计算机网络实践与应用

创新型实验项目报告

题目：

TCP和UDP数据流的带宽竞争分析

2018年 12月 1日

目 录

[第1章 绪 论 1](#_Toc531687750)

[1.1 题目来源 1](#_Toc531687751)

[1.2 项目意义 1](#_Toc531687752)

[1.3 项目创新分析 1](#_Toc531687753)

[第2章 与项目相关的主要技术及其分析 2](#_Toc531687754)

[2.1 socket编程（python） 2](#_Toc531687755)

[2.2 多线程技术 2](#_Toc531687756)

[2.3 流量的实时计算与可视化（python） 2](#_Toc531687757)

[2.4 GUI编程（python） 2](#_Toc531687758)

[第3章 混合数据流带宽竞争分析实现 3](#_Toc531687759)

[3.1 实验方案 3](#_Toc531687760)

[3.2 客户端程序的实现 3](#_Toc531687761)

[3.3 服务器端程序的实现 3](#_Toc531687762)

[3.4 带宽竞争的实现 4](#_Toc531687763)

[第4章 项目测试 4](#_Toc531687764)

[4.1 测试方案 4](#_Toc531687765)

[4.2 测试结果 4](#_Toc531687766)

[结 论 4](#_Toc531687767)

[参考文献 5](#_Toc531687768)

# 绪 论

## 1.1 题目来源

本课题来源于计算机网络实践与应用课程的创新型实验，题目为“TCP和UDP数据流的带宽竞争分析”。实验的主要目的是通过编程实现多进程TCP和UDP数据传输模块，编写相关的TCP/UDP流利用带宽测量软件，设计实验，让多个TCP和UDP流竞争有限带宽，分析结果，进而掌握TCP和UDP流传输的特点，了解工作原理和工作过程，并通过测试验证其竞争机制。

## 1.2 项目意义

在实际应用中，一个端设备往往同时存在多个TCP和UDP的连接。通过此项目，我们可以了解多连接情况下TCP和UDP的运行性能与特点，从而在实际的应用中合理分配混合数据流的TCP和UDP的连接数，从而提高数据流的传输性能，充分利用有限的带宽，提高传输效率。

## 1.3 项目创新分析

该项目通过实现编程代码来进行TCP和UDP的传输和监听，创新性地将TCP和UDP的相关内容特点与现实问题（有限带宽竞争）相联系，可以让我们从新的角度理解课程中以及书本上的TCP和UDP的特点，增强了我们用实践来检验理论知识的创新意识。

# 与项目相关的主要技术及其分析

## 2.1 socket编程（python）

Socket套接字是网络通信过程中端点的抽象表示，是程序进行网络通信的基础，使应用层能使用传输层提供的服务（TCP和UDP），进而实现TCP和UDP的数据传输模块。

本次实验中，我们使用python中的socket包，分别在client和server中将socket与IP地址和端口号建立对应关系，进而实现TCP和UDP的发送和监听。

## 2.2 多线程技术

多线程技术建立多个线程来收发数据，可以用来同时建立多个连接，进而可以在一个程序中进行TCP和UDP的同时收发，进而实现可控的连接数对网络带宽的竞争。

使用python中的threading包来实现函数在多个线程的同时运行。

## 2.3 流量的实时计算与可视化（python）

通过调用pyshark（基于wireshark的Python库）来进行实时捕获，分析每秒内不同目的端口接收的TCP和UDP数据包，计算流量。

使用python中的matplotlib来实时绘制TCP和UDP的流量曲线图，从而进行流量的可视化，体现TCP和UDP的竞争关系。

## 2.4 GUI编程（python）

Tkinter 是 Python 的标准 GUI 库。Python 使用 Tkinter 可以快速的创建 GUI 应用程序，从而使整个操作过程直接明了，易于控制。

# 第3章 混合数据流带宽竞争分析实现

## 3.1 实验方案

用python实现客户端与服务器端的应用程序client.py与server.py，

以及在服务器端运行的流量监听工具Monitor.py。

使用两台计算机设备，分别作为服务器端和客户端。

先在服务器端运行程序Server.exe（生成的可执行文件）与Monitor.py，再在客户端指定服务器IP地址与要发送的文件，进行多种情况下的TCP和UDP的竞争传输，在服务器端查看传输过程以及monitor的流量图像。

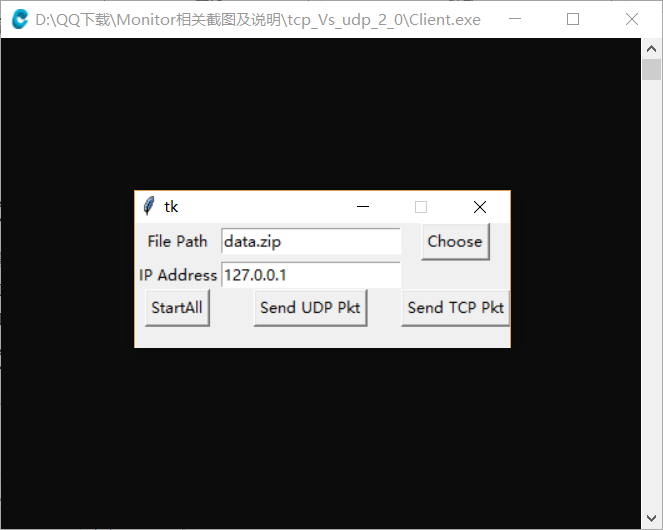
## 3.2 客户端程序的实现

**客户端程序Client.py：**

用来向服务器指定端口发送数据包，TCP的目的端口是10241，UDP的目的端口是10240。

* TCP套接字connect服务器的10241端口，建立连接，再从本地读取文件，发送文件大小后，再发送数据包（分组大小设置为BATCH\_SIZE），在发送数据的同时打印已发送数据的大小。
* UDP进行无连接传输，直接向服务器的10240端口发送文件（如果先发送文件大小, 可能丢包）。
* 为TCP和UDP传输分别启用一个线程，用户自行选择传输方式（只进行TCP传输，只进行UDP传输，两者同时进行）

运行效果：



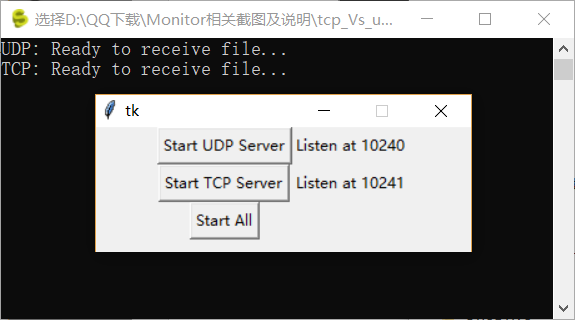
## 3.3 服务器端程序的实现

**服务器端程序Server.py:**

用来接收数据包，在10240接收UDP数据包，在10241接收TCP数据包。

* 同样启用两个线程，设置两个socket分别绑定10240和10241，用来接收数据，使用listen方法进行监听，并将接受的数据包写入指定文件中。
* TCP连接能先获知接受数据的大小。

实现效果:



**流量计算工具Monitor.py:**

通过调用pyshark（基于wireshark的Python库）进行实时捕获 ，

分析每秒内目的端口为10240的UDP数据包和10241的TCP数据包,

计算流量，使用matplotlib实时绘制TCP和UDP流量曲线图。

## 3.4 带宽竞争的实现

按照3.1中的实验方案进行操作，Client端要发送大量数据来竞争带宽，因此在操作过程中选择一个较大的文件进行TCP和UDP的传输，确保达到有限带宽内的竞争。

# 第4章 项目测试

## 4.1 测试方案

（1）基本操作过程按照3.1实验方案进行。

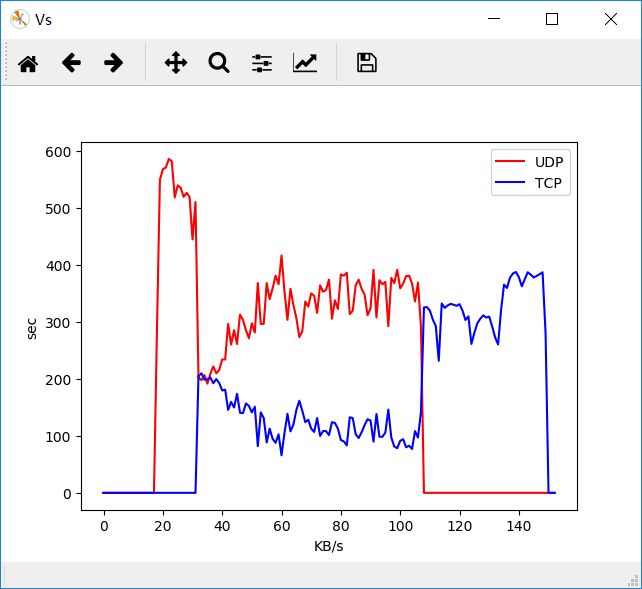
（2）TCP和UDP同时启动时，进行实验，观察流量监听界面，进行分析。

（3）TCP先启动时，UDP后启动时，进行实验分析。

（4）UDP先启动时，TCP后启动时，进行实验分析。

## 4.2 测试结果

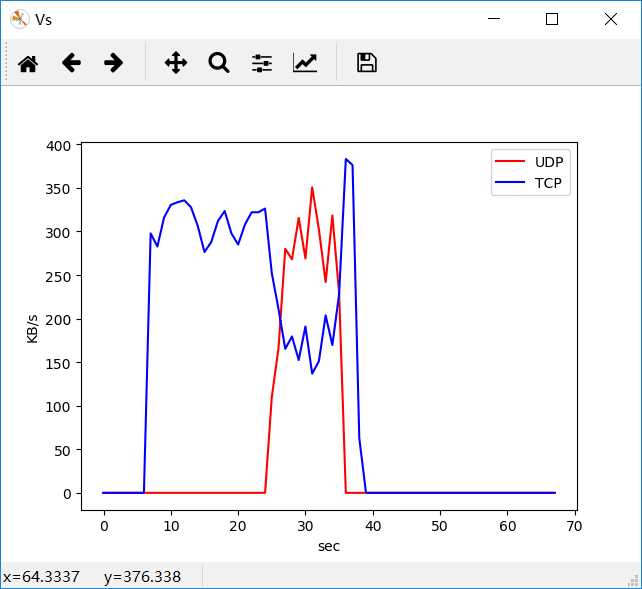
（1）TCP和UDP同时启动时，流量监听界面如下。



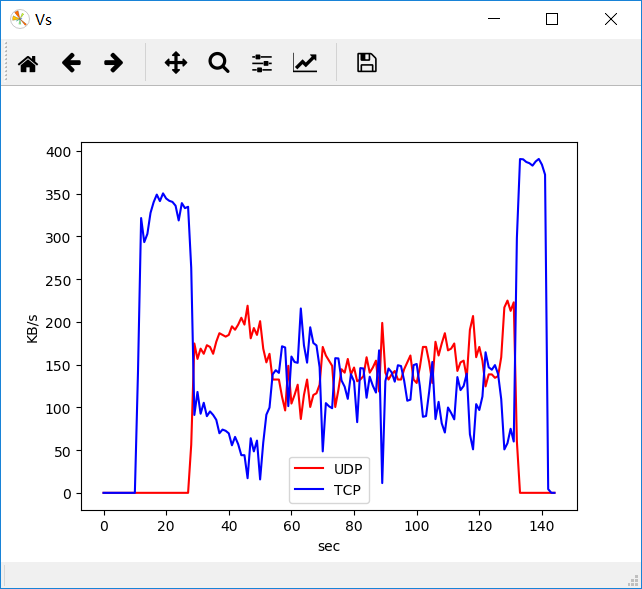
发现在同时启动的情况下，TCP传输相对于UDP传输有一个滞后的过程，UDP发送很多数据后TCP才会进行发送。TCP是面向连接的传输层服务，开始时需要建立连接（三次握手），而UDP是无连接的，直接进行数据的发送。因此会出现这种现象。

（2）TCP先启动时，UDP后启动时，流量监听界面如下。

测试1：

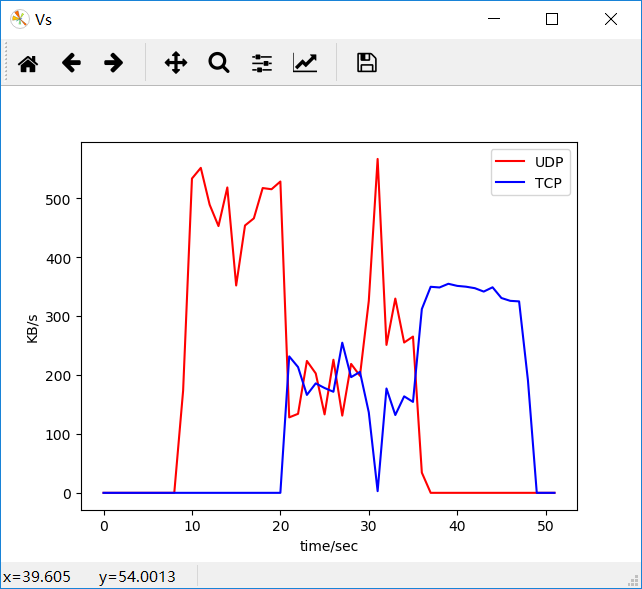


测试2：

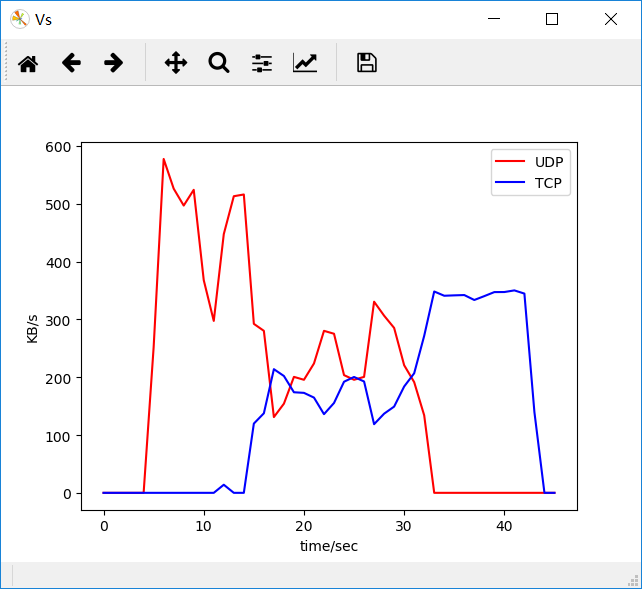


（3）UDP先启动时，TCP后启动时，流量监听界面如下。

测试1：

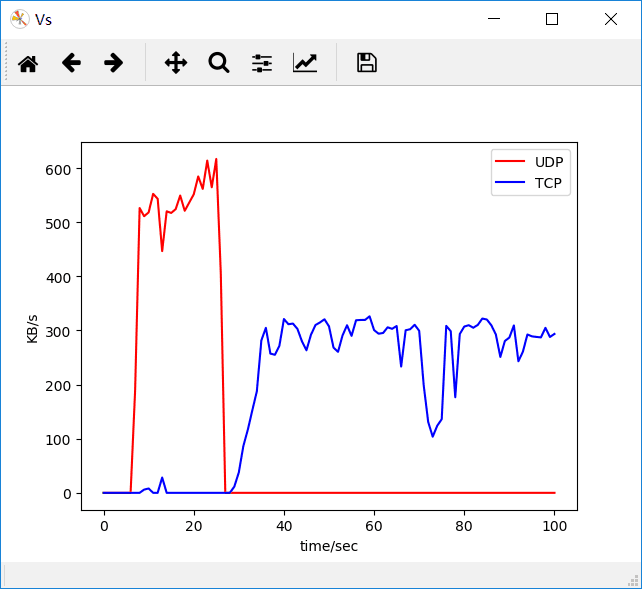


测试2：



经过以上监测到的竞争过程，我们可以发现：

在TCP和UDP竞争中，UDP会占用更多的带宽。**（还可能出现UDP完全压制TCP传输的现象，如下图所示）**只有在UDP传输完成后，TCP才能恢复到较高的速率进行传输。



# 结 论

在测试中，带宽占用率过高时，TCP和UDP竞争中，由于二者对网络拥塞的反应是不同的。TCP具有拥塞控制机制，对拥塞的处理是降低自身的传输速率，从而避免丢包的发生。而UDP缺少端到端的拥塞控制，进而在链路带宽的占用上处于优势，但会丢失一些数据包，其传输是不可靠的。

竞争会使TCP流得不到公平的带宽，因此应当探索一种机制，来处理TCP和UDP的竞争过程。

# 参考文献

[1] 刘烨. 用Socket实现基于TCP和UDP的原理探索[J]. 智能计算机与应用, 2009(3):6-8.

[2] 周喜红. TCP/UDP业务拥塞问题的研究[J]. 西安科技大学学报, 2006, 26(2):253-255.