**计算机网络编程**

**实验报告**

**班级：07111707**

**组长：1120171189 崔程远**

**成员：1120172149 吴沁璇**

**1120172153 张澈**

**1120172163 王晓媛**

**1120172736 张鉴昊**

**1120172765 曾煜瑾**

**1120173326 曾紫飞**

**北京理工大学**

**计算机学院**

**2020年5月**

**第六章 实验7 超时重传时间选择算法**

**1. 实验目的**

加深对超时重传算法的掌握和理解。

1. **实验内容**

按照TCP超时重传时间算法计算RTO的值。

配置文件关键要点：

RTT=26,32,24,…..,30

Alpha=0.125

Beita=0.25

程序运行屏幕输出要点：

显示初始RTT的值

显示本次的RTT值

计算显示平滑后RTTs

计算显示当前的RTO

重复上述步骤，一直到完成所有测量RTT数据为止

1. **实验原理**

TCP的发送方在规定的时间内没有收到确认就要重传已发送的报文段，由于TCP的下层是互联网环境，发送的报文段可能只经过一个高速率的局域网，也可能经过多个低速率的网络，并且每个IP数据报所选择的路由还可能不同。如果把超时重传时间设置得太短，就会引起很多报文段的不必要的重传，使网络负荷增大。但若把超时重传时间设置得过长，则又使网络的空闲时间增大，降低了传输效率。

因此TCP采用了一种自适应算法，它记录一个报文段发出的时间，以及收到相应的确认的时间。这两个时间之差就是报文段的往返时间RTT。TCP保留了RTT的一个加权平均往返时间RTTs（称为平滑的往返时间，因为进行的是加权平均，因此得出的结果更加光滑）。每当第一次测量到RTT样本时，RTTs值就取为所测量到的RTT样本值。但以后每测量到一个新的RTT样本，就按下式重新计算一次RTTs：

 在上式中，0 < α < 1。若α很接近于零，表示新的RTTs值和旧的RTTs值相比变化不大，而对新的RTT样本影响不大。若选择α接近于1，则表示新的RTTs值受新的RTT样本的影响较大。已成为建议标准的RFC 6298推荐的α值为1/8，即0.125。用这种方法得出的加权平均往返时间RTTs就比测量出的RTT值更加光滑。

显然，超时计时器设置的超时重传时间RTO应略大于上面得出的加权平均往返时间RTTs。RFC 6298建议使用下式计算RTO：

 而RTTd是RTT的偏差的加权平均值，它与RTTs和新的RTT样本之差有关。RFC 6298建议这样计算RTTd：当第一次测量时，RTTd值取为测量到的RTT样本值得一半，在以后的测量中，则使用下式计算加权平均的RTTd：

 这里β是个小于1的系数，它的推荐值是1/4，即0.25。

1. **实验环境**

|  |  |  |
| --- | --- | --- |
| 语言 | 集成开发环境 | 编译器 |
| C++ | Visual Studio 2017 | gcc version 4.8.1 |
| Java | Eclipse 2019 | java version "1.8.0\_65" |
| Python | Pycharm 2017 | Python 3.7.0 |

1. **实验步骤**

三份代码的结构和输出完全一致，下面以C++代码为例进行分析。

·变量定义

double RTTArray[] = { 26, 32, 24, 26, 26, 28, 26, 26, 28, 26, 26, 28, 30 };

double Alpha = 0.125, Beita = 0.25;

·第一次计算（特殊情况）

double RTT, RTTs, RTTd, RTO;

RTT = RTTArray[0];

RTTs = RTT;

RTTd = RTT / 2;

RTO = RTTs + 4 \* RTTd;

printf("Initial RTT: %lf\n", RTT);

·之后处理过程

for (int i = 1; i < sizeof(RTTArray) / sizeof(RTTArray[0]); i++)

{

RTT = RTTArray[i];

RTTs = (1 - Alpha) \* RTTs + Alpha \* RTT;

RTTd = (1 - Beita) \* RTTd + Beita \* fabs(RTTs - RTT);

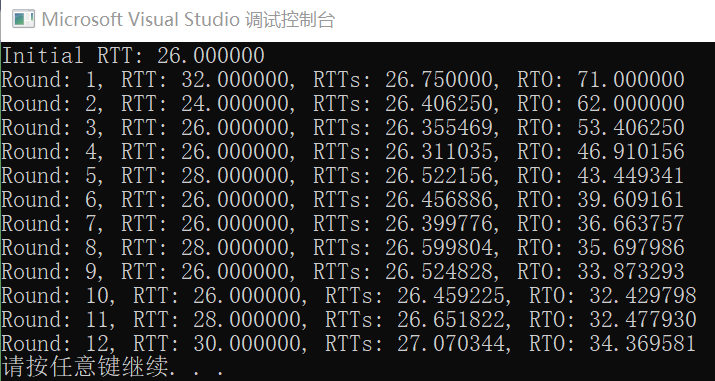
RTO = RTTs + 4 \* RTTd;

printf("Round: %d, RTT: %lf, RTTs: %lf, RTO: %lf\n", i, RTT, RTTs, RTO);

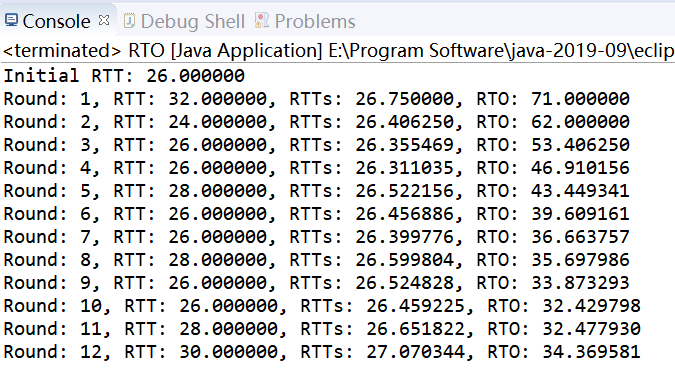
}

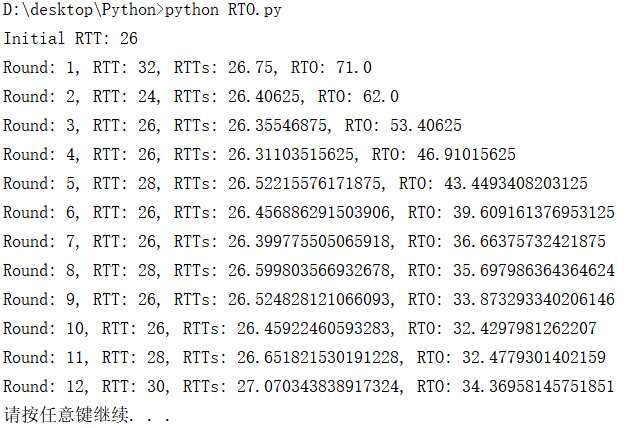
1. **实验结果**

**·C++**



**·Java**

**·python**



1. **实验总结**

这个实验很简单，用几个公式运算下进行了，从结果可以看出RTTs很平滑，RTO也逐渐从一开始远大于RTTs到慢慢地缩小差距。