|  |  |  |  |
| --- | --- | --- | --- |
| 产品名称 | TCPGO:TCP会话欺骗工具 | 产品版本 | 1.0 |
| 文档编号 |  | 密 级 | 公开 |

文 档 控 制

|  |  |  |  |
| --- | --- | --- | --- |
| **修改历史** | | | |
| **修改人** | **修改日期** | **修改版本** | **修改描述** |
| kamuszhou | 2013-12-31 | V1.0 | 初稿. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **关键里程碑** | | |
| **里程碑** | **日期** | 负责人 |
| 发布 | 2013-12-31 | kamuszhou |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **审核记录** | | | |
| **审核人** | **审核日期** | **角色** | **说明** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

TCPGO：TCP会话欺骗工具的

实现与应用

TCPGO: Implementation and

Application of The TCP Session Cheat Tool

[Kamuszhou@tencent.com](mailto:Kamuszhou@tencent.com)

January. 3. 2014.

**目 录**

[1 介绍 1](#_Toc376518593)

[1.1 TCPGO是什么，能做什么 1](#_Toc376518594)

[1.2 核心功能简介 1](#_Toc376518595)

[1.3 开发背景和意义 4](#_Toc376518596)

[1.4 TCPGO的TCPCOPY的主要区别 4](#_Toc376518597)

[2 配置运行TCPGO 6](#_Toc376518598)

[2.1 如何编译 6](#_Toc376518599)

[2.1 如何运行 6](#_Toc376518600)

[2.1 TCPGO的命令行选项 8](#_Toc376518601)

[3 TCP欺骗的理论基础 9](#_Toc376518602)

[4 设计与实现 11](#_Toc376518603)

[2.1 TCPGO的设计图 11](#_Toc376518604)

[2.2 TCPGO的I/O模型 12](#_Toc376518605)

[2.3 TCPGO的核心代码 13](#_Toc376518606)

[5 TCPGO的应用场景举例 15](#_Toc376518607)

[5. 1 为服务器开发工程师搭建随写随调的IDE开发环境 15](#_Toc376518608)

[5.2 为测试工程师提供一个新的自动化测试工具 17](#_Toc376518609)

[6 实战 17](#_Toc376518610)

[7 工程经验 17](#_Toc376518611)

[致 谢 18](#_Toc376518612)

**摘 要**

TCPGO是一个基于TCP欺骗技术的伪造TCP会话的工具。它可以以实时采集的TCP流量为素材，也可以以保存下来的TCP流量包文件为素材甚至定制素材，创建虚假的TCP会话，让服务器误以为在与真实的用户通讯。

本文讲述了TCPGO的使用，原理，和设计，并和开源的类似工具做了比较。

**关键词：TCP欺骗，流量复制，流量回放，TCPGO**

# 1 介绍

## TCPGO是什么，能做什么

TCPGO是一个基于TCP欺骗技术的伪造TCP会话的工具。内部开发时的工程项目名为Horos，是占星术Horoscope的缩写，喻意使用本工具可以占卜未来，多么神秘兼有贵族气质的名字。

TCPGO大概能做以下几件事情：

1. 能够把线上机的真实用户的TCP请求实时复制到测试环境的机器上，使测试环境的TCP服务器误以为和真实的用户通信，观察测试服务器的运行情况。使得TCP服务器在正式灰度前暴露出更多的问题。
2. 先把真实用户的TCP请求保存在PCAP抓包文件中，TCPGO可以以这些抓包文件为素材，把流量重放给服务器。视频广告的投单算法评估系统将用这个思路开发，该项目还在推进之中。
3. TCPGO也是一个TCP压力测试工具。它可以设定并发的用户连接数，模拟少量用户并发，测试TCP服务器的性能。
4. TCPGO还可以是一个自动化功能测试工具。它可以通过定制TCP请求，定制期望的测试器回复来实现TCP级别的自动化功能测试。计划在此基础上建立视频广告开发组的回归测试用例库。(未实现)
5. TCPGO为能服务器开发工程师搭建一个开发用模拟环境，让码农们可以像写桌面应用一样边写边调。更多叙述见第五章。

## 核心功能简介

TCPGO从线上机上复制用户的真实请求，并把这些请求实时播放给测试用服务器；或者把请求保存下来，以后再离线播放给服务器。

如图1.1所示，客户机发给线上服务器的三个请求：1.询问营业时间2.询问菜单3.询问停车位信息。



图1.1

这三个请求被TCPGO记录下再重放给测试服务器，如图1.2：



图1.2

对于不依赖会话上下文的请求，TCPGO把这些请求回放给测试用服务器，给测试服务器提供了一个接近在线实际情况的运行环境。对于TCP一问一答就立即关闭的短连接，可以被认为是请求不依赖上下文的特例。

缺点则是，如果请求依赖上下文，生产服务器和测试服务器的回复有逻辑上的较大区别。那么，回放线上机上录制的客户请求给测试服务器，会让测试服务器发觉并不是和真实的用户会话。

## 开发背景和意义

TCPCOPY是类似的一个开源的TCP流量复制工具。在使用TCPCOPY，以及在TCPCOPY的基础上进行二次开发的过程中，遇到了很多问题。最主要的问题是调试时很难追踪，调试效率很低。TCPCOPY的代码的最核心部分：TCP状态机的实现并不严格遵守TCP的工业标准。TCP标准指定的11个标准状态很难和TCPCOPY的代码对应起来。在和TCPCOPY的作者的沟通中，我提出按照TCP的11个标准状态实现代码，以方便理解和调试。作者认为这样做使得代码太复杂，而我觉得按照标准做事是让代码变得简单。截止本文档撰写的时间2013年12月31号，TCPCOPY的最新版本0.9.6的实现中，最核心的TCP状态机部分，仍然晦涩难懂。

12月初准备在TCPCOPY的基础上二次开发个性定制的TCP流量回放工具时，再一次迷失在TCPCOPY的TCP会话状态的代码中。与其再投入时间去理解这些复杂的代码，不如按照自己的理解，按照TCP标准的11个状态写一个小巧的TCP流量复制和回放工具。

在研发层面上， TCPGO的研发成功使得在TCP流量复制和重放相关的领域继续二次开发，调试定位错误变得得心应手。

在使用部署层面上，TCPGO更易于安装和部署。而且，当都工作在实时流量复制状态时，TCPGO对线上机的性能影响要比TCPCOPY小。

## TCPGO的TCPCOPY的主要区别

正如上一小节已经有所提及，TCPGO和TCPCOPY的主要区别来自代码级别和使用部署级别。

**代码级别的区别：**

TCPGO遵守工业标准，易理解，易调试。TCPGO的代码质量明显高过TCPCOPY，表现在耦合度小，函数命名易理解，注释准确充分。更具体的，在最核心的代码部分，对TCP状态机的实现，TCPGO（当前版本v0.4）的11个标准状态对应11个成员函数，类成员变量中记载了当前状态和滑动窗口的大小，滑动窗口的内容，一目了然。只要对TCP有基本了解的工程师，都能很快读懂TCP会话状态的代码。开发者调试时设置断点，分析错误也就井井有条。

而写作本文时最新TCPCOPY 0.9.6对TCP会话的实现代码 <https://github.com/wangbin579/tcpcopy/blob/0.9.6/src/tcpcopy/tc_session.h> 和 https://github.com/wangbin579/tcpcopy/blob/0.9.6/src/tcpcopy/tc\_session.c

假如调试者要定位TCP主动关闭时的逻辑，先在头文件中找到了fin\_sent\_cnt这个变量涉及发送FIN包，于是在.c实现文件中搜索fin\_sent\_cnt，定位到函数wrap\_send\_ip\_packet，结果所有的TCP状态都耦合在这个函数中，除非调试者理解所有的代码，否则很难追踪下去。再看TCPGO的代码，如果TCP主动关闭，必经标准状态FIN\_WAIT\_1，在函数fin\_wait\_1\_state\_handler中下断点即可。FIN\_WAIT\_1状态又根据收发包情况转至另三个状态，与TCP状态转换图表达的逻辑和代码一一对应，逻辑清楚。

TCPGO对TCPCOPY的代码质量优势还来自于：

1. TCPGO的代码没有TCPCOPY随处可见的各种预编译条件宏搞乱代码的主要逻辑。

2. TCPGO的使用部署方式设计得比TCPCOPY简洁，使得代码量减少。

3. TCPGO的基础数据结构使用工业标准库std::list，std::map，节省了代码量。而TCPCOPY重造轮子，用了自己写的hash\_map和list，增加了调试者和二次开发的编码和理解难度。

4. 函数的命名比TCPCOPY表达意思要清晰，有效代码注释比TCPCOPY多。

**使用和部署级别的区别：**

TCPCOPY需要运行两个程序，一个用于抓取和传送流量包，另一个用于截获测试服务器的回复报文再和另一个程序通讯。而TCPGO只需要一个程序，充分运用标准Unix工具，直接用tcpdump抓取并提供流量，体现了UNIX的KISS设计思想。

TCPGO的部署方式使出错源从两个降到一个，在线上机方便定位错误。如果工作在实时流量复制模式，TCPCOPY在线上机上需要与生产服务器争抢资源，来运行模拟TCP会话的代码，还需要资源与另一个组件通讯；TCPGO只用在线上机上运行两个标准的Unix工具：tcpdump和netcat，对生产服务器的性能影响小。

**TCPCOPY相对于TCPGO的优势：**

Github上的提交记录显示TCPCOPY第一次提交是在2011年10月14号，历经两年多的演化，TCPCOPY能更好的支持多变的网络环境。TCPCOPY历经的实战考验非常充分。而TCPGO目前只在自建的虚拟机网络环境中达到设计预期，缺少实战的考验。

TCPGO还没有非常详细的日志记录。

# 2 配置运行TCPGO

## 2.1 如何编译

视频广告的代码地址：http://sh-svn.tencent.com/adpf/adpf\_iadshummer\_rep/app\_iads\_qqlive\_proj/trunk/code/horoscope

或者我本机上的源代码包，<http://10.30.116.81/E%3A/projs/horoscope.tar.bz2> 我的机器开机时才能连接到。-\_-

编译只需要一步： # make

在bins/目录下生成 tcpgo，还有个horos，两个文件是一样的，tcpgo是horos的硬链接，horos是tcpgo最初的名字。

## 2.2 如何运行

运行可以用单机模式或者双机模式。单机模式指的是TCPGO运行在和测试服务器相同的机器上，双机模式则运行在不同的机器上。单机模式方便一些，但如果不希望TCPGO与测试服务器争夺系统资源，则使用双机模式。

**双机模式：**

下面先介绍双机模式。假如测试用TCP服务器的IP地址是 192.168.1.100，开放端口号80。TCPGO将运行在 192.168.1.200。

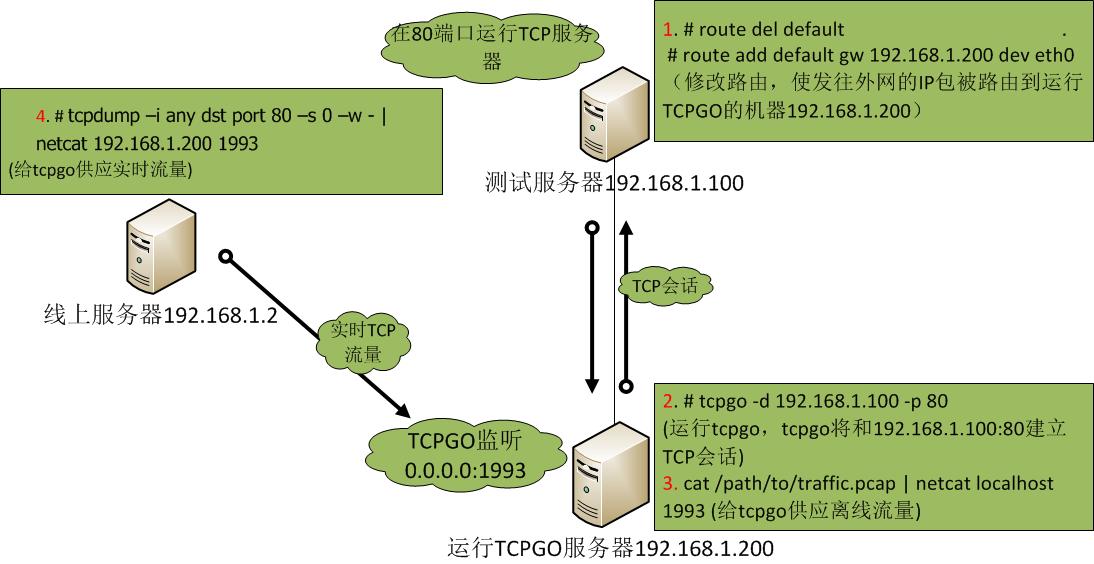


图2.1 TCPGO的双机模式部署示意图

**Step 1**: 修改路由。删除测试用服务器所在机器的默认路由。如：

# route del default.

再加一条默认路由，使得发往外网的IP被路由到将运行TCPGO的机器。设置网卡为正确的出包网卡。如：

# route add default gw 192.168.1.200 dev eth0

**Step 2**: 在机器192.168.1.200上运行tcpgo。如：

# tcpgo –d 192.168.1.100 –p 80

此时TCPGO在机器192.168.1.200上开放了TCP端口号1993，通过该端口号给TCPGO供给流量。

**Step 3**: 假如流量已经被存放在pcap包中了，路径为/path/to/traffic.pcap。那么运行下面的命令给TCPGO供给流量。

# cat /path/to/traffic.pcap | netcat 192.168.1.200 1993

这意味着，实际上可以从任意一台可以TCP连接到192.168.1.200的1993端口的机器上给TCPGO供给流量。

**Step 4**：假如要实时地把线上机上的实时流量供给TCPGO呢。假如线上机上的生产服务器开放的端口也是80，如下给TCPGO供应流量：

# tcpdump –i any dst port 80 –s 0 –w - | netcat 192.168.1.200 1993

注意红色的-s 0，这指定tcpdump抓下整个IP包，不要截断它。如果发送给TCPGO的IP 包是被截断的，TCPGO将不能工作。

步骤3和步骤4的两种供给流量的方式可以并行工作。实际上，可以对每个网卡各起动一个tcpdump抓取流量，比如-i eth0 ，而不是 –i any。技术上的差异是，对N个网卡各起动一个tcpdump，则创建了N个tcp连接到TCPGO，而-i any的方式仅有一条TCP连接。虽然没有测试过，但我认为两种方式没有太大的效果差异。

至此，流量供给方式也可以方便的扩展到同时把多台服务器的流量供给TCPGO。仅仅需要在不同的在线服务器运行tcpdump而已。

**单机模式：**

单机模式较之双机模式的部署区别仅仅是把tcpgo和测试服务器运行在同一台机器而已。

仍需注意：设置好测试服务器的路由，把默认路由指向同网段的任意一台非路由器的机器即可。这一步骤至关重要，原理可参见：理论和技术基础章的TCP欺骗。

TCPGO实际上还能在命令行上指定TCPGO刚运行时即从pcap抓包文件中读取流量。见TCPGO的命令行选项一节的-f 选项的解释。

## 2.3 TCPGO的命令行选项

-f –-pcapfile file\_path:

指定TCPGO加载pcap流量文件file\_path。

-d –-dst-addr xxx.xxx.xxx.xxx：

指定流量被发往IP xxx.xxx.xxx.xxx。

-p –dst-port nnnn：

指定流量被发往端口号 nnnn。

-c –concurrency-limit n：

指定最大并发连接数为n。

-h –help:

简单的帮助

-v –version:

版本信息。

# 3 TCP欺骗的理论基础

TCP协议是一个点对点的，有状态的，设计初期非常简单的传输层协议。TCP本身没有现代密钥体系的身份验证，并不保证通信的对方是可信的。它区分每一个会话的唯一标志被称为SOCKET PAIR，它由四个字段构成：源IP地址，源端口，目标IP地址，目标端口。而IP协议也不校验源地址真伪。所以，理论上只要可以伪造SOCKET PAIR的四个字段，便可以入侵TCP会话。当然可以在IP层用IPSec做IP地址的可靠性验证，那就不能轻易入侵TCP会话了，这个不在本文讨论范围之列。

这里也不讨论如何入侵已经建立的TCP会话，只讨论如何欺骗TCP连接方新建一个假的TCP会话，让TCP连接误以为在和某个（IP，PORT）通讯。

假如我要假冒IP 20.20.20.20，端口号2000与服务器30.30.30.30,端口号80通讯。于是，伪造一个SYN包，源地址端口号填上20.20.20.20和2000，目标地址填上30.30.30.30和80，然后把包发出去。服务器30.30.30.30,端口80便收到了SYN包，服务器回SYN+ACK给20.20.20.20，端口号2000。SYN+ACK翻山越水经过各种路由器，终于到达了真实的20.20.20.20。如果20.20.20.20真实在线的话，绝大多数情况下，20.20.20.20认为它没有向30.30.30.30发送过SYN，于是果断的回了一个RESET。如果20.20.20.20不在线，最后一跳路由可能会回一个DESTINATION UNREACHABLE的ICMP包给30.30.30.30。结果都是30.30.30.30立即撤消了这个TCP会话的创建，反应在应用层，则是accept()系统调用返回-1，设置相应的errno。如下图所示：

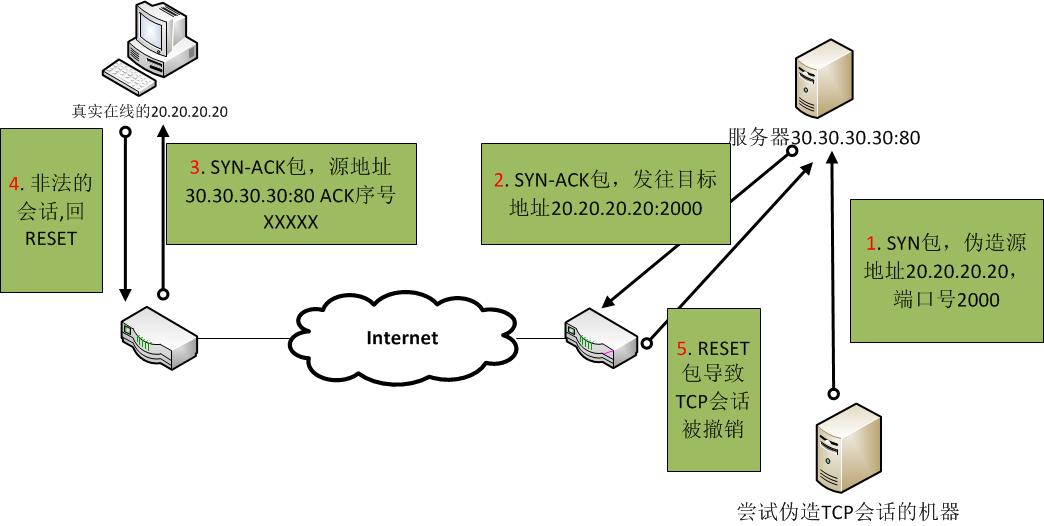


图3.1 一次失败的TCP欺骗示例

这个问题如何解决呢，不让SYN-ACK(其它包也一样)发向20.20.20.20就解决了。这就是前文叙述如何使用TCPGO时要设置发往外网的路由要改成一台没有路由功能的机器的原因。这样，SYN-ACK被发到了一台没路由功能的机器，这台机器不是路由器，所以会把SYN-ACK丢弃，于是SYN-ACK到不了20.20.20.20，自然就收不到撤消会话的REST包或DESTINATION UNREACHABLE的ICMP包。如图3.2：

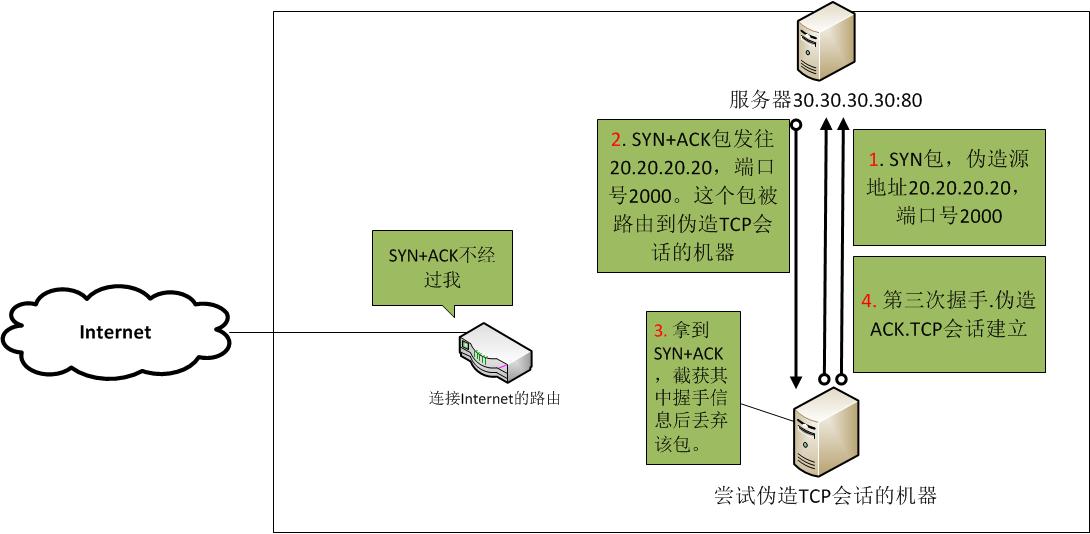


图3.2 成功建立TCP会话

当然，还有其它的方法解决这个问题，比如使用IPQueue或者NFQueue内核模块。这两个内核模块用来在应用层编写防火墙，可以通知内核如何裁决每个包。因为公司有些机器上没有IPQueue或NFQueue模块，而且这两个模块的编程接口很容易出错，各种宏加各种指针偏移，相对设置路由的实现方式而言没太大的优势，因此TCPGO里面使用的是设置路由的解决方案。开源工程TCPCOPY中实现了使用IPQueue或NFQueue的技术方案。

另外，TCP通过给字节流编序号，接收方ACK收到的序号，发送方重发没有被ACK的负载也是每个程序员公知的协议特性。如果能截获发送方发送的TCP报文，解决ACK的问题自然不在话下了。如何截获服务器发送的报文呢，上文提到的IPQueue和NFQueue就可以。另外，广泛使用的libpcap库和AF\_PACKET协议族，SOCK\_RAW原生套接字类型都是现成的解决方案。TCPGO使用了创建链路层的套接字抓取过往链路层的IP包来实现。

至此，解决了两个主要问题：1.不让拥有真实IP机器或互联网上的路由器影响假TCP会话的建立。2.发送伪造的ACK让发送方认为对方已经成功接收。编写TCP欺骗工具在理论上已经没有障碍了。

# 4 设计与实现

## 4.1 TCPGO的设计图

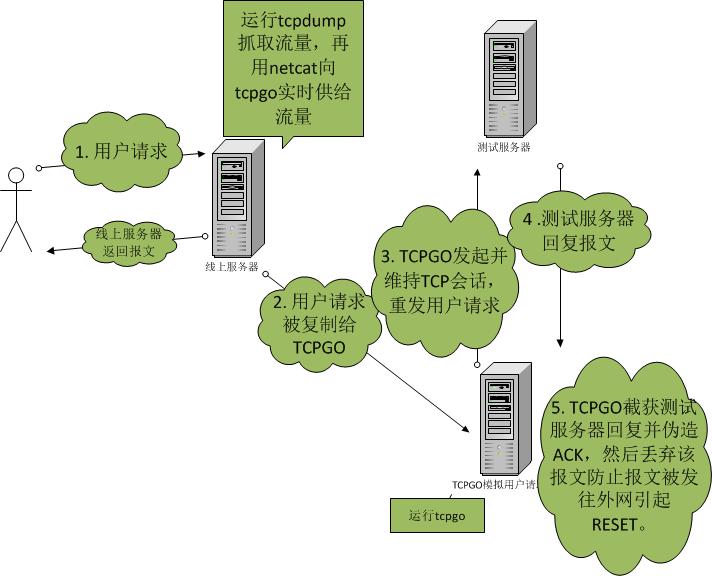


图4.1 TCPGO的设计示意图

TCPGO在设计上相对于TCPGOPY的创新点在于：TCPCOPY在线上服务器端模拟用户，需要和线上服务器争抢较多CPU资源和内存资源。而TCPGO只需要在线上服务器端运行标准Unix工具tcpdump和netcat，对线上服务器的影响降至最少，也更加安全。如下图是TCPCOPY的设计：

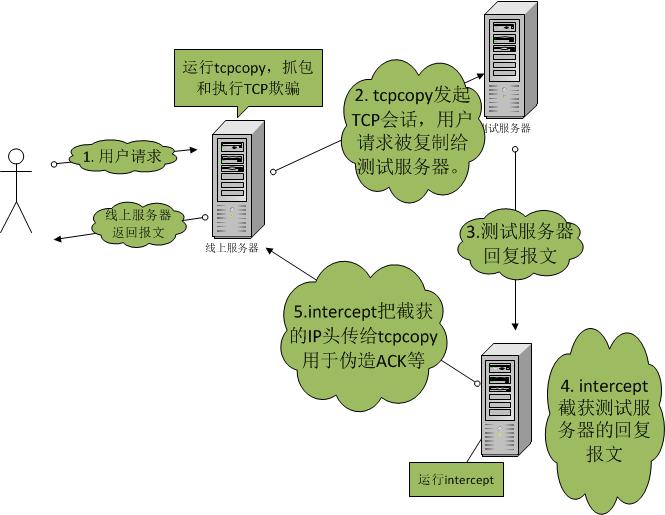


图4.2 tcpcoy的设计示意图

## 4.2 TCPGO的I/O模型

TCPGO的I/O模型目前使用的是常见的单线程，多路传输事件驱动I/O模型。该模型在编程复杂度和运行效率之间取得了一个很好的平衡。虽然该模型天生不能充分利用多核CPU的计算力，但考虑到TCPGO是一个I/O密集型的工具，所以现有的I/O模型是胜任应用场景的。

## 4.3 TCPGO的核心代码

TCPGO在用户空间实现了TCP协议的一个最小集，包括：三次握手，四次挥手，简化后的滑动窗口，重传机制，确认机制。TCPGO对于拥塞避免，只做了简单地控制，因为在内网，网络环境比较简单一般都有足够多的带宽。对于TCP的慢启动，快重传快恢复，RTT的计算，Nagle算法都没有加入。

TCPGO中的类class tcpsession是TCPGO实现TCP协议的部分，它按照TCP的11个标准状态实现了一个简单可用的运行在用户空间的TCP协议：

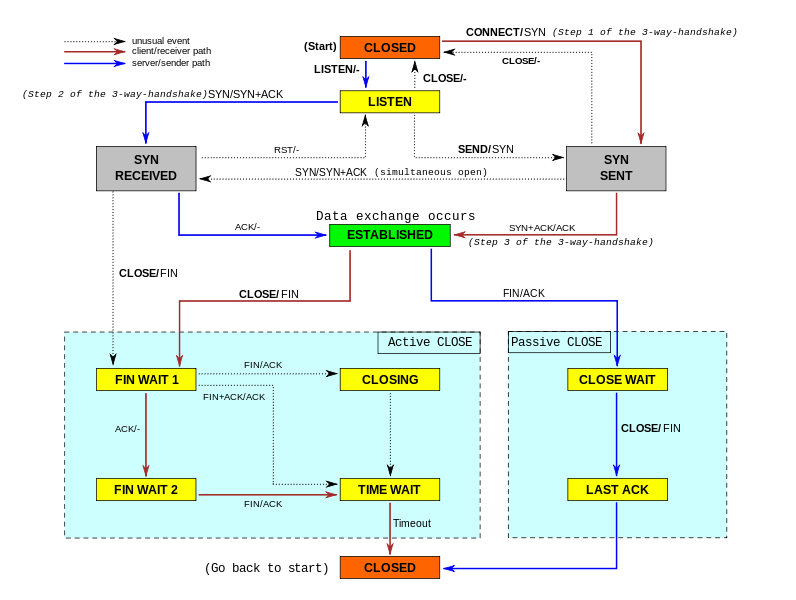


图4.3 TCP的11个标准状态转换图

其最核心的数据结构如下：



图4.4 TCPGO代码中实现TCP状态机的主要数据结构

每个状态对应一个成员函数：



图4.5 TCPGO实现的TCP状态机代码示例

# TCPGO的应用场景举例

## 5. 1 为服务器开发工程师搭建随写随调的IDE开发环境

服务器开发中的难点之一就是开发环境和实际运行环境大不相同。服务器代码的开发环境中没有真实的用户请求，因此在服务器代码的编写过程中不能够即时校验代码的逻辑，很多简单的，本可以在编码被发现的问题，直到交付给测试才暴露出来，甚至会被掩盖到上线后。另外，对于大部分不是vim，emacs大神的凡人开发工程师，需要一款IDE来提高编码和调试的效率。

结合TCPGO的使用，可以有三个解决方案来搭建一个有“真实”用户请求的，随写随便的IDE环境。

**解决方案1：**

这是一个远程调试的方案。主要步骤有：

1.准备一台Linux机器，运行TCPGO，用离线流量包或者实时流量复制的方式给这台Linux提供“真实”的用户连接。

2.在Linux机器上安装支持远程调试的gdbserver。

3.在windows机器上用WinSCP自动与Linux机器同步源代码。

4.安装windows版的eclipse + CDT再加用于在windows上运行远程调试linux机器的GDB(我已有已经编译好的二进制版本)，并配置好eclipse的远程调试功能。

如此，便可以在windows上随写随调有“真实”用户请求的服务器代码。缺点是，如果要eclipse显示STL的容器中的值，需要在windows上用MinGW或Cygwin编译一个支持python插件的交叉调试用gdb。而我尝试用MinGW或Cygwin编译python库时均未成功。所以对这个方案，目前我的探索未能让eclipse显示STL容器的内容，但可以方便的设置断点。

**解决方案2:**

这是一个使用虚拟机技术的解决方案，需要在windows机器上安装vmware或virtual box虚拟机，再在虚拟机上安装Linux，再把代码部署到Linux机器上，用TCPGO提供用户连接。因为Linux安装在自己的虚拟机中，软件配置由自己掌控，也有现成的eclipse, gdb软件包。我用的Debian发行版从官方软件仓库中安装的gdb原生支持python插件，gdb可以比较完美的显示STL容器的内容，虽然与vc仍有较大差距。

缺点是在开发机的内网可能连不上很多公司的服务，而这些服务是正开发的代码所需要的。

**解决方案3:**

在windows上用VC开发。

首先需要把TCPGO移植到windows平台，这个移植的工作量非常小，因为TCPGO的代码没有使用Linux的高级特性。

有了windows版的TCPGO，就可以在自己的windows开发机上用TCPGO提供用户连接了。再自写一个用VC开发的简单的windows库，这个库里面有你的服务器程序用到的linux特有功能函数的简单windows实现，其实很多像getlong这样的unix世界的函数，网上都有现成的代码。无需考虑性能和完美支持高级特性，目的只让代码能够通过编译并完成基本功能。完成这两步，就可以做到用VC即写即调的目的。目前不清楚这个小型windows库有没有人做过。我以前倒有写过一个无需更改源代码，移植windows程序到Linux平台的Linux静态库，工作得很好。所以，做相反的事情也不是太难。

为什么不用Cygwin或Mingw做移植，为什么又重造轮子？用Cygwin或Mingw做移植也是可以的。但是，不能用VC的调试器了~~那还不如用方案一或方案二。

缺点是，对于公司其它部门提供的，没有windows版本的库，则没办法连接它们。对于某些库，应该是可以选择在调试阶段放弃的，比如上报网管平台的库。

## 5.2 为测试工程师提供一个新的自动化测试工具

目前视频广告部门的测试工程师编写测试用例时不能伪造IP源地址。TCPGO可以给测试工程师提供提接口，提供定制TCP请求功能的同时也能定制源IP地址。定制TCP请求的功能还未实现。

测试工程师还可以用TCPGO模拟大量并发TCP连接。

# 6 实战及工程经验

目前，TCPGO在自建的vmware虚拟机网络环境中达到预期设计。正计划部署到线上环境中，并把之前在TCPCOPY基础上开发的WEB前端，暨TCP流量筛选及流量比较功能迁移到TCPGO上来。

TCPGO工程从2013年12月9号开始编码到12月底基本可用，期间的生产效率远远高于12月在TCPCOPY上所做的二次开发。主要得益于大方向上，本工程的理论基础正确可行；代码逻辑遵守工业标准，有条理易调试；另开发工具选用在linux上用eclipse IDE边写边调，取代以前在windows上写代码，linux上调代码；以及全面使用git版本控制，用git快速轻便地为特定bug新建分支，在新分支上为验证bug原因添加修改代码，这一调试方法就像一个超级灵活的条件断点功能，大大加速了bug的定位和修复。

# 致 谢

谢谢Dennybai对本项目的支持，和对文档撰写时提供的帮助和指导。

谢谢Randyxu对文档提出的若干重要意见。

谢谢Tcpcopy作者wangbin579及更早的作者做出的前期探索。

谢谢Ricktian在世界观，人生观，方法论方面的交流。

谢谢Rocketzhang和Adenzhang在画图时提供的关键技术支持。

谢谢Leifzheng提供的对测试工具的顾问服务。

谢谢Dahaijiang，Peterluo陪同吃饭。

谢谢家人和朋友。

谢谢公司提供的平台。

谢谢互联网。