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

第五章 实验 3　IP 数据报分片和重装程序

1. 实验目的

模拟网络层 IP 数据报的分片和重装，加深对 IP 数据报分片和重装的掌握和理解。

2. 实验内容

编写一个 IPv4 数据报分片和重装程序。

配置文件关键要点：

//以下是分片相关的配置参数

BigIPTotalLen=4000

ID=666

MTU=1500

//以下是重装相关的配置参数

FragNum=3

TotalLen=1500,1500,104

ID=888,888,888

FragMF=1,1,0

FragOffset=0,185,370

程序运行屏幕输出要点：

首先屏幕显示原始大数据报的主要分片字段信息，包括总长度，标识，标志（DF，MF）和片段偏移

输出最大 MTU 的值

计算并显示产生的分片数

计算并显示每个分片的信息，包括：总长度，标识，标志（DF，MF）和片段偏移

屏幕显示即将完成分片的重装，显示分片的数量

显示每个分片的信息，包括：总长度，标识，标志（DF，MF）和片段偏移

计算并显示重装后数据报的分片信息，包括：总长度，标识，标志（DF，MF）和片段偏移

3. 实验原理

IP 数据报包含多个字段，然而本实验模拟的 IP 数据报中只包含总长度、标识、DF、MF 和片段偏移。

• 总长度：总长度指首部和数据之和的长度，单位为字节。在 IP 层下面的每一种数据链路层协议都规定了一个数据帧中的数据字段的最大长度，称为最大传送单元（MTU），当一个 IP 数据报封装成链路层的帧时，此数据报的总长度一定不能超过下面的数据链路层所规定的的 MTU 值。若所传送的数据报长度超过数据链路层的 MTU 值，就必须把过长的数据报进行分片处理。

• 标识：数据报由于长度超过网络的 MTU 而必须分片时，这个标识字段的值就被复制到所有的数据报片的标识字段中，相同的标识字段的值使得分片后的各数据报片最后能正确地重装成为原来的数据报。

• DF：（Don't Fragment），意思是“不能分片”，只有当 DF = 0 时才允许分片。

• MF：（More Fragment），MF = 1 表示后面还有分片的数据段，MF = 0 表示这已经是若干数据报片中的最后一个。

•片段偏移：片段偏移指出，较长的分组在分片后，某片在原分组中的相对位置。也就是说，相对于用户数字段的起点，该片从何处开始。片段偏移以 8 个字节为偏移单位。

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，均采用面向对象的构造方法，以 C++代码为例进行分析。

• 定义全局变量

```
private:
    int headLen = 20;
    int bigIPTotalLen = 4000;
    int ID = 666;
    int MTU = 1500;

    int totalLenPos = 0;
    int IDPos = 1;
    int MFPos = 2;
    int DFPos = 3;
    int offsetPos = 4;

    int isEnd = 0;
    int isNotEnd = 1;
    int couldFragment = 0;
    int MAX = 9999;

    int original[5];
    int buffer[bufferLen][5];
    int assemble[5];
```

• 对原始大数据报初始化并打印主要分片字段信息

```
void Initialize()
    {
        original[totalLenPos] = bigIPTotalLen;
        original[IDPos] = ID;
        original[MFPos] = isEnd;
        original[DFPos] = couldFragment;
        original[offsetPos] = 0;
        printf("For original datagram, TotalLen | ID | MF | DF | Offset: %d | %d | %d | %d | %d\n\n",
            original[totalLenPos], original[IDPos], original[MFPos], original[DFPos], original[offsetPos]);
```

```
        for (int i = 0; i < bufferLen; i++)
        {
            buffer[i][offsetPos] = MAX;
        }
    }
```

·分片并打印每个分片的信息

```
void Split()
    {
        int remainder = original[totalLenPos] - headLen;
        cout << "The biggest MTU: " << MTU << endl;
        int number = 0;
        while (remainder > 0)
        {
            if (remainder > (MTU - headLen))
            {
                buffer[number][totalLenPos] = MTU;
                buffer[number][IDPos] = original[IDPos];
                buffer[number][MFPos] = isNotEnd;
                buffer[number][DFPos] = couldFragment;
                buffer[number][offsetPos] = ((original[totalLenPos] - headLen) -
remainder) / 8;
                remainder -= (MTU - headLen);
            }
            else
            {
                buffer[number][totalLenPos] = remainder + headLen;
                buffer[number][IDPos] = original[IDPos];
                buffer[number][MFPos] = original[MFPos];
                buffer[number][DFPos] = couldFragment;
                buffer[number][offsetPos] = ((original[totalLenPos] - headLen) -
remainder) / 8;
                remainder = 0;
            }
            number++;
        }
        cout << "The number of sub-datagram: " << number << endl;
        cout << "The infomation of each sub-datagram: " << endl;
        printf("TotalLen\tID\tMF\tDF\tOffset\n");
        for (int i = 0; i < number; i++)
        {
            printf("%d\t\t%d\t%d\t%d\t%d\n", buffer[i][totalLenPos],
buffer[i][IDPos], buffer[i][MFPos],
                buffer[i][DFPos], buffer[i][offsetPos]);
        }
```

```
        cout << endl;
    }
```

• 重装并打印相关输出

```
void Combine()
    {
        int number;
        for (number = 0; number < bufferLen; number++)
        {
            if (buffer[number][MFPos] == isEnd)
            {
                number++;
                break;
            }
        }
        cout << "Be ready to assemble, the number of sub-datagram: " << number <<
endl;
        cout << "The infomation of each sub-datagram: " << endl;
        printf("TotalLen\tID\tMF\tDF\tOffset\n");
        for (int i = 0; i < number; i++)
        {
            printf("%d\t\t%d\t%d\t%d\t%d\n", buffer[i][totalLenPos],
buffer[i][IDPos], buffer[i][MFPos], buffer[i][DFPos],
                buffer[i][offsetPos]);
        }
        assemble[totalLenPos] = (buffer[number - 1][offsetPos] -
buffer[0][offsetPos]) * 8
            + buffer[number - 1][totalLenPos];
        assemble[IDPos] = buffer[0][IDPos];
        assemble[MFPos] = buffer[number - 1][MFPos];
        assemble[DFPos] = buffer[0][DFPos];
        assemble[offsetPos] = buffer[0][offsetPos];
        cout << endl;
        printf("After assembling, TotalLen | ID | MF | DF | Offset: %d | %d | %d
| %d | %d\n", assemble[totalLenPos],
            assemble[IDPos], assemble[MFPos], assemble[DFPos],
assemble[offsetPos]);
    }
```

• 主函数

```
int main()
{
    SplitAndCombine operation;
    operation.Initialize();
    operation.Split();
    operation.Combine();}
```

6. 实验结果

- C++

```
D:\desktop\C++\x64\Debug>SplitAndCombine.exe
For original datagram, TotalLen | ID | MF | DF | Offset: 4000 | 666 | 0 | 0 | 0

The biggest MTU: 1500
The number of sub-datagram: 3
The infomation of each sub-datagram:
TotalLen        ID      MF      DF      Offset
1500            666     1       0       0
1500            666     1       0       185
1040            666     0       0       370

Be ready to assemble, the number of sub-datagram: 3
The infomation of each sub-datagram:
TotalLen        ID      MF      DF      Offset
1500            666     1       0       0
1500            666     1       0       185
1040            666     0       0       370

After assembling, TotalLen | ID | MF | DF | Offset: 4000 | 666 | 0 | 0 | 0
请按任意键继续. . .
```

- **Java**

```
Console  Debug Shell  Problems
<terminated> SplitAndCombine [Java Application] E:\Program Software\java-2019-09\eclipse\jre\bin\
For original datagram, TotalLen | ID | MF | DF | Offset: 4000 | 666 | 0 | 0 | 0

The biggest MTU: 1500
The number of sub-datagram: 3
The infomation of each sub-datagram:
TotalLen        ID      MF      DF      Offset
1500            666     1       0       0
1500            666     1       0       185
1040            666     0       0       370

Be ready to assemble, the number of sub-datagram: 3
The infomation of each sub-datagram:
TotalLen        ID      MF      DF      Offset
1500            666     1       0       0
1500            666     1       0       185
1040            666     0       0       370

After assembling, TotalLen | ID | MF | DF | Offset: 4000 | 666 | 0 | 0 | 0
```

- Python

```
D:\desktop\Python>python SplitAndCombine.py
For original datagram, TotalLen | ID | MF | DF | Offset: 4000 | 666 | 0 | 0 | 0

The Biggest MTU: 1500
The number of sub-datagram: 3
The infomation of each sub-datagram:
TotalLen        ID      MF      DF      Offset
1500            666     1       0       0
1500            666     1       0       185
1040            666     0       0       370

Be ready to assemble, the number of sub-datagram: 3
The infomation of each sub-datagram:
TotalLen        ID      MF      DF      Offset
1500            666     1       0       0
1500            666     1       0       185
1040            666     0       0       370

After assembling, TotalLen | ID | MF | DF | Offset: 4000 | 666 | 0 | 0 | 0
请按任意键继续. . .
```

7. 实验总结

这个实验相对简单，根据 MTU 对原始大数据报分片并赋予相关的属性，保存在 buffer 数组中。重装的时候先对所有的分片根据偏移量进行升序排列，然后根据偏移量重装即可。然而我有个疑问就是题目中给的两个 ID 不一致，各分片数据报和原始大数据报的 ID 不应该一致吗，我觉得应该是一致的，所以在这个实验中我把 ID 定为 666.