

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。

2. 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

3. 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：

$$\text{新的 RTTs} = (1 - \alpha) \times (\text{旧的 RTTs}) + \alpha (\text{新的 RTT 样本})$$

在上式中， $0 < \alpha < 1$ 。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：

$$\text{RTO} = \text{RTTs} + 4 \times \text{RTTd}$$

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：

新的 $RTTD = (1-\beta) \times (\text{旧的} RTTD) + \beta \times |RTTs - \text{新的} RTT \text{ 样本}|$

这里 β 是个小于 1 的系数，它的推荐值是 1/4，即 0.25。

4. 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

5. 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);
```

```
}
```

6. 实验结果

- C++

Microsoft Visual Studio 调试控制台

```
Initial RTT: 26.000000
Round: 1, RTT: 32.000000, RTTs: 26.750000, RTO: 71.000000
Round: 2, RTT: 24.000000, RTTs: 26.406250, RTO: 62.000000
Round: 3, RTT: 26.000000, RTTs: 26.355469, RTO: 53.406250
Round: 4, RTT: 26.000000, RTTs: 26.311035, RTO: 46.910156
Round: 5, RTT: 28.000000, RTTs: 26.522156, RTO: 43.449341
Round: 6, RTT: 26.000000, RTTs: 26.456886, RTO: 39.609161
Round: 7, RTT: 26.000000, RTTs: 26.399776, RTO: 36.663757
Round: 8, RTT: 28.000000, RTTs: 26.599804, RTO: 35.697986
Round: 9, RTT: 26.000000, RTTs: 26.524828, RTO: 33.873293
Round: 10, RTT: 26.000000, RTTs: 26.459225, RTO: 32.429798
Round: 11, RTT: 28.000000, RTTs: 26.651822, RTO: 32.477930
Round: 12, RTT: 30.000000, RTTs: 27.070344, RTO: 34.369581
请按任意键继续. . .
```

- Java

```
Console  Debug Shell  Problems
<terminated> RTO [Java Application] E:\Program Software\java-2019-09\eclip
Initial RTT: 26.000000
Round: 1, RTT: 32.000000, RTTs: 26.750000, RTO: 71.000000
Round: 2, RTT: 24.000000, RTTs: 26.406250, RTO: 62.000000
Round: 3, RTT: 26.000000, RTTs: 26.355469, RTO: 53.406250
Round: 4, RTT: 26.000000, RTTs: 26.311035, RTO: 46.910156
Round: 5, RTT: 28.000000, RTTs: 26.522156, RTO: 43.449341
Round: 6, RTT: 26.000000, RTTs: 26.456886, RTO: 39.609161
Round: 7, RTT: 26.000000, RTTs: 26.399776, RTO: 36.663757
Round: 8, RTT: 28.000000, RTTs: 26.599804, RTO: 35.697986
Round: 9, RTT: 26.000000, RTTs: 26.524828, RTO: 33.873293
Round: 10, RTT: 26.000000, RTTs: 26.459225, RTO: 32.429798
Round: 11, RTT: 28.000000, RTTs: 26.651822, RTO: 32.477930
Round: 12, RTT: 30.000000, RTTs: 27.070344, RTO: 34.369581
```

- python

```
D:\desktop\Python>python RTO.py
Initial RTT: 26
Round: 1, RTT: 32, RTTs: 26.75, RTO: 71.0
Round: 2, RTT: 24, RTTs: 26.40625, RTO: 62.0
Round: 3, RTT: 26, RTTs: 26.35546875, RTO: 53.40625
Round: 4, RTT: 26, RTTs: 26.31103515625, RTO: 46.91015625
Round: 5, RTT: 28, RTTs: 26.52215576171875, RTO: 43.4493408203125
Round: 6, RTT: 26, RTTs: 26.456886291503906, RTO: 39.609161376953125
Round: 7, RTT: 26, RTTs: 26.399775505065918, RTO: 36.66375732421875
Round: 8, RTT: 28, RTTs: 26.599803566932678, RTO: 35.697986364364624
Round: 9, RTT: 26, RTTs: 26.524828121066093, RTO: 33.873293340206146
Round: 10, RTT: 26, RTTs: 26.45922460593283, RTO: 32.4297981262207
Round: 11, RTT: 28, RTTs: 26.651821530191228, RTO: 32.4779301402159
Round: 12, RTT: 30, RTTs: 27.070343838917324, RTO: 34.36958145751851
请按任意键继续. . .
```

7. 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。