Improve Pseudo-anycast based on MobileIP

Ting-Yun Chi, Han-Chieh Chao, Senior Member, IEEE and Tin-Yu Wu
louk@louk.dyndns.org,
{kcc;tyw}@mail.ndhu.edu.tw
+886-3-8632701 (O)

Department of Electrical Engineering
National Dong Hwa University
Hualien, Taiwan, Republic of China

T. G. Tsuei
Department of Electronics
Ta Hwa Institute of Technology
tgt@thit.edu.tw
Hsinchu, Taiwan, ROC

ABSTRACT
In the paper, we will compare some load balance solutions [1] and propose a pseudo anycast solution based on MobileIPv6 (RFC3775)[2]. By adopting the RFC3775, we can make sure each MobileNode can utilize the service even when the node doesn't support the MobileIPv6. We will use the general ICMPv6 message to measure the RTT and conclude rules to decide which server should be connected. Through our proposal, we can make an IP-layer load balance standard with MobileIPv6.

Keywords: MobileIPv6, anycast, load balance

INTRODUCION
IPv6 develops quickly and correlates many operating systems to support IPv6. The development has extended to Mobile IP on its own standard of RFC 3775[3] [4]. On the contrary, the pace on anycast development of IPv6 was relatively slower. Anycast is a revolutionary development on IPv6 to substitute the load balance in IPv4. It takes the task of load balance to the IP layer and provides an unitary standard on load balance. In other words, anycast is a visionary development on IPv6.

Currently, many IPv6-related technologies has adopted the mechanism of anycast in the optimal search on routing including Dynamic Home Agent Address Discovery (DHAAD) and micro handover[5] for example. The protocol of anycast dictates the Host to create a link to connect an address of any-cast and in return an uni-cast site through router. It is quite similar to the mechanism of Mobile that CN starts a link to home address and HA returns a real connecting to CoA. It is thus obvious to take advantage to the available Mobile IP standard in the following three aspects to realize the anycast development.

1. Mobile IP has already been standardized.
2. The amount software of Mobile IP can be used directly on the nodes.
3. Mobile IP has taken the CN support into consideration initially.

In order to select the best link to each node, we are going to add an effective measuring mechanism and connect strategy by suing ICMPv6 under the current structure of RFC3755.

Related Work
There are many main load balance solutions. One of them is client oriented and each client must have special demand software for one special application [6]. The other one is DNS oriented [7]. It is simple and easy to deploy but less sensitive. The most famous one is NAT [8]. NAT is a popular IP layer solution, but it has a bottleneck problem.

There have several solutions on the implementation for anycast. The first one is "Source Identification Option" which was proposed in an Internet-Draft published in 1996 [9]. The second one is "Anycast Address Mapper". Both of them have been implemented in the paper cited in [10]. Anycast Resolving Layer (ARL) [11] is a new IETF draft and it tries to use a sub layer to resolve anycast address.
If you want to know the recent work for anycast, you can go to Practical Anycasting[12] to get more information. On the website, people discuss how to implement and deploy the anycast service in real world. Someone post a mail in the maillist to encourage everyone to join the anycast work group and deploy anycast service. It was said “Currently, IPv6 Anycast is only used at limited area for limited purpose. It is a pity that IPv6 Anycast is not widely used. This situation should be changed”. [13]

So the development of a whole new API for anycast is needed. Reference [2] have the similar idea to use pseudo mobile IP for anycast, but it is not enough. We need to patch the MobileNode to support server register and thus propose a pseudo-anycast base on MobileIP.

**Pseudo-anycast based on MobileIP**

When CN wants to connect with server, the flowchart is almost the same to the regulation of RFC3775 except we have modified the second step. When CN begins to connect, HA will choose the best MN (server) to execute standard MobileIP linking process. HA after Patching will cheat on CN and MN. HA needs to remember many different Index tables. But all of the MN and CN doesn’t need Patch for support our idea. (figure1)

![Figure 1: CN wants to connect server with TCP](image)

RFC3775 also explains what should be done if CN didn’t support MobileIPv6. If MobileIPv6 couldn’t identify HOTI and COTI that means CN doesn’t support MobileIP. CN will send a reponse with ICMPv6 Bad Parameter-Unrecognized — Next Header Type Encountered (ICMPv6 Type 4, Code 1) message. Upon receipt of the ICMPv6 message, the mobile node and HomeAgent will record the CN that doesn’t support MobileIPv6. HomeAgent will be a relay to forward the data flow between MN and CN. (figure2)

![Figure 2: CN (don’t support Mobile IPv6) wants to connect server with TCP](image)

At first, when servers try to register with the Home Agent, all of the server will share the same HoA. Server will not send special register but binding update. The rules of choosing server

In order not to modify RFC3775 to achieve the function of Anycast, we will not ask server to send the application message to HomeAgent (ex: CPU load, Memory usage... etc). We just use the RTT to decide the linking quality, priority and weight for each server. By RTT, we can also identify the server alive or die.

Two methods we proposed are:

1. HomeAgent will periodically send the ICMPv6 request to each server. (Send to ff02::1 or by the Mobile Node-list). It is simple and useful.

2. Let HA sent a shorter life time of ICMPv6 Home Prefix Advertisement message regularly. MN will send ICMPv6 Home Prefix Solicitation message back after receiving.

We will basically get the same effect no matter which method we use. However, the linking quality may be incorrect under the instant, so we put the result into smoothed RTT equation[14]:

\[
\text{smoothed RTT} = \alpha \times \text{RTT} + (1 - \alpha) \times \text{smoothed RTT}
\]

\[\alpha \approx 0.5\]
\[ SRTT = a \cdot SRTT + (1-a) \cdot RTT \]

RTT represents the time of receiving reaction information after sending. \( a \) is set to 0.9. We can thus calculate the smoother number of linking quality.

Citing from reference [15], we can find some parameters to decide the network state. We send the request in interval \( H \) and it is calculated by

\[ H = DefaultInterval \times (1 + \delta) \]

Where DefaultInterval is a constant and \( \delta \) is a random value uniformly distributed between -0.5 and 0.5 to represent the fluctuation of loads of computers or networks.

From [16], we get the equation to calculate the priority and decide which server should be

\[
\text{Server Priority} = \frac{1}{\sum \frac{SRTT[i]}{1 + T[i]}}
\]

Each time when HomeAgent get the anycast connect request form the CN, the HomeAgent will random generate a number (from 0-1). For example, if we have four servers and their server priority is 0.3, 0.2, 0.35 and 0.15. Then we will get the server list with probability:

Server1 (0-0.3)
Server2 (0.3-0.6)
Server3 (0.5-0.85)
Server4 (0.85-1.0)

If random generate number is 0.7, HomeAgent will connect with Server3.

The following is our pseudo-code.

```plaintext
# for HomeAgent
Get binding message
{  
  If (address is anycast address)
    
    Add DB (anycast address)(server list)
    /* Pay attention, the server sends the normal binding message, but the HomeAddress is anycast address. */
  
  Else
    
    Get connect message from CN
    
    {  
      If (the dest is anycast address)
        
        X = rand(1);  
        Connect server = which_serverDB(anycast, x)
      
      Else
        
        Run for normal MobileIP
      
    }
  
    Thread
    
    {  
      SRTT DB (anycast)
      For (i=0, DBend, i++)
        
        ServerPriority(anycast)
        
        # Find the server is down (serverdown parameter)
        
        If the server priority < 0.1
          
          Send BU to CN to change server
        
    }
}
```

**Simulation and Results**

We use OMNet++ to simulate the environments of our idea.

In our configuration, we have 20 CNs and four servers. The server provides the UDP service and each CN asks 32kbps for UDP service. The topology looks like in Figure 3.

![Figure 3: Simulation Topology](image)

The simulation was run for 800 seconds, DefaultInterval is set to 20sec and timeout is set to 10sec. CN will begin to send the connect