7-2 加餐 - TCP 全连接队列与 tcpdump 抓包

TCP 相关实验

理解 listen 的第二个参数

- 基于刚才封装的 TcpSocket 实现以下测试代码
- 对于服务器, listen 的第二个参数设置为 1, 并且不调用 accept
- 测试代码链接：https://gitee.com/whb-helloworld/linux-plus-meal/tree/master/testbacklog

test_server.cc

```
C
#include "tcp_socket.hpp"

int main(int argc, char* argv[]) {
  if (argc != 3) {
    printf("Usage ./test_server [ip] [port]\n");
    return 1;
  }
  TcpSocket sock;
  bool ret = sock.Bind(argv[1], atoi(argv[2]));
  if (!ret) {
    return 1;
  }
  ret = sock.Listen(2);
  if (!ret) {
    return 1;
  }
  // 客户端不进行 accept
  while (1) {
    sleep(1);
  }
  return 0;
}
```

test_client.cc

```
C
#include "tcp_socket.hpp"

int main(int argc, char* argv[]) {
  if (argc != 3) {
    printf("Usage ./test_client [ip] [port]\n");
    return 1;
  }
  TcpSocket sock;
  bool ret = sock.Connect(argv[1], atoi(argv[2]));
  if (ret) {
    printf("connect ok\n");
  } else {
    printf("connect failed\n");
  }
  while (1) {
    sleep(1);
  }
  return 0;
}
```

此时启动 3 个客户端同时连接服务器, 用 netstat 查看服务器状态, 一切正常.

但是启动第四个客户端时, 发现服务器对于第四个连接的状态存在问题了

```
C
tcp        3      0 0.0.0.0:9090            0.0.0.0:*
LISTEN      9084/./test_server
tcp        0      0 127.0.0.1:9090          127.0.0.1:48178
SYN_RECV    -
tcp        0      0 127.0.0.1:9090          127.0.0.1:48176
ESTABLISHED -
tcp        0      0 127.0.0.1:48178         127.0.0.1:9090
ESTABLISHED 9140/./test_client
tcp        0      0 127.0.0.1:48174         127.0.0.1:9090
ESTABLISHED 9087/./test_client
tcp        0      0 127.0.0.1:48176         127.0.0.1:9090
ESTABLISHED 9088/./test_client
tcp        0      0 127.0.0.1:48172         127.0.0.1:9090
ESTABLISHED 9086/./test_client
tcp        0      0 127.0.0.1:9090          127.0.0.1:48174
ESTABLISHED -
tcp        0      0 127.0.0.1:9090          127.0.0.1:48172
```

```
ESTABLISHED -
```

客户端状态正常, 但是服务器端出现了 SYN_RECV 状态, 而不是 ESTABLISHED 状态

这是因为, Linux 内核协议栈为一个 tcp 连接管理使用两个队列:

1. 半链接队列（用来保存处于 SYN_SENT 和 SYN_RECV 状态的请求）
2. 全连接队列（accpetd 队列）（用来保存处于 established 状态，但是应用层没有调用 accept 取走的请求）

而全连接队列的长度会受到 listen 第二个参数的影响.

全连接队列满了的时候, 就无法继续让当前连接的状态进入 established 状态了.

这个队列的长度通过上述实验可知, 是 listen 的第二个参数 + 1.

使用 TCP dump 进行抓包，分析 TCP 过程

同学们自己测试的时候要注意哦，注意和代码结合哦，我们代码中故意在close(sockfd)哪里留了一个问题

TCPDump 是一款强大的网络分析工具，主要用于捕获和分析网络上传输的数据包。

安装 tcpdump

tcpdump 通常已经预装在大多数 Linux 发行版中。如果没有安装，可以使用包管理器进行安装。例如 Ubuntu，可以使用以下命令安装：

```
Bash
sudo apt-get update
sudo apt-get install tcpdump
```

在 Red Hat 或 CentOS 系统中，可以使用以下命令：

```
Bash
sudo yum install tcpdump
```

常见使用

1. **捕获所有网络接口上的 TCP 报文**

使用以下命令可以捕获所有网络接口上传输的 TCP 报文：

```
Bash
```

```
$ sudo tcpdump -i any tcp
```

注意：`-i any` 指定捕获所有网络接口上的数据包，`tcp` 指定捕获 TCP 协议的数据包。`i` 可以理解成为 `interface` 的意思

2. **捕获指定网络接口上的 TCP 报文**

如果你只想捕获某个特定网络接口（如 eth0）上的 TCP 报文，可以使用以下命令：

```
Bash
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.45.153  netmask 255.255.192.0  broadcast
172.18.63.255
        inet6 fe80::216:3eff:fe03:959b  prefixlen 64  scopeid
0x20<link>
        ether 00:16:3e:03:95:9b  txqueuelen 1000  (Ethernet)
        RX packets 34367847  bytes 9360264363 (9.3 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34274797  bytes 6954263329 (6.9 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

$ sudo tcpdump -i eth0 tcp
```

3. **捕获特定源或目的 IP 地址的 TCP 报文**

使用 `host` 关键字可以指定源或目的 `IP` 地址。例如，要捕获源 `IP` 地址为 `192.168.1.100` 的 `TCP` 报文，可以使用以下命令：

```
Bash
$ sudo tcpdump src host 192.168.1.100 and tcp
```

要捕获目的 IP 地址为 `192.168.1.200` 的 TCP 报文，可以使用以下命令：

```
Bash
$ sudo tcpdump dst host 192.168.1.200 and tcp
```

同时指定源和目的 `IP` 地址，可以使用 `and` 关键字连接两个条件：

```
Bash
$ sudo tcpdump src host 192.168.1.100 and dst host 192.168.1.200
and tcp
```

4. **捕获特定端口的 TCP 报文**

使用 `port` 关键字可以指定端口号。例如，要捕获端口号为 `80` 的 TCP 报文（通常是

HTTP 请求），可以使用以下命令：

```
Bash
$ sudo tcpdump port 80 and tcp
```

5. **保存捕获的数据包到文件**

使用 `-w` 选项可以将捕获的数据包保存到文件中，以便后续分析。例如：

```
Bash
$ sudo tcpdump -i eth0 port 80 -w data.pcap
```

这将把捕获到的 HTTP 流量保存到名为 `data.pcap` 的文件中。

- 了解：pcap 后缀的文件通常与 PCAP（Packet Capture）文件格式相关，这是一种用于捕获网络数据包的文件格式

6. **从文件中读取数据包进行分析**

使用 `-r` 选项可以从文件中读取数据包进行分析。例如：

```
Bash
tcpdump -r data.pcap
```

这将读取 `data.pcap` 文件中的数据包并进行分析。

注意事项

- 使用 tcpdump 时，请确保你有足够的权限来捕获网络接口上的数据包。通常，你需要以 root 用户身份运行 tcpdump。
- 使用 tcpdump 的时候，有些主机名会被云服务器解释成为随机的主机名，如果不想要，就用-n 选项
- 主机观察三次握手的第三次握手，不占序号

使用 wireshark 分析 TCP 通信流程(了解)

wireshark 是 windows 下的一个网络抓包工具. 虽然 Linux 命令行中有 tcpdump 工具同样能完成抓包, 但是 tcpdump 是纯命令行界面, 使用起来不如 wireshark 方便.

下载 wireshark

https://1.na.dl.wireshark.org/win64/Wireshark-win64-2.6.3.exe

或者

链接：https://pan.baidu.com/s/159UUIoZ8b7guWDeuAHoF9A

提取码：k79r

安装 wireshark

直接双击安装, 没啥太多注意的.

启用 telnet 客户端

参考 https://jingyan.baidu.com/article/95c9d20d96ba4aec4f756154.html

启动 wireshark 并设置过滤器

由于机器上的网络数据报可能较多, 我们只需要关注我们需要的. 因此需要设置过滤器

在过滤器栏中写入

```
C
ip.addr == [服务器 ip]
```

则只抓取指定 ip 的数据包.

或者在过滤器中写入

```
C
tcp.port == 9090
```

则只关注 9090 端口的数据

更多过滤器的设置, 参考

https://blog.csdn.net/donot_worry_be_happy/article/details/80786241

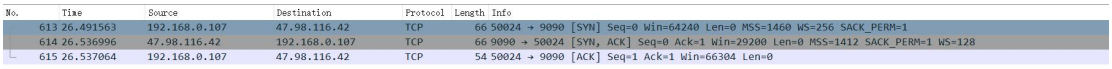
观察三次握手过程

启动好服务器.

使用 telnet 作为客户端连接上服务器

```
C
telnet [ip] [port]
```

抓包结果如下:



观察三个报文各自的序列号和确认序号的规律.

在中间部分可以看到 TCP 报文详细信息

```
Transmission Control Protocol, Src Port: 50024, Dst Port: 9090, Seq: 0, Len: 0
    Source Port: 50024
    Destination Port: 9090
    [Stream index: 23]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window size value: 64240
    [Calculated window size: 64240]
    Checksum: 0x770e [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
```

观察确认应答

在 telnet 中输入一个字符

No.	Time	Source	Destination	Protocol	Length	Info
40	2.798906	192.168.0.107	47.98.116.42	TCP	55	50024 → 9090 [PSH, ACK] Seq=1 Ack=1 Win=259 Len=1
41	2.842804	47.98.116.42	192.168.0.107	TCP	63	9090 → 50024 [PSH, ACK] Seq=1 Ack=2 Win=229 Len=9
42	2.889264	192.168.0.107	47.98.116.42	TCP	54	50024 → 9090 [ACK] Seq=2 Ack=10 Win=259 Len=0

可以看到客户端发送一个长度为 1 字节的数据, 此时服务器返回了一个 ACK 以及一个 9 个字节的响应(捎带应答), 然后客户端再反馈一个 ACK(注意观察 序列号和确认序号)

观察四次挥手

在 telnet 中输入 ctrl +], 回到 telnet 控制界面, 输入 quit 退出.

No.	Time	Source	Destination	Protocol	Length	Info
91	7.353143	192.168.0.107	47.98.116.42	TCP	54	50024 → 9090 [FIN, ACK] Seq=1 Ack=1 Win=259 Len=0
92	7.394765	47.98.116.42	192.168.0.107	TCP	54	9090 → 50024 [FIN, ACK] Seq=1 Ack=2 Win=229 Len=0
93	7.394806	192.168.0.107	47.98.116.42	TCP	54	50024 → 9090 [ACK] Seq=2 Ack=2 Win=259 Len=0

实际上是 "三次挥手", 由于捎带应答, 导致其中的两次重合在了一起.

注意事项

如果使用虚拟机部署服务器, 建议使用 "桥接网卡" 的方式连接网络. NAT 方式下由于进行了 ip 和 port 的替换.

使用云服务器测试, 更加直观方便.