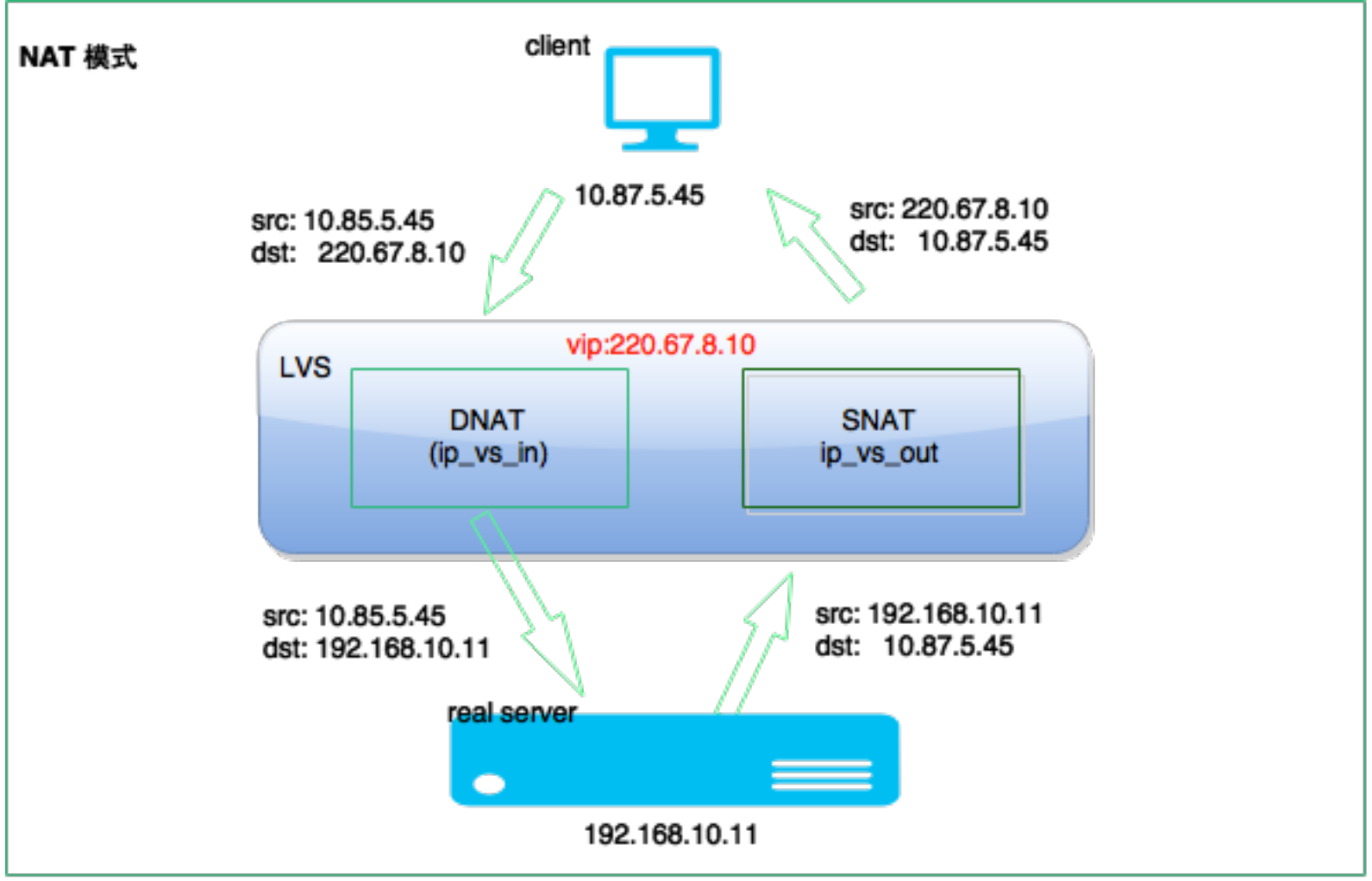
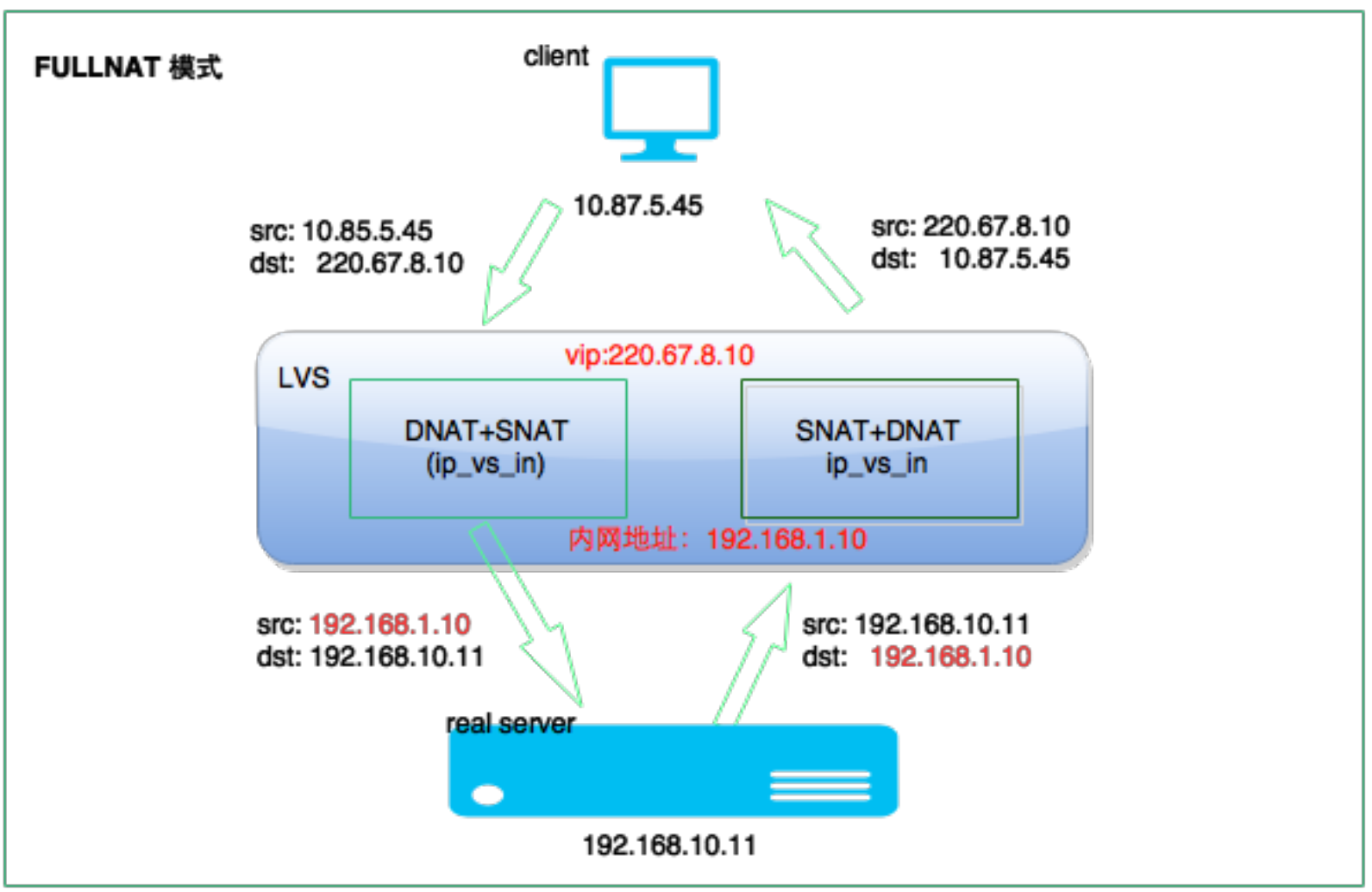
lvs 负载均衡fullnat 模式clientip 怎样传递给 realserver

关于LVS和FULLNAT的介绍可以看一下 淘宝吴佳明(普空)的视频 http://blog.aliyun.com/1750 ，
FULLNAT模式很大简化了LVS的配置和部署，目前淘宝和百度基本上都在使用FULLNAT模式来作为接入侧的
负载均衡模式.

百度的LVS叫做BVS, Baidu Virtual Server, 是在LVS基础上修改的增加了L3 Though 和
SYN Porxy，貌似也是吴佳明(普空)在百度搞的, 类似FULLNAT 项目.

下面的图来自吴佳明(普空)的PPT, 自己重画了一遍，关于NAT和FULLNAT的区别如下图所示:



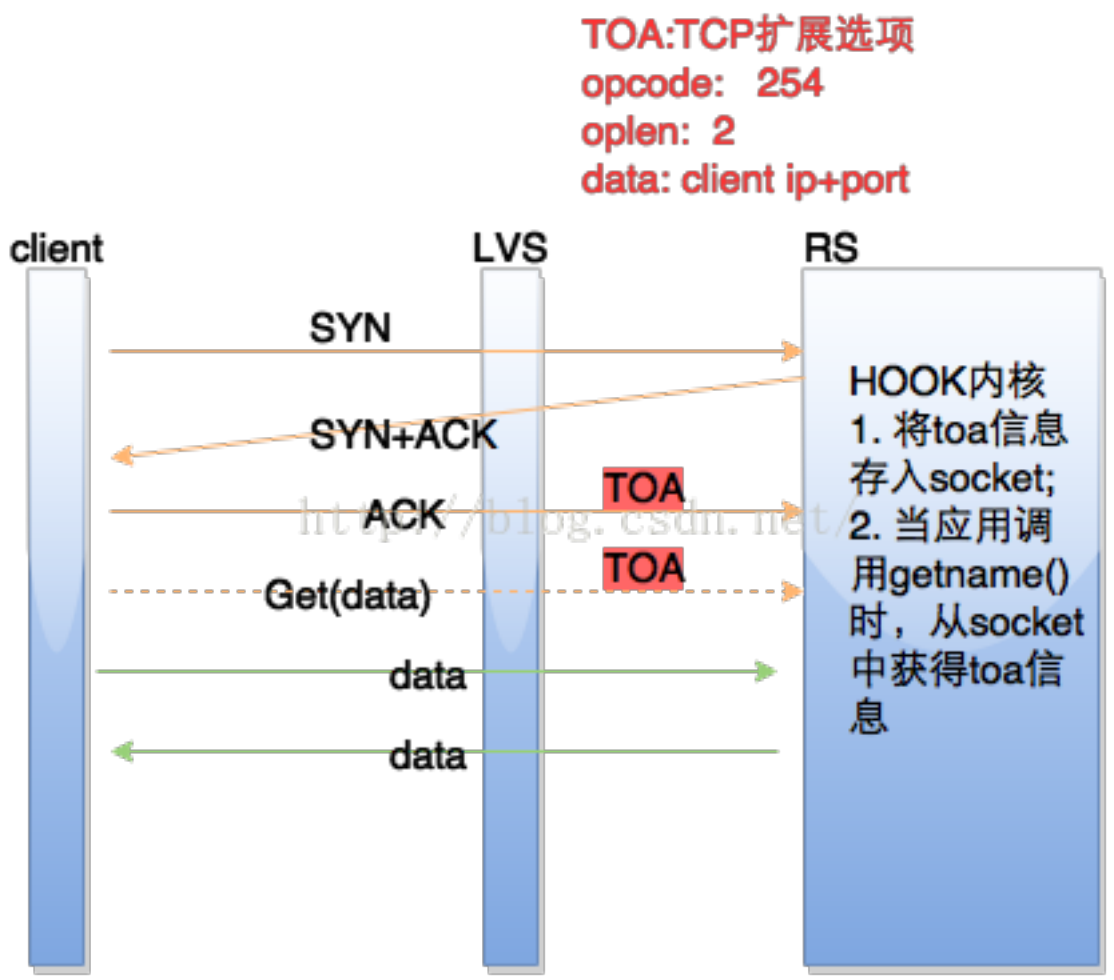


看完上图后发现 FULLNAT有一个问题是：RealServer无法获得用户IP；淘宝通过叫TOA的方式解决的，

主要原理是：将client address放到了TCP Option里面带给后端RealServer，RealServer收到后保存在socket

的结构体里并通过toa内核模块hook了getname函数，这样当用户调用getname获取远端地址时，返回的是保
存在socket的TCPOption的IP. 百度的BVS是通过叫ttm模块实现的，其实现方式跟toa基本一样，只是没有开源.

实现原理图如下：



下面看下上面说的逻辑的实现代码 https://github.com/alibaba/LVS

lvs侧在TCP报文的选项中插入clientip代码: tcp_fnat_in_handler()

```
01056:       * for syn packet
01057:       * 1. remove tcp timestamp opt,
01058:       *    because local address with diffrent client have the diffrent timestamp;
01059:       * 2. recompute tcp sequence
01060:       * 3. add toa
01061:       */
01062:      if (tcph->syn & !tcph->ack) {
01063:          tcp_opt_remove_timestamp(tcph);
01064:          tcp_in_init_seq(cp, skb, tcph);
01065: #ifdef CONFIG_IP_VS_IPV6
01066:          if (cp->af == AF_INET6)
01067:              tcp_opt_add_toa_v6(cp, skb, &tcph);
01068:          else
01069: #endif
01070:              tcp_opt_add_toa(cp, skb, &tcph);
01071:      }
01072:
```

RS侧收到建连报文时，取出toa里面的client ip和port 存放在socket的use_data里,toa.c

```
00196: static struct sock *
00197: tcp_v4_syn_recv_sock_toa(struct sock *sk, struct sk_buff *skb,
00198:                struct request_sock *req, struct dst_entry *dst)
00199: {
00200:     struct sock *newsock = NULL;
00201:
00202:     TOA_DBG("tcp_v4_syn_recv_sock_toa called\n");
00203:
00204:     /* call orginal one */
00205:     newsock = tcp_v4_syn_recv_sock(sk, skb, req, dst);
00206:
00207:     /* set our value if need */
00208:     if (NULL != newsock && NULL == newsock->sk_user_data) {
00209:         newsock->sk_user_data = get_toa_data(skb);
00210:         if (NULL != newsock->sk_user_data)
00211:             TOA_INC_STATS(ext_stats, SYN_RECV_SOCK_TOA_CNT);
00212:         else
00213:             TOA_INC_STATS(ext_stats, SYN_RECV_SOCK_NO_TOA_CNT);
00214:         TOA_DBG("tcp_v4_syn_recv_sock_toa: set "
00215:             "sk->sk_user_data to %p\n",
00216:             newsock->sk_user_data);
00217:     }
00218:     return newsock;
```

HOOK挂载：

```
00268:     inet_stream_ops_p->getname = inet_getname_toa;
00269:     TOA_INFO("CPU [%u] hooked inet_getname <%p> --> <%p>\n",
00270:         smp_processor_id(), inet_getname, inet_stream_ops_p->getname);
00271:
00272: #ifdef CONFIG_IP_VS_IPV6
00273:     inet6_stream_ops_p->getname = inet6_getname_toa;
00274:     TOA_INFO("CPU [%u] hooked inet6_getname <%p> --> <%p>\n",
00275:         smp_processor_id(), inet6_getname, inet6_stream_ops_p->getname);
00276: #endif
00277:
00278:     ipv4_specific_p->syn_recv_sock = tcp_v4_syn_recv_sock_toa;
00279:     TOA_INFO("CPU [%u] hooked tcp_v4_syn_recv_sock <%p> --> <%p>\n",
00280:         smp_processor_id(), tcp_v4_syn_recv_sock,
00281:         ipv4_specific_p->syn_recv_sock);
00282:
00283: #ifdef CONFIG_IP_VS_IPV6
00284:     ipv6_specific_p->syn_recv_sock = tcp_v6_syn_recv_sock_toa;
00285:     TOA_INFO("CPU [%u] hooked tcp_v6_syn_recv_sock <%p> --> <%p>\n",
00286:         smp_processor_id(), tcp_v6_syn_recv_sock,
00287:         ipv6_specific_p->syn_recv_sock);
00288: #endif
```

当应用层调用getpeername() 或者 getsocketname() 时，会进入到

inet_getname_toa,如果存在toa信息则将
socket里存放的真是的clientip 返回给应用层。