TOA & UOA 协议设计文档

背景

在full nat 模式下 real server 没办法拿到VIP VPORT 和CIP CPORT 地址，拿到的地址是 RS 的 IP 和 port 以及 ttgw的内网IP 和port。有一些业务的业务逻辑依赖 CIP CPORT ，有一些统计依赖于VIP和VPORT， 因此在 real server上拿到 VIP VPORT 和CIP CPORT 是full nat模式需要支持的。

当前状态

当前 TOA 会把 CIP CPORT 和VIP VPORT 塞入到TCP OPTION字段中。UOA会把 CIP CPORT 和VIP VPORT 塞入到IP OPTION中。

当前可能存在的问题

1. TOA 和 UOA 添加 CIP CPORT 和 VIP VPORT的方式不一样。需要两个模块取数据
2. TOA 之前只支持 CIP CPORT 并没有 VIP VPORT ，现在数据结构组织不太合理
3. TOA 和UOA 最开始没有考虑到IPV6支持，在IPV6 情况下 TCP OPTION 不能塞入数据。

改造方案

统一 TOA 和UOA，将 VIP VPORT 和CIP CPORT 通过 ip option 发送给 real server。

Load Balancer 与 Real Server 通信协议

该协议描述了44（ipv4->ipv4），46，64，66等所有环境下UOA与TOA模块的LB与RS之间的通信规范。

协议使用IP option作为传递uoa，toa信息的载体。

1. ipv4 option 设计

```
1
2 +--------+--------+--------+--------+
3 |                                   |
4 +--------+--------+--------+--------+
```

```
|  custom ipv4 header(5 * 4 bytes)  |
+--------+--------+--------+--------+
|  no specific change except hdrlen |
+--------+--------+--------+--------+
|                                   |
+--------+--------+--------+--------+
|                                   |
+--------+--------+--------+--------+
|option  |length  |operation|padding|
+--------+--------+--------+--------+
|      CPORT      |      VPORT      |
+--------+--------+--------+--------+
|                CIP                |
+--------+--------+--------+--------+
|                VIP                |
+--------+--------+--------+--------+

```

Type: 31

length: IPv4 payload: (1 + 1 + 1 + 1 + 2 + 2 + 4 + 4) = 16

IPv6 payload: (1 + 1 + 1 + 1 + 2 + 2 + 16 + 16) = 40

Operation: the lowest bit indicates the option payload is IPv4(0) or IPv6(1).

Padding: 0

2. ipv6 option 设计

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  NextHeader0  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
```

```
|                                                               |
+                     Destination Address                       +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  NextHeader1  | Header Length |  OptionType   | OptionLength  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         CPort                 |             VPort             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                        Client Address                         +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                        Virutal Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

在IPv6协议的规范中， ipv6头部中的next header表明后面的option header的类型, 而option header中的next header表明跟在后面的TCP协议的类型。

NextHeader0 = 60 (Destination Option)

NextHeader1 = 6 (TCP)

Header length: IPv6 payload: ((1 + 1 + 1 + 1 + 2 + 2 + 16 + 16) / 8 - 1) = 4

IPv4 payload: ((1 + 1 + 1 + 1 + 2 + 2 + 4 + 4) / 8 - 1) = 1

OptionType = 31

OptionLength: ipv6 payload: （2 + 2 + 16 +16） = 36

ipv4 payload: (2 + 2 + 4 + 4) = 12

ipv6 rfc:https://tools.ietf.org/html/rfc2460

3. TOA/UOA ip option整体设计

```
union two_addr{
    struct{
        unsigned char saddr[4];
        unsigned char daddr[4];
    }ipv4;
    struct{
        unsigned char saddr[16];
        unsigned char daddr[16];
    }ipv6;
};

struct ip_option{
    union{
        struct{
            __u8 type;
            __u8 length;
            __u8 operation;
            __u8 padding;
        }ipv4;
        struct{
            __u8 nexthdr;
            __u8 hdrlen;
            __u8 option;
            __u8 optlen;
        }ipv6;
    }header;

    __be16 sport, dport;

    union two_addr addrs;
};

#define IPV4_OPTION_TYPE 31
#define IPV6_HEADER_DST 60
#define IPV6_HEADER_OPTION 31

```

其中头部4字节根据使用的是ipv4 option或ipv6 option有所变化。后面的空间存储携带需要的cport, vport, cip, vip。 整个结构的长度根据内部负载的是ipv4 或ipv6地址而有所变化。

port 和 ip地址都使用网络字节序（大端序）。

因为 TOA/UOA 的ip option协议与TCP/UDP 无关， 且option头部支持两种外层协议（ipv4/ipv6），option尾部支持两种类型（ipv4/ipv6）的地址内容， 所以该协议同时支持2 * 2 * 2 = 8种场景。

用户程序与模块接口

1.TOA

用户态程序与TOA模块的通信使用getsockname和getpeername，依然使用struct sockaddr作为通信格式。

注意因为64场景的存在，一个ipv4连接依然可能拿到ipv6地址，因此struct sockaddr 必须传入足够的长度，以免程序出错。即在可能出现ipv6地址的情况下，必须使用 struct sockaddr_in6。

```
1
2  void sockaddr_display(struct sockaddr_in6* addr)
3  {
4      const int addr_buf_size = 100;
5      char addr_buf[addr_buf_size];
6
7      if  (addr->sin6_family == AF_INET)
8      {
9          struct sockaddr_in* sa = (struct sockaddr_in*)addr;
10         printf("ipv4: %s:%d\n", inet_ntop(AF_INET, &sa->sin_addr,
11                 addr_buf,  addr_buf_size), ntohs(sa->sin_port));
12     }
13     else if  (addr->sin6_family == AF_INET6)
14     {
15         struct sockaddr_in6* sa = (struct sockaddr_in6*)addr;
16         printf("ipv6: %s:%d\n", inet_ntop(AF_INET6, &sa->sin6_addr,
17                 addr_buf, addr_buf_size), ntohs(sa->sin6_port));
18     }
19     else
20         printf("invaild sockaddr\n");
21 }
22
23 void checksockname(int fd)
24 {
25     struct sockaddr_in6 addr;
26
27     int sin_size = sizeof(addr);
28     printf("getpeername: ");
29     if (getpeername(fd, (struct sockaddr *)&addr, (socklen_t *)&sin_size)
30             == 0)
31         sockaddr_display(&addr);
32     else
33        printf("error\n");
```

```
34
35      sin_size = sizeof(addr);
36      printf("getsockname: ");
37      if (getsockname(fd, (struct sockaddr *)&addr, (socklen_t *)&sin_size)
38              == 0)
39          sockaddr_display(&addr);
40      else
41          printf("error\n");
42  }
```

2.UOA

用户态程序与UOA模块的通信使用getsockopt系统调用。

因为历史原因，当前维护两个版本的接口，v0为原始接口，v1为现在接口。

```
1  // kernel module
2
3  enum {
4      UOA_SO_BASE          = 2048,
5      /* set */
6      UOA_SO_SET_MAX       = UOA_SO_BASE,
7      /* get */
8      UOA_SO_GET_LOOKUP    = UOA_SO_BASE,
9      UOA_SO_GET_LOOKUP1   = UOA_SO_GET_LOOKUP + 1,
10     UOA_SO_GET_MAX       = UOA_SO_GET_LOOKUP1,
11 };
12
13 union inet_addr {
14     struct in_addr      in;
15     struct in6_addr     in6;
16 };
17
18 // v0 param,  for compatibility
19 struct uoa_param_map {
20     /* input */
21     __be16          af;
22     union inet_addr  saddr;
23     union inet_addr  daddr;
24     __be16          sport;
25     __be16          dport;
26     /* output */
27     union inet_addr  real_saddr;
28     __be16          real_sport;
29 } __attribute__((__packed__));
30
```

```

struct four_tuple{
    unsigned int type; // indicate this is ipv4 or ipv6 addresses;
    __be16 sport, dport;
    union two_addr addrs;
};

// v1 param
union uoa_sockopt_param{
    struct four_tuple input;
    struct four_tuple output;
};
```

```
// user interface v0
struct uoa_param_map param;
 int param_len = sizeof(param);

if  (getsockopt(fd, IPPROTO_IP, UOA_SO_GET_LOOKUP, &param,
(socklen_t*)&param_len) == 0)
    /* this get client ip and client port*/;
else
    printf("get sock opt failed\n");
```

```
// user interface v1
union uoa_sockopt_param param;
int param_len = sizeof(union uoa_sockopt_param);

if  (getsockopt(fd, IPPROTO_IP, UOA_SO_GET_LOOKUP1, &param,
(socklen_t*)&param_len) == 0)
    /*this get client ip, client port, virt ip, virt port*/;
else
    printf("get sock opt failed\n");
```