TCPCopy Manual

目 录

1 引言

XCopy 是由网易主导，多家公司成员[1]参与开发的具有在线 server 流量复制功能的一系列开源产品的总称，目的是复制在线 server 流量到测试系统中去，并引入在线的复杂性到测试系统，从而可以在测试系统中充分暴露在线的问题，帮助提前解决在线问题，降低上线失误率或者实现零失误。

XCopy 系列包括 TCPCopy、UDPCopy、MysqlCopy 等开源产品。

XCopy 分为在线版本和离线版本，在线版本主要用来实时捕获在线数据包，而离线版本主要从 pcap 格式的文件中去读取在线数据包。由于离线版本依赖于抓包工具，在压力比较大的场合，抓包工具一般丢包非常严重，而且还会严重影响在线 IO，因此一般不推荐在高压情况下使用离线版本（以下内容一般不声明的话，默认是在线实时复制版本）。

XCopy 可以应用在以下领域：

1） 性能压力测试，如系统性能瓶颈查找，内核参数调优，不同程序性能比较等

2） 系统可用性测试

3） 冒烟测试

4） 回归测试，特别是针对技术类重构的测试

5） 。。。

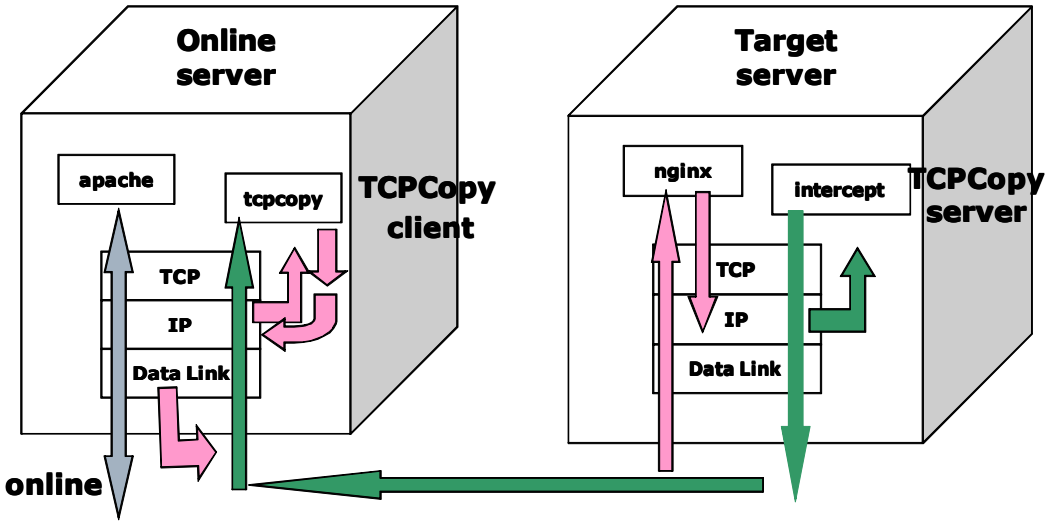
TCPCopy 是 XCopy 系列中最重要的一个组件，其功能是复制在线 server 数据包，修改 TCP/IP 头部信息，发送给测试服务器，达到欺骗测试服务器的 TCP 程序的目的，从而为欺骗测试服务器上面的上层应用打下坚实基础。

由于互联网大部分应用是基于 TCP 的，并且其中的大部分应用是很容易被欺骗的，因此只需 TCPCopy 就能欺骗大部分上层协议，如 HTTP 协议、memcached 协议，POP3 协议等等。

本文的结构如下：

[1] 目前成员包括淘宝，畅游等公司的相关人员

2 TCPCopy 原理及介绍



2.1 TCPCopy 原理图

2.1 原理

TCPCopy 包含两部分：TCPCopy client（*tcpcopy*） 和 TCPCopy server（*intercept*）。其中，TCPCopy client 运行在在线服务器端，用来捕获在线请求数据包，而 TCPCopy server 运行在测试服务器（或者 target 服务器）端，用来做一系列辅助配合工作，如返回响应包头信息、维护路由信息和决定响应包的最终命运等。

TCPCopy client 默认利用 raw socket input 技术，从在线服务器的 IP 层抓取在线应用的请求数据包，修改这些包的属性，利用 raw socket output 技术（packet injection 技术之一）发送给测试服务器。

发送过去的数据包会被测试服务器的 TCP/IP 协议栈所识别，从而数据包里面的请求（针对带有 payload 的数据包）可以进入测试服务器的上层应用，测试服务器的上层应用一般会处理这个请求，传递响应给测试服务器的 TCP/IP 协议栈。

TCPCopy server 主要负责传递响应包头信息给 TCPCopy client。IP Queue 内核模块（内核版本<3.5，默认采用此模块）会在 IP 层传递响应包给 netlink socket，通过此接口我们可以获得这些响应包的信息，并且可以决定这些响应包的最终命运。

由于 TCP 交互是相互的，一般情况下，为了使 TCPCopy client 能够顺利发送下一个数据包信息，TCPCopy server 必须传递响应包的包头信息给 TCPCopy client，我们专门采用了一个 TCP 连接来传递这些信息，TCPCopy client 得到这些响应包头信息后，就可以继续修改在线数据包的 ack 等属性，进入下一个发包循环。

这里需要注意的是测试服务器的响应信息一般最终都会被 TCPCopy server 丢弃掉，并不

会返回给客户端，这样做的好处有如下几个方面：

1）不会增加出口带宽费用，节省测试成本

2）不会在互联网产生 ghost 数据包

3）不会对客户端的 TCP/IP 协议栈进行干扰

2.2 TCPCopy vs Tcpreplay

tcpdump+Tcpreplay 或者 *wireshark*+Tcpreplay 可以用来回放在线流量，这种方案可以解决 TCP 层以下的问题，如防火墙问题，然而，此方案仍有如下缺陷：

1）*tcpdump* 抓在线数据包，必然或多或少影响在线 IO，压力大的时候尤其明显

2）*tcpdump* 自身会丢包，特别是压力大的时候，丢包会很严重，远远超过 TCPCopy

3）*tcpdump* 抓的包需要保存下来，必然使其抓的包有限，一般很少抓几天的数据包

4）利用 Tcpreplay 重放，一般需修改数据包的源 IP 地址，这样必然跟在线环境不一样，比如对内核协议栈的冲击就很不一样

5）由于是离线回放，导致 Tcpreplay 回放的时候网络环境可能已经与抓包的时候不同了，而且 Tcpreplay 很难根据不同的环境做相应的调整，从而导致回放的效果与在线会有一定的差距

6）Tcpreplay 官方称不支持到服务端， 即回放对上层应用无效

7）Tcpreplay 相对 TCPCopy，使用更为复杂

因此，即使要测试 TCP 层以下的应用，也推荐使用 TCPCopy（加代理即可实现）。

TCPCopy 主要用来解决 TCP 层及其以上（如 http 协议）的流量复制问题，用于 server 的流量回放领域。总体来说，TCPCopy 有如下优点：

1）TCPCopy 能够对 server 进行回放，不仅可以离线回放，还可以实时回放

2）TCPCopy 实时复制在线请求包给测试服务器，并不需要大量 IO 操作，因此不影响在线 IO

3）实时回放所在的网络环境与当时的在线环境几乎是一样的

4）TCPCopy 不会去修改数据包的源 IP 地址

5）TCPCopy 实时复制转发过去的数据包，能够更好地继承在线数据包的网络延迟特征

6）TCPCopy 使用非常简单

2.3 影响 TCPCopy 的因素

从图 2.1 可以看出，可能影响到 TCPCopy 的地方有如下几个：

1） Raw socket input 接口

2） Raw socket output 接口

3） 去往测试服务器的路上

4） 测试系统 OS

5） 后端应用

6） Netlink socket 接口

我们分别介绍如下：

2.3.1 Raw socket input 接口

抓包默认利用了 raw input socket 接口，由于系统参数设置不合理的原因，系统内核并不一定能把所有你关心的包都给你，压力越大，一般丢得越多，所以这方面相应的系统参数要设置好。

值得注意的是，TCPCopy 还支持利用 pcap 接口进行抓包，可以充分发挥 pcap 库的内核过滤功能，能够极大地提高抓包的效率，而且利用 pcap 接口，还能充分利用 pfring（目前还不是很成熟，使用需谨慎），能够进一步降低丢包率。

2.3.2 Raw socket output 接口

发包利用了 raw socket output 接口，系统内核由于种种原因，并不一定能够成功发送所有包，特别是压力比较大的时候（需要设置好相应的系统参数）。

例如，当所抓的包的大小超过发送 MTU 的值，在 0.5 以下版本，就会发送不出去，所以最好采用 0.5 版本及其以上版本。

2.3.3 去往测试服务器的路上

发包以后，如果跨网段的话，路途可能不会一帆风顺，因为被复制的数据包的源 IP 地址还是客户端的 IP 地址（这是为了引入在线复杂性，不会去改变数据包的源 IP 地址），可能会造成路途中的安全检测设备认为这些数据包是非法的或者是伪造的数据包，将 *tcpcopy* 转发的数据包给丢弃掉。这时候，在测试服务器端利用 *tcpdump* 来抓包，将不会抓到任何复制转发过来的数据包。你可以进一步在同一网段试验一下，如果同网段复制转发成功而跨网段复制转发不成功，那么跨网段转发出去的数据包应该在中途被丢弃掉了。

需要注意的是，如果跨网段请求复制不出去，你可以先复制请求到本网段的代理，通过代理再把请求传递给跨网段的测试服务器。

2.3.4 测试系统 OS

1）有些机器设置了 *rpfilter*（反向过滤技术），将会判断源 IP 地址是否是所信任的 IP 地

址，如果不是，将会在 IP 层丢弃掉该数据包。因此在测试服务器端，如果利用 *tcpdump* 能够抓到转发过来的数据包，但利用类似于 netstat 的工具却看不到目标应用端口的任何 TCP 连接存在，那么很有可能系统设置了 *rpfilter*。

这种情况下，可以查看系统有没有设置 *rpfilter*，如果设置了，尝试去掉此设置后，利用类似于 *netstat* 的工具能不能检测到目标应用端口的 TCP 连接的存在（也就是这些被复制转发的数据包能否穿过测试服务器的 IP 层）。

2）*iptables* 的问题，如果全局 *iptables* 设置有冲突，会导致系统产生一系列诡异问题，比如如下例子：

https://github.com/wangbin579/tcpcopy/issues/56

3）内核中同时存在 IP Queue 和 NFQueue 的情况下，IP Queue 模块就存在被损坏的可能。这种情况常见于在一切设置都合理的情况下，复制数据包给测试机器的时候，测试机器上面待测试的应用的连接只有 SYN_RECV 一种状态，这时候可以采用 NFQueue（*configure --enable-nfqueue*）方式来运行。

4）在某些情况下，比如在极端情况下，如果你利用压力测试工具来发请求，也许压力测试工具已经测试完了，而转发的数据包中的第一个 syn 包还没有到达测试服务器，这样就造成原先串行的请求变成了并行请求，造成大量 syn 涌向测试服务器，这样就可能被测试服务器误报为 syn flood 攻击，一旦 syn 包被丢弃，那么相应的应用请求将无法传递给测试服务器(TCPCopy 针对三次握手的数据包，不会去重传)。

针对这种情况，要优化系统参数，才能有好的效果，或者你可以采用一些长连接的压力测试工具来发请求，比如 *httpsender*。

2.3.5 测试服务器的应用程序

并不是每个请求过去，测试服务器上面的应用程序都能够及时处理或者都能处理，有可能触发了应用程序的 bug，导致请求迟迟没有响应，也有可能应用程序的协议不支持 *tcpcopy* 提前发送请求（类似 pipeline 形式的请求），仅仅处理 socket buffer 中的第一个请求，这些情况都会导致请求大量丢失。

2.3.6 Netlink socket 接口

下述内容只针对采用默认 IP Queue 模式的情况下，而对于 NFQueue 模式则无效。

假设后端响应都顺利，压力比较大的时候，IP Queue 模块传递响应包给 netlink socket 的时候，也会遇到丢包现象，这时候可以通过 *cat /proc/net/ip_queue* 命令，查看 IP Queue 运行情况：

1） 如果 *Queue dropped* 的数值不断增大，则需要修改 ip_queue_maxlen 参数，比如：

echo 4096 > /proc/sys/net/ipv4/ip_queue_maxlen

2） 如果 *Netlink dropped* 的数值不断增大，修改 rmem_max 和 wmem_max 参数，比如：

sysctl -w net.core.rmem_max=16777216

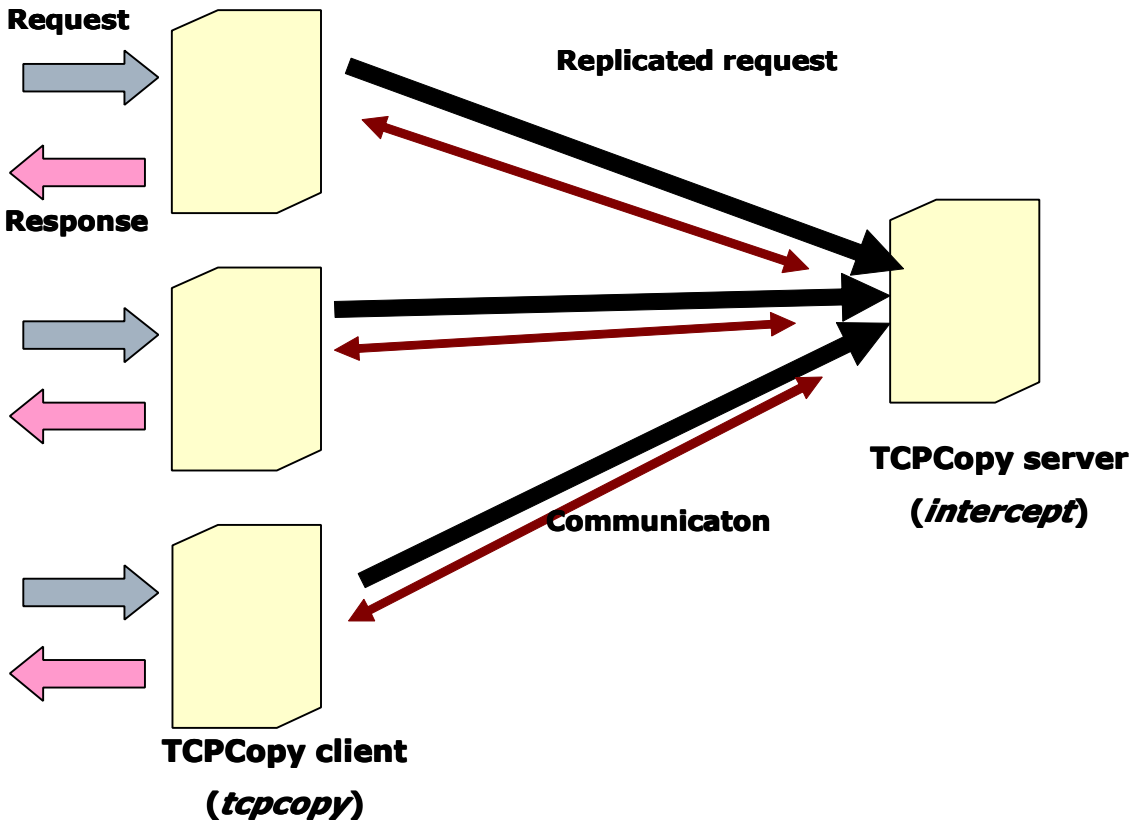
sysctl -w net.core.wmem_max=16777216

3）如果 IP Queue 相关系统参数的值不管设置为多大，*Netlink dropped* 还是在不断增大，可以尝试重启 IP Queue 模块：

modprobe -r ip_queue

modprobe ip_queue

2.4 TCPCopy 分布性特征



2.4 TCPCopy 分布式架构（第二期）

为了充分挖掘在线流量资源，我们可以复制多台在线流量到同一台测试服务器上去，这样可以演绎在分布式攻击（非带宽攻击）情况下，系统能否承受此压力。

值得注意的是，分布式压力测试的时候，同等压力情况下，连接数越少，性能越高，理由如下：因为 TCPCopy 是基于 session 的，每一个 session 由客户端 IP 地址、客户端端口号、服务器 IP 地址和服务器应用端口号决定，因此如果出现大量 session，必然要在 TCPCopy server（非 single 模式）端保存大量的路由信息供响应包头返回使用，这会显著降低 TCPCopy

server(*intercept*)的性能。

2.5 二次开发 TODO

任何 web 回放工具（目前为止，还没有发现同类型的 web 回放工具）,不管是实时的,还是非实时的,都不能一劳永逸地解决所有问题,，因为 TCP 层上面的协议很多，特别是用户自定义的协议，更不可能提前知道该协议的内容，因此二次开发是非常有必要的。我们将在以后版本尽量给出二次开发的接口，以满足各方面应用的需求。

值得注意的是，为了进一步降低二次开发的难度，可以先把请求转发给常规的应用服务器，然后在应用服务器端修改请求内容后再转发给真正待测试的应用服务器。比如，针对 http 协议，为了能够修改 url 里面的参数，先把在线请求复制转发给标准的 web 服务器，在 web 服务器端进行相应修改后，再转发给待测试的程序。

2.6 TCPCopy 优化

要让 TCPCopy 更好地工作，需要在各个方面进行优化，除了上面所讲的影响 TCPCopy 的地方外，我们还可以进行如下方面的优化。

我们主要提供的优化参数如下：

1）TCPCopy server（*intercept*）的-s 参数，如果 session 过多，压力也很大，请修改该参数为更高的值（默认为 65536），建议设置 65536 的倍数

2）TCPCopy（*tcpcopy* 和 *intercept*）的-t 参数，对于 *tcpcopy* 默认为 60s，对于 *intercept* 默认为 120s，如果后端普遍响应很慢或者是长连接应用的场合，建议设置更大一些（intercept 设置的-t 参数值要大于等于 tcpcopy 所设置的-t 参数值）

3）TCPCopy client（*tcpcopy*）的-m 参数（此参数只有在内核版本>=2.6.32 才有效），由于默认所占内存最大 512M，如果内存需要超过此值，请设置合理的值

4）如果仅仅复制在线的一台流量（比如单台在线流量很大）给一台测试服务器，那么推荐使用 single 模式的 TCPCopy 版本，这个模式的版本在 *intercept* 程序中不会保留路由信息，因此 *intercept* 性能会更高（据我们测试，提升 30%～50%性能）

2.7 TCPCopy 不足的地方

1）对于非 single 模式的版本，过多路由条目信息造成 TCPCopy server 性能瓶颈

2）TCPCopy 的性能跟所处理的包的多少关系很大

3）由于 *tcpcopy* 和 *intercept* 程序之间只有一条 TCP 连接来传递响应包头信息，在高压情况下可能会是个瓶颈

4）无法保持不同连接的请求的有序性

5）过分依赖于内核，需系统参数调优才能有好效果

6）一般情况下，不能测试网络底层协议性能瓶颈（不过可通过设置代理测试网络底层协议）

7）TCPCopy 会消耗一定的带宽，主要在转发在线请求数据包方面

8）TCPCopy 文档严重缺乏，有时候还无法及时更新文档，在此表示抱歉

2.8 TCPCopy CHANGES

1）2011.09 发布 TCPCopy 0.1 版本

2）2011.11 发布 0.2 版本，解决若干 bug

3）2011.12 发布 0.3 版本，支持 mysql 请求复制

4）2012.04 发布 0.3.5 版本，新增多重复制功能

5）2012.05 发布 0.4 版本，解决若干 bug

6）2012.07 发布 0.5 版本，重构代码，解决大文件下载和上传的复制问题，
解决抓包超过 MTU 的问题

7）2012.08 发布 0.6 版本，新增离线回放功能

8）2012.10 发布 0.6.1 版本，支持 intercept 多线程模式

9）2012.11 发布 0.6.3 版本, 解决 fast retransmitting 问题

10）2012.11 发布 0.6.5 版本, 支持 *NFQueue*（针对 Linux 高版本内核）

11）2013.03 发布 0.7.0 版本，支持 single 模式、DR 模式（复制给带有 LVS 的子系统），
以及支持响应延迟控制的 paper 模式，并解决 mysql 的若干 bug。

2.9 TODOS

1）新架构（第三期），也是终极架构，目的是为了解决高并发情况下的请求复制问题

2）代码优化

3）英文推广,发 paper

4）预热功能

5）二次开发接口

6）…

3 UDPCopy 原理及介绍

基于 UDP 的复制很简单，一般直接改变 UDP 头部的目的 IP 地址和目的端口，并且如果数据包的长度超过 MTU，则进行 IP 分片即可。

UDPCopy 默认情况下是实时复制版本，由于从 IP 模块末尾抓包，避免了组装 IP 分片的问题，大大简化了程序的设计；对于离线模式的版本，由于抓包一般都是从数据链路层抓包的，所以遇到 IP 分片后的数据包，目前为止还不能正常处理。

4 MysqlCopy 原理及介绍

MysqlCopy 是建立在 TCPCopy 基础上的，针对 mysql 应用的请求复制工具。

MysqlCopy 原先所依据的协议文档地址如下：
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol
由于上述地址已经无效，最新的协议文档如下：
http://dev.mysql.com/doc/internals/en/client-server-protocol.html

这里需要注意的是 MysqlCopy 只支持 4.1 and later，不支持之前的老版本

由于 mysql 协议是有状态的，mysql 协议与其它无状态的协议有如下不同：

1） 中途截获 mysql 请求，一般无法去构造有效请求，因为缺乏前面的有效信息
2） Mysql 会话需要登入过程
3） 如果测试服务器的 mysql 关闭掉会话过程，而在线还在会话，这时候重新建立会话的过程中，还需要重新发送登入过程的 packets 和 prepare 语句的 packets（如果之前会话有 prepare 语句的话）

正因为这些特征，导致了其处理的复杂性，为了降低复杂度，MysqlCopy 分为两种工作方式：

1） skip-grant-tables 工作方式
2) 正常 mysql 工作方式

在线服务器上面的 mysql 程序采用正常的工作方式，测试服务器上面的 mysql 程序可以采用 skip-grant-tables 模式或者正常模式。如果测试服务器采用 skip-grant-tables 来运行 mysql，那么 MysqlCopy 在 configure 的时候需要指定 *./configure --enable-mysqlsgt*；如果测试服务器上面的 mysql 程序跟在线一样，采用正常的模式运行，那么 MysqlCopy 在此 configure 的时候需要指定 *./configure --enable-mysql*。这两种模式所不同的是正常模式需要提供用户名和密码，而且源代码也略微不一样（为了性能考虑，目前采用了从 configure 来区分两种工作方式）。

下面讲述 mysql 协议登入过程的原理：

采用 skip-grant-tables 模式，无需 *intercept* 传回 crypted password，所以只需传递 TCP 和 IP header 给 *tcpcopy* 即可完成 mysql 协议的正常交互，而 mysql 正常模式下，需要传递 server 端的 crypted password 给 *tcpcopy* 以便 *tcpcopy* 来根据 crypted password 修改 mysql 协议的登

入过程的数据包的内容，才能欺骗测试服务器的 mysql 服务器。

最后，对 mysql 进行流量复制时，需要注意如下事项：

1）如果测试服务器的 mysql 需要工作在 skip-grant-tables 模式下，configure 的时候指定 *--enable-mysqlsgt*，编译后的代码仅仅适用于这种工作方式

2）如果采用正常模式，也即非 skip-grant-tables 模式，configure 的时候需指定 *--enable-mysql*，编译后代码仅仅适用于正常模式；执行的时候需要设置-u 参数，也即用户名密码，用户名必须跟在线一致，密码可以不一样，而且最重要的是这些用户在测试系统 user 表中的配置信息，必须要跟在线系统一致（比如测试走默认的“%”，而在线走非默认的“%”，这样配置就不一样，会导致登入过程校验不一样，从而复制失效）。

具体执行命令举例如下（采用 TCPCopy 0.6.3+版本）：

./tcpcopy -x 3306-xxx.xxx.xxx.xxx:3306 -u user1@password1,user2@password2 -d

3）如果采用 sysbench 来测试 MysqlCopy，由于 sysbench 的连接可能是永久连接，所以要求 tcpcopy 运行在 sysbench 启动之前；如果复制在线系统的 mysql 请求，MysqlCopy 需要捕获到新建连接才能开始工作，所以最好能够经常杀连接或者重启应用，使其重新建立到 mysql 的连接，这样就能够被 MysqlCopy 捕获到登入信息和 prepare statement 语句

4）正常模式的 mysql 请求复制，会比 skip-grant-tables 方式传递更多的响应内容，所以推荐后一种方式

5 TCPCopy 用法参考

5.1 How to configure

我们可以在 *configure* 的时候指定 TCPCopy 的运行模式，详细介绍如下：

1) *./configure*

经过编译后，TCPCopy 工作方式为实时复制模式。

值得注意的是，在内核 3.5 之前，*intercept* 默认采用 IP Queue *模块*；在内核>=3.5 后，intercept 采用 NFQueue 模块（因为 3.5 以后就不支持 IP Queue 了）

2) *./configure --enable-nfqueue*

经过编译后，*intercept* 采用 NFQueue （内核版本 3.5 以后默认采用 NFQueue），实时复制模式

3) *./configure --enable-thread*

经过编译后，*intercept* 工作在多线程模式，目的是为了发挥 *intercept* 的潜能

4) *./configure --enable-single*

经过编译后，TCPCopy 工作在非分布式模式，只能复制一台在线流量到测试服务器，*intercept* 不会维护路由信息，以提高单台机器复制的潜能，适合于复制单台在线机器大流量应用的场合

5）*./configure --enable-offline*

经过编译后，工作在离线回放方式，适合于不方便在在线服务器上面实时复制流量的场合。

这里需要注意如下内容：

a）是离线回放所依据的 pcap 文件，最好只包含在线请求数据包，不要包含在线的响应包，这样做的好处就是减少 pcap 文件所占大小，而且会提升离线回放程序的性能

b）抓包生成 pcap 文件的时候，一定要确保不要丢包。

c）离线模式需要有-i 参数，用来指定 pcap 文件所在的路径

6）*./configure --enable-dr*

经过编译后，工作在 DR 模式。DR 模式取之于 LVS 的 DR 应用场景，是为了解决复制在线流量给带有 LVS(由于大部分场景下采用 DR)的子系统。

由于在线流量复制给 LVS 后，LVS 会把不同连接的数据包按照一定的策略分发给不同的测试服务器，导致了与传统 TCPCopy 的不同。

为了解决响应包如何返回的问题，此模式下的 tcpcopy 会把每个会话的路由信息传递给每一台 *intercept*。因此 *tcpcopy* 执行命令的时候会多出-s 参数，用来指定真正的 real server 的 IP 地址，而 *intercept* 则需要设置-x 参数，参数的值就是 LVS 服务器的所用的实际 IP 地址，设置的目地是让 LVS 发出的数据包（即检测 real server 的健康程度的数据包）能够通过测试服务器的 IP 层，这样 LVS 才会认为这些 real server 是健康的，而不是处于崩溃的状态。

理论上，如果数据包到达测试服务器之前，其目的 IP 地址存在 NAT 变换（Destination Network Address Translation.），也就是说目的 IP 地址会被改变的情况下，那么 DR 模式也能适用，这种场景下的应用可以看成是 DR 的一种特殊应用场景。

值得注意的是，如果 LVS 子系统中的一台 real server 机器上面的 *intercept* 进程崩溃后，仅仅重新启动 *intercept*， TCPCopy 不会正常工作，需要重新启动所有 *tcpcopy* 才能顺利工作。

7）*./configure --enable-paper*

默认情况下，响应数据包的网络延迟是不保持的，此模式加入了响应包的返回延迟，目的是更加逼真地模拟在线的情况。

8）*./configure --enable-pcap*

这种模式是为了能够利用 pcap 库进行抓包，利用 pcap 的过滤接口，可以提升抓包的效率，比如当在线机器应用种类比较多，需要复制某一个应用的请求到测试系统的时候，这种方式效率最高。

9）*./configure --enable-debug*

这种模式是为了输出 debug 日志，方便问题诊断的，如果你遇到了无法解决的问题，请采用此模式运行 TCPCopy（一般运行 5 分钟即可）

需要注意的是，采用 *configure* 的时候， *tcpcopy* 和 *intercept* 程序一般需要采用同样的 configure 配置, 以防诡异问题产生。

5.2 How to run

一般 TCPCopy 执行命令可如下：

在测试服务器 (需要 root 用户权限):

采用 IP Queue 模块（内核<3.5，默认采用 IP Queue）:

1) *# modprobe ip_queue # if not running*

2) *# iptables -I OUTPUT -p tcp --sport port -j QUEUE # if not set*

3) *# ./intercept*

Or

采用 NFQueue 模块（内核>=3.5，默认采用 NFQueue）:

1) *# iptables -I OUTPUT -p tcp --sport port -j NFQUEUE # if not set*

2) *# ./intercept*

这里需要注意，*iptables* 命令中的 port 是变量，应根据具体应用项目而定。

在在线服务器端：

 # ./tcpcopy -x localServerPort-targetServerIP:targetServerPort

具体地，可能还需要加入其他参数，下面讲述 TCPCopy 的参数的含义

5.3 TCPCopy client

tcpcopy -h 可以得到帮助文档，不同模式，命令的个数可能有所不同。

下面详细介绍常用参数设置

-x 参数

格式：-x <transfer,>

Transfer 具体格式如下：

服务器对外 IP 地址:服务器应用端口号-测试服务器 IP 地址:测试服务器应用端口

或者

服务器应用端口号-测试服务器 IP 地址:测试服务器应用端口

Transfer 之间用“,”隔开，IP 地址和端口号之间用“:”隔开，服务器应用端口号和测试服务器 IP 地址之间用‘-’隔开

举例：

./tcpcopy -x 80-192.168.0.2:18080

复制在线机器的 80 端口应用的请求到 192.168.0.2 上面的 18080 端口

-i 参数

格式：-i <file>

其中 file 是 pcap 文件的文件路径

需要注意的是此参数只有在离线回放才有效（*configure --enable-offline* 的情况下）

-c 参数

格式：-c <IP>

如果你复制的请求是本地 localhost(127.0.0.1)的请求（比如抓包得到的结果类似于 127.0.0.1:80→127.0.0.1:8080），那么直接复制这样的请求到其它机器上去，一般是不行的，需要修改数据包的源 IP 地址，-c 参数就是用来改变源 127.0.0.1 地址的。

举例:

./tcpcopy -x 8080-192.168.0.2:8080 -c 192.168.0.1

复制 127.0.0.1 上面的 8080 端口应用的请求到 192.168.0.2 上面的 8080 端口，同时修改源 IP 地址 127.0.0.1 为 192.168.0.1 地址

值得注意的是，-c 参数只对源 IP 地址为 127.0.0.1 的数据包有效

-n 参数

如果你要进行多重复制，那么此参数的值就是代表复制过去的流量是在线的 n 倍，倍数越小，效果越好，因为多重复制的原理是修改端口号，因此复制的倍数越大，端口冲突的概率越大，特别是源 IP 地址非常少，短连接的的内网应用场合。系统默认最大值为 1023 倍。

举例

./tcpcopy -x 80-192.168.0.2:8080 -n 3

复制 3 倍的在线服务器的 80 端口应用请求流量到 192.168.0.2 的 8080 端口

-f 参数

如果你要运行多个 *tcpcopy* 实例，复制请求到同一台测试服务器上面去，那么此参数就是为此设置的，常见于逐步增大在线流量的场合。最大值为 1023。

举例：

./tcpcopy -x 80-192.168.0.2:8080

./tcpcopy -x 80-192.168.0.2:8080 -f 1

./tcpcopy -x 80-192.168.0.2:8080 -f 2

这里我们在在线服务器运行三个实例，可以把测试服务器的流量放大到在线的 3 倍。

需要注意的是，如果要用-f 参数，复制给同一台测试服务器上面的同一个应用的不同 *tcpcopy* 实例之间的-f 参数的值是不同的。

-r 参数

如果你想复制在线服务器应用的部分流量，可以采用-r 参数来实现，参数范围是 1～99，其它值都是全流量复制。

举例：

./tcpcopy -x 80-192.168.0.2:8080 -r 20

这里 *tcpcopy* 复制在线服务器 8080 端口应用的 20%流量给后端服务器，需要注意的是 20%是根据 session（这里 session 是由客户端 IP，客户端端口决定）来统计的。

-r 参数常见于对在线应用进行 profile 的场合或者测试服务器配置不如在线服务器的场合。

-m 参数

此参数只有在 Linux 内核 2.6.32 版本及其以后版本才有效，如果内存超过了此设置值，那么 *tcpcopy* 就自动退出，默认是 512M，这是为了保护在线，防止 *tcpcopy* 占用过多内存。

-s 参数

格式：-s <iplist,> real server ip addresses behind lvs

指定真正运行 *intercept* 的 IP 地址列表，通过与这些 IP 地址的 *intercept* 建立连接，可以让 i*ntercept* 顺利返回响应信息。

举例如下：

./tcpcopy -x 80-10.120.12.211:28080 -s 10.120.12.161,10.120.12.162

复制在线 80 端口的流量到 LVS（虚拟 IP 10.120.12.211）子系统中去，并设置 tcpcopy 与 intercept 的通信 IP 地址列表（10.120.12.161 和 10.120.12.162 服务器上面需要部署 28080 端口的应用）。

注意，-s 参数只有 *configure --enable-dr* 的情况下才有效

-t 参数

如果你的在线应用响应非常慢，那么推荐设置更大的 -t 参数值（默认是 60s，如果 60s 没有收到测试服务器的响应，那么这个请求会话就被 *tcpcopy* 给丢弃掉）。

-l 参数

设置错误日志文件的路径

-p 参数

格式:-p <num> remote server listening port

远程 *intercept* 的监听端口，默认是 36524。

-P 参数

格式：-P <file>

save PID in <file>, only used with -d option

-d 参数

设置 *tcpcopy* 以 daemon 运行

5.4 TCPCopy server

可以通过 *intercept -h* 来查看 *intercept* 命令的帮助文档

-x 参数

格式：-x <passlist,>

Passlist 指通过 *intercept* 的 IP 地址列表，IP 地址之间以“,”隔开，这个参数的作用是指这些客户端 IP 地址的访问，其响应将返回给客户端，不会被 *intercept* 丢弃掉，常见于需要管理测试服务器应用的场合或者其它应用需要访问测试服务器被测应用的场合。

举例

./intercept -x 192.168.0.3,192.168.0.4

从 192.168.0.3 或者 192.168.0.4 发出的请求，其响应并不会被测试服务器 drop 掉。

-p 参数，设置监听端口，默认为 36524

需要注意的是，一旦改了这个默认端口，*tcpcopy* 端必须设置-p 参数为与之对应的端口值

-t 参数，设置 *intercept* 内部路由信息的超时大小，默认是 120s。适合应用于连接保持很久的场合

此参数只有在非 single 模式下才有效

-s 参数，系统内部 hash slots 的大小，如果应用需要大量连接，那么可以加大这个值，以提高性能，默认大小 65536

此参数只有在非 single 模式下才有效

-b 参数，后跟 IP 地址，用来设置允许监听的 IP 地址，常见于安全性比较高的场合。默认情况下，监听 IP 地址默认为*，也即这台机器的所有 IP 能够监听 36524 端口，如果不允许外网 IP 地址访问，可设置内网 IP 地址，这样就只允许内网 IP 地址监听这台测试服务器的 36524 端口。

例如

./intercept -b 192.168.0.1

只有 192.168.0.1 才能监听 36524 端口，目的地址是其它 IP 地址的访问，均会被 OS 拒绝。

-l 参数

设置错误日志文件的文件路径

-P <file>

save PID in <file>, only used with -d option

-d 参数

设置 *intercept* 以 daemon 运行

6 UDPCopy 用法参考

一般操作如下：

在在线服务器端 (需要 root 用户权限):

./udpcopy -x local_port-remote_ip:remote_port

在测试服务器端，设置 iptables 命令 drop 掉响应信息，如果 UDP 应用程序有响应的话

iptables -I OUTPUT -p udp --sport port -j QUEUE

需要注意的是，这里 port 是变量，随具体应用端口而定

7 MysqlCopy 用法参考

除了 TCPCopy 的 *configure* 命令选项外，要复制 mysql 请求，必须在 *configure* 的时候指定工作模式：

./configure -- enable-mysql ---mysql 模式

./configure -- enable-mysqlsgt ---mysql skip-grant-tables 模式

7.1 Mysql 正常模式

需要指定用户名密码对，举例如下：

./tcpcopy -x 3306-xxx.xxx.xxx.xxx:3306 -u user1@password1,user2@password2 -d

这里复制在线的 3306 端口请求，也即复制 mysql 请求到 IP 地址为 xxx.xxx.xxx.xxx 机器的 3306 端口中去，其中指定用户名 user1,密码为 password1;用户名 user2，密码 password2。用户名 user1 和 user2 在测试服务器和在线机器上面的 user 表中的信息必须一致，密码可以不一样。

需要注意的是，TCPCopy 0.6.3 以下版本针对多个用户名之间的间隔符采用了冒号“:”，具体是哪一种，可以查看 *tcpcopy -h* 命令

7.2 Mysql skip-grant-tables 模式

这种工作方式是推荐的工作方式，因为简单，无需强制要求用户的权限一样，无需提供密码，唯一要求就是要捕获到新的连接才能工作。

命令很简单：

./tcpcopy -x 3306-10.130.12.161:3306 -d

8 QA

1） 事件机制为什么不用性能更高的 epoll

答：首先 epoll 并不一定总是性能很高，其次我们处理的 fd 数量极少，因此采用 select 更适合这种应用场合，这样既简单，性能可能还更高（TODO 性能方面需要确认）

2） 为什么默认不用 libpcap 库来抓包

答：为了方便用户安装 TCPCopy，并不是每个 OS 都默认安装 libpcap 开发库的。

如果想用的话，*configure* 的时候指定*--enable-pcap* 即可

3） 为什么需要 TCPCopy server（*intercept*）

答：这是因为需要它来传递响应包头信息，这样才能完成 TCP 交互，还有默认情况下，TCPCopy server 需要解决响应包头如何返回的问题。

值得注意的是，针对 mysql 的正常模式，还需要 *intercept* 返回 greet 数据包信息。

4） 为什么 ab 发出的请求，在压力很大的场合，TCPCopy 丢失率非常高

答：TCPCopy 是基于 session 的，一个连接一个 session，由 4 元组构成，源 IP 地址，源端口，目的 IP，目的端口。在同一台机器 ab 发出的请求是基于短连接的，可变的一般只有源端口号，再加上压力大时候，QPS 往往非常高，ab 重复利用源端口的速度非常快，导致测试服务器还没有来得及处理具有同一四元组的前一个 session 的数据，后面的 session 就过来了。这样的问题比较类似于 TCP 的 timewait 问题，因此我们可以采用如下策略：

a）从多个机器发送 ab 请求到测试服务器，这样可变的不仅仅是端口号，还有 IP 地址

b）采用长连接的压力测试工具，比如 *httpsender*

5）机器之间无法复制请求

答：请确认是否是跨网段复制请求, 如果不是，请参考 QA 19；如果是，采用如下方法进行诊断。

请先在同一网段复制请求，看能不能成功。如果同一网段可以成功复制请求，那说明跨网段转发过程中遇到安全方面的问题（TCPCopy 一般是支持不同网段的请求复制的，由于不可控因素的存在，导致跨网段无法复制请求的情况比较常见），导致复制的数据包在中途被丢弃掉了。一般解决方案如下：

a） 如果条件允许，只在同网段进行流量复制

b） 先复制请求到同一网段，然后设置代理，把流量导向不同网段的服务器

6）TCPCopy 崩溃了,怎么办？

答：感谢你发现进程崩溃的情况，请设置 *ulimit -c unlimited*，再运行一遍，这时候如果崩溃，会产生 core 文件，然后 *gdb ./tcpcopy core.xxx*，进入 coredump 文件，执行 bt，打印出堆栈情况，请把此堆栈情况通过电子邮件（wangbin579@gmail.com）发送过来。

7）发现复制过去的请求比在线多，这是怎么回事？

答：如果在线服务器前面有 LVS，这很正常，如果没有 LVS，对比在线 acess.log 和测试服务器的 access.log，应该就能看出为什么多出请求了。

8）我下载了 TCPCopy，但我想使用 MysqlCopy，怎么编译？

答：从 github 上面下载的最新版本，在 configure 的时候指定工作模式，比如：

--enable-mysqlsgt	mysql skip-grant-tables mode
--enable-mysql	mysql mode

9）能不能只复制 post 请求？

答：目前 TCPCopy 还不能区分这些请求，目前可以采用的策略就是先复制请求给 web 服务器代理，在 web 代理服务器上面抽取出 post 请求，再转发给待测试的服务器。

10）基于用户 session 的请求复制过去以后都被测试服务器给拒绝了，为什么？

答：一般方法有两个：

a） 进一步对 TCPCopy 进行二次开发,可参考 mysql 方式

b） 在测试服务器的上层应用去掉校验过程，类似 mysql 采用 skip 方式跳过校验过程

c） 先把请求复制给标准的 web 服务器，在 web 服务器端进行应用层面的修改，然后再转发给后端的测试服务器。

11）MysqlCopy 复制后，没有看到一个有效请求过来？

答：由于 mysql 协议是有状态的，中途截获请求，由于无法自动填补前面的内容，导致截获的请求是没有意义的，所以 MysqlCopy 需要能够捕获 mysql 的登入过程才可以。

一般解决方案如下几种：

a）定期杀 TCP 连接

b）重启访问 mysql 的应用服务器

c）重启 mysql

12）错误日志文件在哪儿？

答：当你在/home/xxx/下运行/usr/local/bin/tcpcopy，那么 log 文件默认情况下就在/home/xxx/目录下，你也可以在执行 *tcpcopy* 或者 *intercept* 的时候设置-l 参数来指定错误日志文件的路径

13）如何复制多个端口的数据到测试服务器中去？

答：例如复制 80 和 8080 端口的请求到测试服务器中去，方法如下：

./tcpcopy -x 80-192.168.0.2:80,8080-192.168.0.2:8080

14）如何放大在线压力？

答：可以通过多种方法，一般推荐-n 参数或者-f 参数

15）如何缩小在线压力？

答：查看 tcpcopy -h ,看看有没有-r 参数，如果有，那说明此版本支持复制部分流量到测试服务器中去。

16）如何以 daemon 运行 *tcpcopy* 和 *intercept*?

答：加-d 参数即可

17）如何在测试过程中访问测试服务器上面的测试应用

答：一般可以利用-x 参数设置可通过 IP 地址列表（这些 IP 发出的请求一定要直接访问测试服务器，而不能再访问在线服务器上面的应用）或者参考老外的例子（http://globaldev.co.uk/2013/01/migrating-memcached/）

18）由于安全原因，我们不想对外暴露 36524 端口，怎么办？

答：设置-b 参数，如设置为内网 IP 地址，那么只有目的地址为内网 IP 地址才能访问测试服务器的 36524 端口。

19）请求数据包都过来了，但服务器还是没有访问记录？

答：利用 *tcpdump* 抓包，数据包都过来了，说明已经到达了测试服务器的数据链路层。如果利用 *netstat* 查看是否有应用的连接，如果没有，说明数据包在 IP 层给 drop 掉了，请查看是否有 rpfilter 设置，如果有，去掉此设置，一般即可解决此问题。

如果没有设置 rpfilter，那么确认一下 *iptables* 设置有没有冲突，如果有，请先整理好 *iptables* 相关规则。

20）源 IP 地址为 127.0.0.1（localhost）的请求能转发到其它服务器吗？

答：如果直接复制过去，一般是不可行的，但通过-c 参数，改变 127.0.0.1 地址为本机器的 IP 地址，一般就能够复制到其它机器。

21）请求过去了，但实际在线请求丢失率还是很高，怎么办？

答：请首先确认测试机器的相关系统参数是否配置合理，这里举几个常见例子：

a) 如果采用 IP Queue 的话，看看 IP Queue 模块本身是否丢包（cat /proc/net/ip_queue，具体详细内容参考 Netlink socket 接口部分），这种情况常见于压力比较大的场合

b) 由于应用程序的文件句柄默认设置太小，导致在流量大的情况下应用程序的吞吐量始终上不去

这时候为了排除系统参数设置不合理的因素，你可以复制在线流量的一部分流量到测试系统（比如复制十分之一的流量到测试系统，测试系统的 QPS 是否也是在线的十分之一左右），看看请求丢失率是否还是很高，如果压力小的时候，请求丢失率不高，那一般就是系统参数设置不合理，如果请求丢失率还是很高，请以 debug 模式运行 TCPCopy 5 分钟，把错误日志文件通过电子邮件（wangbin579@gmail.com）发送给我，最好还能附加上其它有用的信息，原则上信息越多越好。

22）市面上已经有了 *tcpdump* 和 Tcpreplay，为什么重复造轮子？

答：*tcpdump* 是用来抓包的，Tcpreplay 是用来重放的，仅仅在 TCP 层以下进行重放，连 TCP 协议都未能通过，所以重放这个说法不准确，容易误人子弟。TCPCopy 重放可以做到实时重放，也可以离线重放，这是针对大部分协议的重放，不仅仅只在 TCP 以下。

23）如何测试防火墙等网络设备？

答：把在线请求复制给代理，代理再把流量导向目的地，防火墙后面需要配置相应的应用，因为 TCPCopy 是针对 server 的流量复制。

24）TCPCopy 采用什么开源协议？

答：BSD 协议

25）对线上的应用，会有影响吗？

答：只要测试系统独立于在线系统，业务上就不会有影响，另外，在消耗系统资源方面（带宽，内存，cpu），一般影响几乎可以忽略不计

26）如何离线回放？

答：*configure* 的时候加上*--enable-offline* 参数（这种模式不能工作在实时复制）；离线回放还需要安装 pcap 库和 pcap 开发库（需要用到 pcap 库的头文件），如果系统还没有安装的话；另外运行的时候需要指定-i 参数。

举例：

./tcpcopy -x 110-xxx.xxx.xxx.148:110 -i ./110.pcap

这里 110.pcap（利用类似于 *tcpdump* 的工具来抓数据包，存放到 pcap 格式的文件中去）文件作为数据源，转发这里面的 pop3 请求到测试服务器 148 上面的 pop3 服务中去

27）如何充分利用 *intercept -x* 参数，达到多个架构层次的同时测试

答：假设在线是 nginx→memcached 应用，在线 nginx 有 10 台机器，memcached 有 3 台机器，测试服务器只有一台，测试服务器部署了整个应用(测试服务器上面的 nginx 配置是采用 127.0.0.1 访问本地 memcached 应用)。

测试的目的是复制在线的 nginx 两台机器到测试服务器，同时还想复制 memcached 机器上面的所有请求到测试服务器上去，如何做到这一点？

我们在测试服务器设置 iptables 如下：

iptables -I OUTPUT -p tcp --sport 11211 -j QUEUE

iptables -I OUTPUT -p tcp --sport 80 -j QUEUE

运行*./intercept -x 127.0.0.1 -d*

这样配置后，除了 127.0.0.1 发起的请求不会被 *intercept* 丢弃掉外，其它所有请求都会被 drop 掉。

在线服务器运行 TCPCopy client：

复制所有 memcached 流量，在所有 memcached 机器运行如下命令：

./tcpcopy -x 11211-测试服务器 IP 地址:11211 -d

复制两台在线 nginx 流量，在两台在线服务器运行如下命令：

./tcpcopy -x 80-测试服务器 IP 地址:80 -d

这样就达到了整个系统复杂的测试。

28）如果复制请求过去后，中途源 IP 地址被修改了，还能工作吗？

答：这种场合，TCPCopy 无法正常工作，原因如下：

比如复制内网请求到外网中去，由于抓的在线包的源 IP 地址是内网地址，发送的路由信息（*tcpcopy* 发送给 *intercept* 的路由信息）也是内网地址的信息，访问测试服务器的源 IP 地址是外网地址，这样 IP 地址就不一致，当 TCP 返回第二次握手信息给测试服务器 IP 层的时候，*intercept* 就会去找路由信息，由于测试服务器 TCP 协议返回的数据包的目的 IP 地址是外网地址，利用外网地址去找如何返回的信息，是找不到的，因此第二次握手数据包无法返回给 *tcpcopy*，从而导致会话无法继续下去。

如果大家遇到此问题，可以采用离线版本，就可以避开源 IP 地址修改的问题，一般就没有问题了。

29）intercept 报 cannot set mode:: Connection refused

答：一般是因为 IP Queue *模块*没有启动导致的,还有可能是 IP Queue 模块已经被破坏掉了

30）错误日志文件报大量 Message too long，怎么办？

答：请使用 0.5 或者以上版本

31）出现大量 warn 级别的日志 *unsend:too many packets*，是什么意思？

答：原因可能有很多，候选原因如下：

a）测试服务器处理速度慢，导致测试服务器积累的数据包越来越多。

这种情况常见于传统压力测试工具发出的请求

b）测试服务器对某些请求无响应，导致缓存的数据包的数目迅速上升

c）系统参数设置不合理，比如没有设置好 IP Queue 系统参数，测试服务器传递给 *intercept* 的过程中就丢失了很多响应数据包，导致会话无法继续下去，缓存的数据包越积越多

d）……

首先查看 IP Queue 系统参数有没有问题，如果没有问题，你可以尝试在压力小的时候，请求丢失率是否还是很高，如果压力小的时候丢失率不高，那么尝试修改此 *tcpcopy* 的参数-t，默认是 60s，增大此值可以增大 session 不被提前 kill 的机会；如果压力小的时候丢失率还是很高，*./configure --enable-debug*，运行 5 分钟，把错误日志文件发送给我，我来查明具体可能原因。

32）为什么离线版本不能实时转发流量？

答：考虑到性能方面，不同版本代码不一样，分开编译，性能会更高；考虑到可使用性，由于离线版本依赖于 libpcap 库，而在线版本不需要这个库，所以离线版本和在

线版本分开编译。

33）利用 *tcpdump* 抓包，为什么推荐只抓请求包？

由于离线版本 TCPCopy 只需要请求包，不需要响应包，如果全抓会影响 *tcpcopy* 的处理速度，而且还浪费 pcap 文件所占磁盘空间的大小。

34）如何保证 github 下载的版本可用？

修改代码后，我们会进行回归测试，只有通过了回归测试，我们才会更新到 github master 版本

35）TCPCopy 只能运行在 Linux 下面吗？

目前 TCPCopy 只能运行在 linux

36）TCPCopy 只能 root 运行吗？

答：需要 root 权限，由于 raw socket 和与内核打交道的函数需要 root 权限才能运行

37）*intercept* 能不能在同一台机器运行多份？

答：经我们测试，对于 *IP Queue 模块*，不行，而且对于 *IP Queue 模块*也没有必要运行多份，设置多条 *iptables* 命令即可

38）先关闭 *tcpcopy*,还是先关闭 *intercept*?

答：一般推荐先关闭 *tcpcopy*，再关闭 *intercept*，但如果是分布式测试，且想快速关闭整个测试过程，那么推荐先关闭 *intercept*。

39）怎么加入到这个项目？

答：首先很欢迎加入这个项目，目前代码都在 github；其次任何修改，哪怕是英文注释修改，我们都是非常欢迎的。

目前参与人员包括淘宝的 skoo87 等人员，我们希望队伍越来越大，共同为开源作出一份贡献。

40）如何复制 pop3 请求？

答：跟一般复制差别不是很大，但还是需注意以下内容：

a）对于在线和测试服务器，pop3 用户名和密码必须一致

b）长连接场合下，请求复制效果可能不是很理想，因为针对此类型的应用，还没有类似于 mysql 的自动补充登入的过程。

41）离线回放，错误日志文件报大量 truncated packets warning，是怎么回事？

答：由于抓包过程中并没有把请求内容抓全，导致了此问题。

建议利用 *tcpdump* 抓包的时候，指定-s 大小为 65535 或者-s 0

42）TCPCopy client 需要 *IP Queue 模块*支持吗？

答：不需要

43）*ip_conntrack: table full, dropping packet*，这是怎么回事？

答：在在线服务器内，由于 ip_conntrack 是在 IP 层中做的，而 tcpcopy 转发数据包，会走部分 IP 函数，导致了 ip_conntrack 瞎推测连接状态。

为了解决此问题，可以采用如下三种方案：

a）去除 ip_conntrack 内核模块

modprobe -r ip_conntrack

b）设置对 *tcpcopy* 转发的包不进行 track

如果需要对在线机器进行操作：

iptables -t raw -A OUTPUT -p tcp --dport 转发目的端口 -j NOTRACK

如果需要针对测试服务器，那么可以设置如下：

iptables -t raw -A PREROUTING -p tcp --dport 转发目的端口 -j NOTRACK

c）加大 ip_conntrack_max 值

echo "262144" > /proc/sys/net/ipv4/ip_conntrack_max

44）*sudo iptables -I OUTPUT -p tcp --sport 36524 -j QUEUE*，这样有什么问题？

由于 intercept 默认采用 36524 端口进行监听，所以设置上述命令，会导致 tcpcopy client 和 intercept 无法建立连接，从而无法继续工作下去。

如果应用是 36524 端口，那么你需要改变 intercept 监听的端口，可以采用-p 参数来改变监听端口。

45）离线版本，编译出现找不到 pcap.h 文件，怎么办？

需要安装 pcap 的 dev 库，比如 centos 机器上：

yum install -y libpcap-devl

46）为什么加了‘&’，tcpcopy 还经常退出？

仅仅‘&’是不够的，还需要加 nohup，不过建议采用加-d 参数的方式

47）每天运行到某一个时刻，tcpcopy 就无法把请求复制给测试系统了，为啥？

答：请检测一下 iptables -L，看看相应的设置是否已经被定期清理掉了

48）遇到问题怎么办？

访问 https://github.com/wangbin579/tcpcopy/issues，看看有没有类似的问题。

如果没有，最好*./configure --enable-debug* 运行，输出 5 分钟的 log（如果压力大，可以在 *tcpcopy* 命令端加上-r 参数，复制部分在线流量到测试服务器中去），发送过来，当然再加上辅助信息更好，原则是给的信息越多越好。

49）Paper 何时放出来？

答：需要等中了论文才能放出来

50）按照 README 文件来操作，命令都一样，为啥不见请求过来？

答：完全按照例子中的命令，一般是不行的，因为不同的应用有不同的端口，所以相应的命令也需要做相应的修改，请理解好 TCPCopy 原理后再试验。

51）遇到问题后，该发哪些信息？

答：俗话说，不幸的家庭各有个的不幸，仅仅告诉啥问题，一般是很难推测出原因的，所以希望大家尽可能地提供详细的信息，方便问题诊断。

52）如何查看请求复制的效果？

答：一般可以通过这些值来分析出复制的效果：

a) 在线请求数据包方面：

syn cnt:xxx	抓到了多少在线 syn 数据包
all clt packs:xxx	抓到了多少在线数据包
clt cont:xxx	抓到了多少带有 payload 的在线数据包

b) 复制转发数据包方面：

send Packets:xxx	发送了多少数据包给测试服务器
send content packets:xxx	发送了多少带有 payload 的数据包给测试服务器

c) 响应方面：

conns:xxx	已经成功了建立了多少连接
resp packs:xxx	收到了多少带有 payload 的响应包
c-resp packs:xxx	收到了多少响应包

52）出现大量 send to:Inappropriate ioctl，怎么回事?

答：如果测试机器内核系统参数设置合理的话，一般是在线机器压力比较大的时候，才会出现这类问题，请设置在线机器的相关的系统参数。

53）出现 Set netlink socket(5) mode failed 或者 can't set mode,check if ip queue is up:Illegal seek，怎么回事？

答：请查看 IP Queue 模块是否已经启动

54）intercept 报 nl recvfrom (No buffer space available)，怎么办？

答：加大设置 intercept 所在的机器的网络内核 buffer，比如：

sysctl -w net.core.rmem_max=16777216

sysctl -w net.core.wmem_max=16777216

55）测试机器全都是 SYN_RECV 状态，怎么办？

答：*iptables* 针对端口的设置是否正确。如果没有问题，可能原因如下：

1）IP Queue 模块虽然加载了，但由于 NFQueue 的干扰，导致 IP Queue 模块工作不正常，这时候要么只用 NFQueue,要么卸载 NFQueue 相关内核模块，再重启 IP Queue 内核模块（具体参考：https://github.com/wangbin579/tcpcopy/issues/60）

2）中途设备只让第一次握手数据包过去，而不让后续的第三次握手数据包过去

56）TCPCopy 能修改应用层的数据（比如 url 参数）吗？

答：一般需要二次开发，当然最简单的方式就是先用 TCPCopy 转发请求给应用服务器（比如 http 请求，可以采用 web 服务器），在应用服务器上面进行修改，再转发给测试服务器。

57）通过查看 tcpcopy 的日志，看到 c-resp packs 的数值不为 0，怎么上层应用服务器没有看到访问日志？

答：一般是因为访问量比较小，而且是刚运行的时候，操作系统还没有把访问日志输出到日志文件，所以还不能看到。

58）在同一网段，复制请求给虚拟机，在测试服务器抓不到任何复制过来的数据包，而在在线服务器能抓到，怎么回事？

答：这个应该是虚拟机把复制的包给丢弃掉了，在同一网段的虚拟机，相当于跨网段的情况。

59）sh autogen.sh 运行错误，怎么办？

答：参考 https://github.com/wangbin579/tcpcopy/issues/63

60）复制在线请求，在测试服务器只能看到内网 IP 地址（或者可信任的 IP 地址）过来的请求过来，怎么回事？

答：据统计，目前查询到的原因有如下两个：

1）可能你的机器没有外网 IP 地址，导致 IP 模块在流量进入的时候对外网 IP 地址的数据包进行了过滤。这时候给机器分配外网 IP 地址即可解决。

2）网卡 IP 地址设置错误，比如用 ifconfig 针对网卡设置了错误的 IP 地址

61）除了在 intercept 设置-x 参数外，还有哪些手段能够达到访问测试服务器的目的？

答：如果用户熟悉 iptables 设置，可以通过 iptables 设置命令达到类似的效果

举例如下：

http://globaldev.co.uk/2013/01/migrating-memcached/

100）Why not just go through normal tcp stack to maintain connections between production server and test server ? That would simplify the code dramatically, would scarifies some performance though.

Answer：

If it goes through normal tcp stack to maintain connections between production server and test server, external requests would be transferred to internal requests. The characteristics of external network, such as latency, would be lost. Moreover, it will affect the online machines, as using normal tcp stack would occupy system resources, such as ports.

9 总结

TCPCopy 的使命有如下几个方面：

a) 让每个潜在的 IT 程序员能够成为一名优秀的架构师，不再仅仅依赖经验做事情
b) 尽可能使上线过程轻松，bug free
c) 为开源做贡献
d) 改变上线前测试的手段

总体来说，如果你对上线没有信心，如果你的单元测试不够充分，如果你对新系统不够有把握，如果你对未来的请求压力无法预测，如果你想对比诸如 apache 和 nginx 的性能，如果你想放大在线流量，如果你想 profile 在线应用，如果你想成为出色的架构师， TCPCopy 可以帮助你解决上述难题。