INTERNET DRAFT
draft-shyam-site-multi-07.txt
Intended status: Experimental
Expires: March 22, 2015

S. Bandyopadhyay
September 22, 2014

Solution for Site Multihoming in a real IP environment
draft-shyam-site-multi-07.txt

Abstract

This document provides a solution for Site Multihoming of stub networks in a real IP environment. Each user interface in a customer network will have as many global unicast addresses as many service providers it will be connected with. Users can establish multiple connections through different service providers simultaneously. A customer network can maintain private address space to communicate within its users and can share its load while maintaining VPN services. Customer networks can provide IP mobility services as well.

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 March 22, 2015.

1. Introduction

Based on the definition of "multihoming" as stated in RFC3582,

"A "multihomed" site is one with more than one transit provider. "Site-multihoming" is the practice of arranging a site to be multihomed."

This is a general solution for site multihoming of stub networks in a real IP world irrespective of the framework supported by the service provider network. The solution is applicable to any customer network that receives globally unique IP addresses for all of its nodes and communicates with the rest of the world without the help of NAT[8]. It is applicable to any version of IP, i.e. IPv4, IPv6 or any new generation of IP that may emerge by removing the drawbacks associated with IPv6[7]. Within a provider assigned address space, each customer network will posses as many global unicast address space as many service providers it gets connected with. So, an user interface of a host will have as many global unicast addresses as many service providers it will be connected with. Users will have an option of selecting the service provider while initiating a connection with the outside world. Users can maintain multiple connections through multiple service providers simultaneously. A customer network can maintain private IP addresses to communicate within its users and can share its load while maintaining VPN services. Customer networks can provide IP mobility support as well.

There are many variants of UNIX systems (as well as real time operating systems) which make use of BSD source code for their implementation of TCP/IP stack. The solution given below highlights the changes required with the BSD (FreeBSD Release 8.0) source code with the notations used by IPv4. All other implementations of TCP/IP have to be updated in the similar manner.

In this document the term "default router" will refer to the customer edge (CE) router that communicates with the provider network. Also the term "intermediate routers" will refer to all the routers apart from the CE routers.

2. Solution for site multihoming

RFC1122[2] made an extensive study related to different aspects of multihoming. Some of the requirements suggested in that document related to UDP and the application layer were avoided for multihomed hosts in a connected network with a single gateway to reach the outside world. This was achieved by the implementation of TCP/IP by making sure that the interface address of an outgoing packet gets selected based on the route to be followed by the destination

address. This criterion holds good in a connected environment with a single gateway to reach the outside world. Once more than one gateway comes into play to reach the outside world, either routing table of the entire world has to be brought in or needs some enhancements within the existing system to make the things work.

Whenever a customer network gets service from more than one service provider, the customer network can be viewed as having multiple source-id (user-id) space. Each of these IP domain gets connected to different service providers through different routers. So each interface of customer network will have IP addresses as many service providers as it is connected with. Number of routing entries in the routing table will (roughly) become a multiple of IP domains that it supports. Communication between any two hosts within the customer network will follow the traditional routing mechanism. In order to provide multihoming services it is needed that a host computer always forwards packets to the customer edge router associated to the same IP domain while communicating to someone in the outside world. i.e. if the interface of a host computer H receives an IP address 'addr1' and 'addr2' from two service providers P1 and P2 which are connected through routers R1 and R2 respectively, host H has to forward a packet to R1 while using its IP address as 'addr1' in order to send packets to the outside world. So, host computers as well as the intermediate routers have to use default routing based on the source domain of the source address in the IP header.

In order to achieve this, host computers as well as intermediate routers need to have information related to its IP domain (net address/net mask) and the associated default router for all of its IP domains. They need to have a route entry per IP domain for all of its default routers. These information should be uploaded at the system start up time. As each interface is going to have multiple IP addresses, hosts need to have a provision to select its default IP domain (or default router) while initiating communications with the outside world. Users can select this option based on their need (i.e. whether a link is up/down/busy) dynamically. Users can execute multiple applications through different routers simultaneously as well. If no source address is specified by an application, source address has to be selected based on the outgoing interface and the default router as selected by the user.

UDP based servers that need to support multiple clients simultaneously need to respond to a client's request with the same source address that the client had specified as the destination address. In order to satisfy this, system needs to introduce two system calls along with the existing system calls (i.e. read, write, send, sendto, recv, recvfrom)

```
int recvwithdstaddr (int sockfd, char *buf, int nbytes,
    int flags, struct sockaddr *from, int fromlen,
    struct sockaddr *fromcladdr, int fromcladdrlen,
    struct sockaddr *dst, int dstlen);
```

'recvwithdstaddr' receives data with destination address as specified by the sender. It is similar to 'recvfrom' with the additional fields related to the address of the receiving interface of the host. If the sender happen to be mobile or having PI address, it will have a field related to the co located care of address; 'fromcladdr' will hold this value.

```
int sendwithsrcaddr (int sockfd, char *buf, int nbytes,
    int flags, struct sockaddr *to, int tolen, struct sockaddr
    *dstcladdr, int dstcladdrlen, struct sockaddr *src, int srclen);
```

'sendwithsrcaddr sends data specifying the source address of the outgoing interface of the host. It is similar to sendto with additional parameters related to source address. It behaves like sendto if no address is specified for 'src'. The receiver may have may have co located care of address. 'dstcladdr' will hold this value. If application layer calls bind with an address != INADDR_ANY then the address specified by bind prevails over src of 'sendwithsrcaddr'.

All the UDP based servers that need to support multiple clients simultaneously, need to replace 'sendto' with 'sendwithsrcaddr' and 'recvfrom' with 'recvwithdstaddr'.

It has been expressed in several documents including RFC4291[3], that a single interface will posses multiple IP addresses in a real IP environment. In these cases, all the UDP servers have to be updated with the system calls 'sendwithsrcaddr' and 'recvwithdstaddr' even if a customer site gets attached to a single gateway to reach the outside world.

The same logic will apply to server applications with RAW sockets. Server applications that are TCP based should work in the usual manner.

Another system call needs to be introduced to get the source address based on the destination address.

```
struct in_addr getsrcaddr(struct in_addr *dst);
```

Client applications need to use 'getsrcaddr' and 'bind' the source address before communicating with their peer.

Routing of IP packets (in the ip_output module of the hosts and in the ip_forwarding module of the intermediate routers) need to be modified in the following manner.

If destination address of a packet falls outside of its IP domains, it has to be forwarded to the default router based on the domain that the source address belongs to.

If destination address of the IP header falls within any one of its IP domains, usual routing mechanism has to be followed.

If customer network maintains private IP domain, communication using private IP has to be restricted within private IP space.

2.1. Multihoming and IP Mobility

For a mobile node, its co-located care-of IP address[4] has to be bound to one of the IP addresses supported by the service providers (if mobile node advertises more than one address, the home agent will get confused, also there are other implications). Transport layer must ensure that the 'home address' gets tightly coupled with that particular IP address.

A mobile node in a foreign site will have all the IP addresses supported by the foreign site as well as its "Home Address". As the mobile node will also communicate with the outside world with its "Home Address", user should get a provision to choose its "Home Address" while initiating communication. Selection of default router and "Home Address" will be mutually exclusive. One should not interpret it as a selection of one of the global unicast addresses. This is just because a host may have multiple interfaces.

If "Home Address" is selected for communication, the transport layer of the mobile node should use its care of address as the source address and pass its "Home Address" as an option field in the stack. This is because multihoming expects the source address as the deciding factor for packet forwarding.

The IP address of a node with a provider independent address have to be mapped with one of the global unicast addresses. So for the purpose of multihoming whatever will be applicable to a mobile node will also be applicable to a node with provider independent address.

All the issues that need to be handled for IP mobility, provider independent addressing related to multihoming have been thoroughly discussed in section 4 of the architectural specification[7].

2.2. Implementation aspects

Following changes are expected with the source code of BSD.

Introduce ip_domain structure and some parameters as follows:

```
struct ip_domain {
    struct in_addr net_addr;
    struct in_addr net_mask;
    struct in_addr def_router;
};
#define MAX_IP_DOMAINS    16
short num_ipdomains;
struct ip_domain *ipdomain[MAX_IP_DOMAINS];
```

If customer network maintains private IP domain (along with the user-id space provided by the service providers) and expects its communication to be confined within its own space, def_router field has to be set as NULL.

Upload IP domain information for all of its IP domains during system start up. These domain information can be uploaded through router advertisement or through DHCP. The domain information should contain the next hop address to reach the corresponding default router as well.

There has to be a provision to upload these information through sysctl to configure them manually.

Three new sysctl routines have to be introduced under the 'ip' node of the MIB tree (i.e. under CTL_NET, PF_INET, IPPROTO_IP) IPCTL_NUM_DOMAINS, IPCTL_DOMAIN and IPCTL_DEFROUTER. Both IPCTL_NUM_DOMAINS and IPCTL_DEFROUTER are of type CTLTYPE_INT and IPCTL_DOMAIN is of type CTLTYPE_NODE. Using 'sysctl' IPCTL_NUM_DOMAINS has to be configured first. Configuration of IPCTL_NUM_DOMAINS has to populate IPCTL_NUM_DOMAIN entries of nodes under IPCTL_DOMAIN and for each of these nodes three MIB attributes DOMAIN_NET_ADDR, DOMAIN_NET_MASK and DOMAIN_DEF_ROUTER (each of type CTLTYPE_NODE) has to be allocated.

Users should get provision to change IPCTL_DEFROUTER attribute dynamically. As each interface is going to have multiple IP addresses, IPCTL_DEFROUTER has to be assigned a value that will match any one of the entries assigned for DOMAIN_DEF_ROUTER.

Add a route entry for all the default routers during system start up.

System call 'getsrcaddr' has to be processed in the following manner:

```
   If destination address of the IP packet falls outside of its
   IP domains {
       If destination address is from private address space {
           get source address as the private IP address of any of
           its interfaces.
       }

       If user has selected its "Home Address" instead of one
       of the default routers{ /*Applicable to IP mobility/PI address*/
           return its "Home Address";
       }
       else {
           get default router based on the selected
           'default IP domain'

           use 'rtalloc' to get the next hop address for the def router.

           select source address based on the outgoing interface 'ia',
           and the 'default IP domain' as selected by the user.
       }
   }
   else { /* i.e. destination address is inside its IP domains */
       use 'rtalloc' to get the next hop address for the
       destination address.

       If destination address is from private address space {
          select source address based on the outgoing interface
          and the private address assigned to it.
       }
       else {
          select source address based on the outgoing interface
          and the domain that the destination address belongs to.
       }
   }
```

System call 'sendwithsrcaddr' needs to check whether the source address is part of any of the IP domains or not. If it does not belong to any one of these domains, either it is a PI address or a "Home Address" of a mobile node . In these cases it needs to go through the list of interfaces and find out the care of address. IP header needs to be formed with the care of address and the source address as described in section 2.1.

Execute the following steps in the 'ip_output' routine of the IP stack before it calls 'rtalloc' for route look up.

```
   If destination address of the IP packet falls outside of its
   IP domains {
```

```
        get def router address based on the IP domain
        the source address belongs to.

        use 'rtalloc' to get the next hop address for the def router.

        Forward the packet to the next hop.
    }
    else { /* i.e. destination address is inside its IP domains */
       follow the usual procedure to forward packets
    }
```

In BSD, the 'ip_forwarding' routine calls 'ip_output'; so it should be left as it is.

2.3. Multihoming, VPN and load sharing

For a corporate, that maintains multiple offices and communicates within themselves through private address space using VPN, can do load sharing of outgoing traffic of private IP space by segregating private IP domain of each office into number of sub domains through suitable configuration. Let us consider one of its offices gets connected to two providers P1 and P2 and gets address space as 'unicastNetAddr1'/'unicastNetMask1' and 'unicastNetAddr2'/'unicastNetMask2' respectively. It also gets assigned private address space as 'privateDomainNetAddr'/'privateDomainNetMask' from its corporate. For load sharing, it wants to maintain two sub domains with its ID space as 'subDomainNetAddr1'/'subDomainNetMask1' and 'subDomainNetAddr2'/'subDomainNetMask2' respectively. Domain 1 gets associated with the default router CE1 and domain 2 gets associated with CE2. Host computers and intermediate routers will be configured in the following manner:

All hosts of sub domain 1 will have three entries of ip_domain:

```
1: 'net_addr = 'unicastNetAddr1'
   'net_mask = 'unicastNetMask1'
   'def_router = CE1

2: 'net_addr = 'unicastNetAddr2'
   'net_mask = 'unicastNetMask2'
   'def_router = CE2

3: 'net_addr' = 'privateDomainNetAddr'
   'net_mask' = 'privateDomainNetMask'
   'def_router' = CE1
```

All hosts of sub domain 2 will have three entries of ip_domain:

```
1: 'net_addr = 'unicastNetAddr1'
   'net_mask = 'unicastNetMask1'
   'def_router = CE1

2: 'net_addr = 'unicastNetAddr2'
   'net_mask = 'unicastNetMask2'
   'def_router = CE2

3: 'net_addr' = 'privateDomainNetAddr'
   'net_mask' = 'privateDomainNetMask'
   'def_router' = CE2
```

All intermediate routers will have four entries of ip_domain:

```
1: 'net_addr = 'unicastNetAddr1'
   'net_mask = 'unicastNetMask1'
   'def_router = CE1

2: 'net_addr = 'unicastNetAddr2'
   'net_mask = 'unicastNetMask2'
   'def_router = CE2

3: 'net_addr' = 'subDomainNetAddr1'
   'net_mask' = 'subDomainNetMask1'
   'def_router' = CE1

4: 'net_addr' = 'subDomainNetAddr2'
   'net_mask' = 'subDomainNetMask2'
   'def_router' = CE2
```

If any of the CE-PE link fails, that particular CE needs to forward its outgoing traffic to the other CE whose CE-PE link remains active. This can be achieved through tunneling mechanism or by providing a hot link between the CEs. Forwarding of packets should be restricted to packets with private IP space. CE routers need to communicate within themselves at regular intervals and elect a leader within themselves. The elected leader should get privilege to forward private IP broadcast packets to other sites in order to avoid multiplicity. Broadcast packets that are originated only at the local site needs to be forwarded to the other sites. For a remote site, which is connected with PE routers RPE1 and RPE2, PE router of local site can load share its outgoing traffic by segregating its outgoing traffic with a suitable manner. If any of the link between RPE1 or RPE2 fails, it needs to forward all the traffic to the active link as well.

3. Security Consideration

This document provides a solution for site multihoming of stub networks. It does not introduce any security related issue. All the issues related to separation of locator and identifier that were addressed in RFC4218[5] are not applicable here but for common security related issues that any site may experience, one needs to consult with the "Site Security Handbook", RFC2196[6]. For issues related to IP Mobility, section 5 of RFC5944[4] has to be consulted.

4. IANA Consideration

This draft does not request any action from IANA.

5. Normative References

[1] J. Abley, B. Black, V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC3582, August 2003.

[2] R. Braden, "Requirements for Internet Hosts -- Communication Layers", RFC1122, October 1989.

[3] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture.", RFC4291, February 2006.

[4] C. Perkins, "IP Mobility Support for IPv4, Revised", RFC5944, November 2010.

[5] E. Nordmark, T. Li, "E. Nordmark, "Threats Relating to IPv6 Multihoming Solutions", RFC4218, October 2005.

[6] B. Fraser, "Site Security Handbook", RFC2196, September 1997.

[7] S. Bandyopadhyay, "An architectural framework of the internet for the real IP world", draft-shyam-real-ip-framework-12.txt (work in progress), July 2014.

6. Informative References

[8] P. Srisuresh, K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC3022, January 2001.

7. Author's Address

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