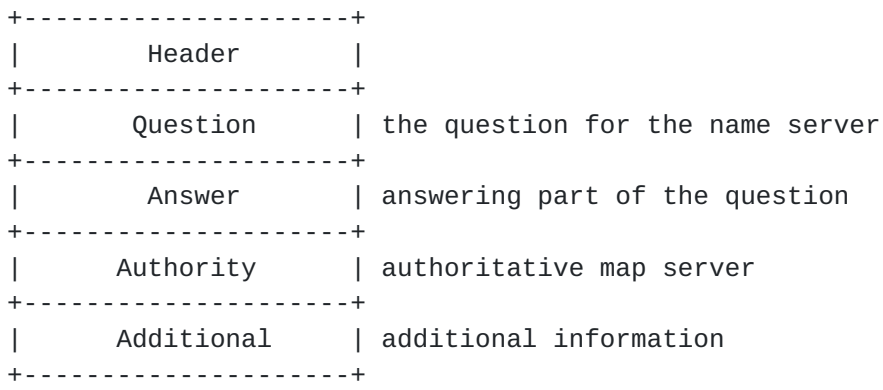
While used in communication data file name is used as its length (8 bit unsigned int) followed by the octets of the string.

TTL value of a record has to be set to 0 if it is not relevant or to accept the value associated with the record of SOA.

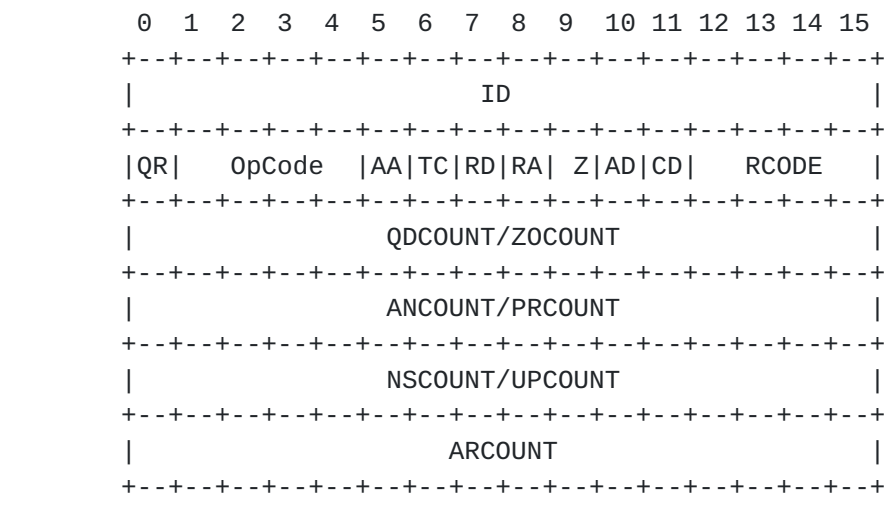
2.2. Messages

In order to support most of the features of DNS, message format has been retained almost same as that of DNS. So, all the relevant fields will be processed exactly in the same manner as that have been done in DNS and all the irrelevant issues have to be ignored. Rest of this section describes where and how changes have to be made.

As defined in RFC 1035, the top level format of message is divided into 5 sections (some of which are empty in certain cases) shown below:



The header section has been retained as defined in RFC 5395[4] as follows:



The question section will have two parts:
QType(one octet unsigned int)+QData.

Query types are defined as follows:

QTypes	values	comments
SEN	1	query for mapped PA address
SOA	6	query information related to SOA
DFL	8	query information related to data file
DFXFR	249	data file transfer
DFIXFR	250	incremental data file transfer
IXFR	251	incremental authoritative data file xfr
AXFR	252	authoritative data file transfer

QData will hold values based on QType.

Following section describes issues related to QType=SEN. Issues related to all other QTypes (i.e. related to file transfer) will be discussed afterwords.

For QType=SEN(1): QData=PI address that needs to be resolved.

The answer section, authority section and additional section will have a number of resource records where the number will be specified in the header.

On receiving a query, map server will return the matching record from its database. If response is address, the answer section will hold the record of any one of these two types: SEN/DMN.

If Type=DMN, resolver needs to extract the mapped addresses as

described in section 2.

If Type=DMN, entire address range will appear in the form of NetAddress/NetMask. This will have advantages while catching data for any particular address, but getting the information of the entire address range.

If the response is referral, answer section will be empty and the authoritative section will hold the record with Type=REF.

If server supports recursion, for each iterative process that it receives a record with Type=REF, it needs to push the record to the additional section of the message that needs to be sent to the resolver. So, additional section will hold the records of Type=REF of the chain of the tree through which PA addresses have been resolved.

2.3. Master file and data file

Section 5 of RFC 1035 states:

"Master files are text files that contain RRs in text form. Since the contents of a zone can be expressed in the form of a list of RRs a master file is most often used to define a zone, though it can be used to list a cache's contents."

Section 5.1 of RFC 1035 states:

"The format of these files is a sequence of entries. Entries are predominantly line-oriented, though parentheses can be used to continue a list of items across a line boundary, and text literals can contain CRLF within the text. Any combination of tabs and spaces act as a delimiter between the separate items that make up an entry. The end of any line in the master file can end with a comment. The comment starts with a ";" (semicolon)."

Master files follow the same approach and format in the line of DNS as described in section 5 of RFC 1035 with necessary differences.

An example master file may look like as follows:

```
@ "PI NetAddr"/"Net Mask"  SOA  "PI address of primary server" (
                                 20     ; SERIAL
                                 7200   ; REFRESH
                                 600    ; RETRY
                                 3600000; EXPIRE
                                 60)    ; MINIMUM
"PI NetAddr"/"Net Mask"    MPS  0  NAddr "PA addresses"
"PI NetAddr"/"Net Mask"    SEN  0  NAddr "PA addresses"
```

```
"PI NetAddr"/"Net Mask"    DMN  0  NAddr "Net addresses"
"PI NetAddr"/"Net Mask"    DEF  0  NAddr "Net addresses"
"PI NetAddr"/"Net Mask"    IND  0  NAddr(=1) "Data file name"
```

A data file contains a sequence of entries where each entry appears in a separate line. Each entry is a mapping between a PI address and its associated PA address separated by space(s). Entries are generally sorted with PI address. As in case of master file comments can be inserted with the start of a ";" (semicolon) that will end at the end of the line. Data files are commonly associated with the map servers maintained by regional authority, but they are not generally associated with the map servers maintained by individual customer networks. A data file entry may appear to be as follows:

"PI Address" NAddr "PA Addresses"

A map server may have a number of data files. These files have to be defined in another file (a supporting file, the way boot file "named.boot" is used in BIND) that will have information of each of them. An entry in that file will follow the same format of a record (Type=DFL) and will have the following fields:

"PI NetAddr"/"NetMask" Type(DFL) TTL "Data File Name" SERIAL "Number of records".

This file will be used to process message with QType=DFL which will be used to support data file transfer/incremental data file transfer.

For QType=DFL(8): QData="PI NetAddr"/"NetMask" of the desired network
For QType=SOA(6): QData="PI NetAddr"/"NetMask" of the desired zone

A map server will return a record of Type=DFL on receiving a query with QType=DFL where as it will return a record of Type=SOA on receiving a query with QType=SOA.

2.4. Zone maintenance and transfers

Section 4.3.5 of RFC 1034 states:

"The general model of automatic zone transfer or refreshing is that one of the name servers is the master or primary for the zone. Changes are coordinated at the primary, typically by editing a master file for the zone. After editing, the administrator signals the master server to load the new zone. The other non-master or secondary servers for the zone periodically check for changes (at a selectable interval) and obtain new zone copies when changes have been made.

To detect changes, secondaries just check the SERIAL field of the SOA for the zone. In addition to whatever other changes are made, the SERIAL field in the SOA of the zone is always advanced whenever any change is made to the zone."

Section 1.2 of RFC 5936 states:

"A DNS implementation is not required to support AXFR, IXFR, and NOTIFY, but it should have some means for maintaining name server coherency. A general-purpose DNS implementation will likely support AXFR (and in the same vein IXFR and NOTIFY), but turnkey DNS implementations may exist without AXFR."

Zone maintenance and transfer will follow the same approach as DNS with few minor updates. Frequency of update of data files will be high compared to the frequency of update of master file. That is why transfer(/incremental transfer) of data file has been treated separately from the transfer(/incremental transfer) of master file.

For all the messages of QType=AXFR/DFXFR/IXFR/DFIXFR, QData="PI NetAddr"/"NetMask" of the desired zone or the desired network. NOTIFY message needs to include which file has been updated followed by the related information. So, if master file has been changed, NOTIFY message with query type SOA will be sent and query type DFL will be sent if a data file has been changed.

Transfer of master file will be same as transfer of master file in DNS followed by transfer of all the data files. i.e. processing of AXFR will have the same approach as DNS followed by DFXFR for all the data files. In order to make this happen, at the end of transferring the contents of the master file, server (of AXFR message) needs to send NOTIFY message for all of the data files belonging to that zone to the client(i.e. the secondary server). Processing of NOTIFY of a data file by the secondary server needs to send DFIXFR to the primary if data file already exist; otherwise it needs to send DFXFR. Incremental update of master file (IXFR) will be same as IXFR in DNS with a minor update. If client of IXFR finds a new data file gets introduced, it calls DFXFR corresponding to that data file. Similarly if an entry of a data file gets deleted, client deletes corresponding data file.

Processing of DFXFR will have same approach of AXFR in DNS. Similarly processing of DFIXFR will have same approach as IXFR in DNS. While transferring a data file record, an equivalent record of type SEN needs to be sent with the values of PI address and mapped PA address(es) from the record of data file. Where ever a record of type SOA is sent while processing AXFR/IXFR in case of DNS, record of type DFL needs to be sent while processing DFXFR/DFIXFR.

For AXFR, IXFR and NOTIFY in DNS, one needs to follow RFC 5936[5], RFC 1995[6] and RFC 1996[7] respectively.

3. IANA Consideration

IANA has assigned a port number <IANA_TBD1> and service name <IANA_TBD2> for PI address resolution for both TCP and UDP.

4. Security Consideration

This document does not include any security related issues.

5. Normative References

[1] S. Bandyopadhyay, "An Architectural Framework of the Internet for the Real IP World" <draft-shyam-real-ip-framework-61.txt> (work in progress).

[2] P.V. Mockapetris., "Domain names - concepts and facilities", RFC 1034, November 1987.

[3] P.V. Mockapetris, "Domain names - implementation and specification", RFC 1035, November 1987.

[4] D. Eastlake 3rd, "Domain Name System (DNS) IANA Considerations", RFC 5395, November 2008.

[5] E. Lewis, A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, June 2010.

[6] M. Ohta, "Incremental Zone Transfer in DNS", RFC 1995, August 1996.

[7] P. Vixie, "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.

6. Author's Address

Shyamaprasad Bandyopadhyay
HL No 205/157/7, Kharagpur 721305, India
Phone: +91 3222 225137
e-mail: shyamb66@gmail.com