tcp\_tw\_recycle和tcp\_timestamps导致connect失败问题

近来线上陆续出现了一些connect失败的问题，经过分析试验，最终确认和proc参数tcp\_tw\_recycle/tcp\_timestamps相关；

# 现象

第一个现象：模块A通过NAT网关访问服务S成功，而模块B通过NAT网关访问服务S经常性出现connect失败，抓包发现：服务S端已经收到了syn包，但没有回复synack；另外，模块A关闭了tcp timestamp，而模块B开启了tcp timestamp；

第二个现象：不同主机上的模块C（开启timestamp），通过NAT网关（1个出口ip）访问同一服务S，主机C1 connect成功，而主机C2 connect失败；

# 分析

根据现象上述问题明显和tcp timestmap有关；查看linux 2.6.32内核源码，发现tcp\_tw\_recycle/tcp\_timestamps都开启的条件下，60s内同一源ip主机的socket connect请求中的timestamp必须是递增的。

源码函数：tcp\_v4\_conn\_request(),该函数是tcp层三次握手syn包的处理函数（服务端）；

源码片段：

if (tmp\_opt.saw\_tstamp &&

tcp\_death\_row.sysctl\_tw\_recycle &&

(dst = inet\_csk\_route\_req(sk, req)) != NULL &&

(peer = rt\_get\_peer((struct rtable \*)dst)) != NULL &&

peer->v4daddr == saddr) {

if (get\_seconds() < peer->tcp\_ts\_stamp + TCP\_PAWS\_MSL &&

(s32)(peer->tcp\_ts - req->ts\_recent) >

TCP\_PAWS\_WINDOW) {

NET\_INC\_STATS\_BH(sock\_net(sk), LINUX\_MIB\_PAWSPASSIVEREJECTED);

goto drop\_and\_release;

}

}

tmp\_opt.saw\_tstamp：该socket支持tcp\_timestamp

sysctl\_tw\_recycle：本机系统开启tcp\_tw\_recycle选项

TCP\_PAWS\_MSL：60s，该条件判断表示该源ip的上次tcp通讯发生在60s内

TCP\_PAWS\_WINDOW：1，该条件判断表示该源ip的上次tcp通讯的timestamp 大于 本次tcp

分析：主机client1和client2通过NAT网关（1个ip地址）访问serverN，由于timestamp时间为系统启动到当前的时间，因此，client1和client2的timestamp不相同；根据上述syn包处理源码，在tcp\_tw\_recycle和tcp\_timestamps同时开启的条件下，timestamp大的主机访问serverN成功，而timestmap小的主机访问失败；

参数：/proc/sys/net/ipv4/tcp\_timestamps - 控制timestamp选项开启/关闭

/proc/sys/net/ipv4/tcp\_tw\_recycle - 减少timewait socket释放的超时时间

# 解决方法

echo 0 > /proc/sys/net/ipv4/tcp\_tw\_recycle;

tcp\_tw\_recycle默认是关闭的，有不少服务器，为了提高性能，开启了该选项；

为了解决上述问题，个人建议关闭tcp\_tw\_recycle选项，而不是timestamp；因为 在tcp timestamp关闭的条件下，开启tcp\_tw\_recycle是不起作用的；而tcp timestamp可以独立开启并起作用。

源码函数： tcp\_time\_wait()

源码片段：

if (tcp\_death\_row.sysctl\_tw\_recycle && tp->rx\_opt.ts\_recent\_stamp)

recycle\_ok = icsk->icsk\_af\_ops->remember\_stamp(sk);

......

if (timeo < rto)

timeo = rto;

if (recycle\_ok) {

tw->tw\_timeout = rto;

} else {

tw->tw\_timeout = TCP\_TIMEWAIT\_LEN;

if (state == TCP\_TIME\_WAIT)

timeo = TCP\_TIMEWAIT\_LEN;

}

inet\_twsk\_schedule(tw, &tcp\_death\_row, timeo,

TCP\_TIMEWAIT\_LEN);

timestamp和tw\_recycle同时开启的条件下，timewait状态socket释放的超时时间和rto相关；否则，超时时间为TCP\_TIMEWAIT\_LEN，即60s；

内核说明文档 对该参数的介绍如下：

tcp\_tw\_recycle - BOOLEAN

Enable fast recycling TIME-WAIT sockets. Default value is 0.

It should not be changed without advice/request of technical

experts.