**组流技术与流特征分析实验报告**

学院：

班级：

学号：

姓名：

# 实验背景

在网络测量领域，流是指在一定时间内通过网络传输的一系列数据包。这些数据包通常具有相同的源地址、目的地址和协议信息。流可以划分为单向流（One-way Flow）和双向流（Bidirectional Flow）两种类型，它们的概念如下：

单向流（One-way Flow）

单向流是指从一个源（发送端）到一个目的地（接收端）的一组连续的数据包。这些数据包具有相同的源IP地址、目的IP地址、源端口、目的端口和协议类型。在单向流中，数据传输只在一个方向上进行。不考虑从接收端返回的任何数据包，如确认包（ACK）等。单向流通常用于分析来自特定源的数据传输特性，例如数据包丢失、延迟和抖动。

双向流（Bidirectional Flow）

双向流是指在两个方向上进行数据传输的一组数据包，即从源（发送端）到目的地（接收端）以及从目的地（接收端）返回到源（发送端）。双向流中的数据包具有相互对称的源IP地址、目的IP地址、源端口、目的端口和协议类型。双向流表示了典型的网络通信模式，例如TCP会话，其中数据包在发送端和接收端之间来回传输。双向流可以用于分析网络性能，如往返时延（RTT）和双向传输速率。

总结一下，单向流关注的是网络中仅在一个方向上进行的数据传输，而双向流关注的是双向交互的数据传输。在网络测量中，研究人员可以根据需要选择分析单向流或双向流，以获取关于网络性能和特性的详细信息。

# 实验内容

## 2.1 使用文档及运行样例

### 2.1.1 项目结构

文本, 信件

描述已自动生成

### 2.1.2 使用方法

在IDE中直接运行即可，或在终端中输入运行：

python ./flow\_aggregation.py

## 2.2 UML图与主要数据结构

文本

描述已自动生成

## 2.3 主要算法说明

### 2.3.1 任务一：处理pcap文件

使用dpkt库中的pcap.Reader()方法可以实现对pcap文件的读取，返回的数据类型为dpkt.pcap.Reader，再通过readpkts()方法可以读取其中的数据包，并返回所有数据包组成的列表。

接下来遍历数据包，返回每个数据包的时间戳和原始二进制数据组成的列表。

在控制台打印数据包的部分信息：包括IP层数据包长度、TTL和应用层数据，输出结果详见2.4.1节。

def pcap\_read(*pcap\_file*):  
 pcap = dpkt.pcap.Reader(open(*pcap\_file*, "rb"))  
 pkt\_list = pcap.readpkts()  
 pkt\_result = []  
 tms\_result = []  
 for (ts, pkt) in pkt\_list:  
 try:  
 ip\_packet = get\_IP\_packet(pkt)  
 print("该Pcap数据包IP层数据包长度：", ip\_packet.len)  
 print("该Pcap数据包存活时间：", ip\_packet.ttl)  
 trans\_packet = ip\_packet.data # 传输层的数据  
 data = trans\_packet.data # 应用层数据  
 print("该Pcap数据包应用层数据：", data)  
 pkt\_result.append(pkt)  
 tms\_result.append(ts)  
 except Exception as e:  
 print(e)  
 continue  
 return tms\_result, pkt\_result

### 2.3.2 任务二：组流

将流分为单向流和双向流（判断逻辑有所不同）。

对于单向流而言，五元组（源、目的IP，源、目的端口号，传输层协议）完全匹配即可归为同一条流；而对于双向流，则还要添加源IP、端口号与目的IP、端口号调换的判断，具体可见代码中的if语句（双向流的判断因为较长所以先设置了一个flag）

整个flow\_combine()方法中有两层循环，第一层循环是遍历传入的数据包，第二层循环是遍历现有的流，如果有匹配的流，则添加进该流中（调用Flow类中的append\_packet()方法将数据包加入流，并将is\_exist设为True），进入下一个循环（第一层循环）；如过遍历完已有的流没有找到匹配项，则添加一条新的流，五元组使用该数据包的值。

下面的代码为判断相同流的部分。

if *flow\_definition* == 1: # 单向流  
 for flow\_unit in flow\_list:  
 # 根据五元组判断是否同流  
 if ((flow\_unit.src\_ip, flow\_unit.dst\_ip, flow\_unit.src\_port, flow\_unit.dst\_port,  
 flow\_unit.trans\_layer\_proto) == (src\_ip, dst\_ip, src\_port, dst\_port, trans\_layer\_proto)):  
 # 同流则将该数据包加入  
 flow\_unit.append\_packet(pkt, tms, pkt\_size)  
 flow\_unit.length += 1  
 flow\_is\_exist = True  
 break  
elif *flow\_definition* == 2: # 双向流  
 for flow\_unit in flow\_list:  
 # 双向流还应添加源、目的相反的判断逻辑  
 same\_flow = (((flow\_unit.src\_ip, flow\_unit.dst\_ip, flow\_unit.src\_port,  
 flow\_unit.dst\_port, flow\_unit.trans\_layer\_proto) == (  
 src\_ip, dst\_ip, src\_port, dst\_port, trans\_layer\_proto)) or (  
 (flow\_unit.src\_ip, flow\_unit.dst\_ip, flow\_unit.src\_port, flow\_unit.dst\_port,  
 flow\_unit.trans\_layer\_proto) == (  
 dst\_ip, src\_ip, dst\_port, src\_port, trans\_layer\_proto)))  
 if same\_flow:  
 # 同流则将该数据包加入  
 flow\_unit.append\_packet(pkt, tms, pkt\_size)  
 flow\_unit.length += 1  
 flow\_is\_exist = True  
 break

流归类完成后打印输出整条流的相关信息，代码如下：

def print\_flow\_list(*flow\_list*):  
 print('流总数：', len(*flow\_list*))  
 idx = 0  
 for flow in *flow\_list*:  
 idx += 1  
 if flow.trans\_layer\_proto == dpkt.ip.IP\_PROTO\_TCP:  
 protocol = 'TCP'  
 else:  
 protocol = 'UDP'  
 print('第', idx, '条流：')  
 print(' - [{}, {}, {}, {}, {}]'.format(inet\_to\_str(flow.src\_ip), inet\_to\_str(flow.dst\_ip), flow.src\_port,  
 flow.dst\_port, protocol))  
 print(' - 源MAC地址：', flow.src\_mac)  
 print(' - 目的MAC地址：', flow.dst\_mac)  
 print(' - 报文长度序列：', flow.pkt\_size\_seq)  
 print(' - 报文长度方差：', flow.var)

将IP地址格式化后输出，协议相应地转为TCP或UDP打印，方差的计算见下一节，打印输出结果详见2.4.2节。

### 2.3.3 任务三：流特征分析

每个流的报文长度序列在上一节的append\_packet()方法中已经包含，即把数据包长度添加进Flow的成员变量pkt\_size\_seq中，下面的代码是流的报文长度方差的计算。

传入流列表，对每个流的报文长度序列使用numpy库的var()方法计算方差，为了便于打印，在Flow类中添加了一个成员变量var用来存储方差，返回的方差列表则是用于后续画图。

def var\_set\_calc(*flow\_list*):  
 var\_set = []  
 for flow in *flow\_list*:  
 flow.var = np.var(flow.pkt\_size\_seq)  
 var\_set.append(flow.var)  
 return var\_set

### 2.3.4 任务四：实现简单的数据可视化

绘制直方图的代码如下。需要传入的参数有：x - 用于绘制直方图的数据集；plotspot – 子图的位置描述；title – 子图标题。为了避免图中部分值过大，导致其他值的相对关系不明显，需要对最大数据计算对数，并根据对数值生成一组对数均匀分布的直方柱：

def hist\_log\_plot(*x*, *plotspot*, *title*):  
 logbins = np.append([0], np.power(10, np.arange(0, int(np.log10(max(*x*))) + 1, 0.5)))  
 plt.subplot(*plotspot*)  
 plt.hist(*x*, bins=logbins)  
 plt.xscale('log')  
 plt.title(*title*)  
 plt.xlabel("报文长度方差")  
 plt.ylabel("流数量")

在主函数中，需要分别计算出单向流和双向流的方差，调用绘图函数绘制两个子图：

# 单向流  
flow\_list\_one\_way = flow\_combine(pkt1, tms1, 1) # 组流  
var\_set\_one\_way = var\_set\_calc(flow\_list\_one\_way) # 计算方差  
print\_flow\_list(flow\_list\_one\_way) # 打印流信息  
hist\_log\_plot(var\_set\_one\_way, 221, "方差分布-单向流") # 绘制流方差分布图  
  
# 双向流  
flow\_list\_two\_way = flow\_combine(pkt1, tms1, 2) # 组流  
var\_set\_two\_way = var\_set\_calc(flow\_list\_two\_way) # 计算方差  
print\_flow\_list(flow\_list\_two\_way) # 打印流信息  
hist\_log\_plot(var\_set\_two\_way, 222, "方差分布-双向流") # 绘制流方差分布图  
plt.show()

绘制结果详见2.4.4节。

## 2.4 结果展示

### 2.4.1 任务一：处理pcap文件

在控制台打印了pcap中每个数据包的部分信息：包括IP层数据包长度、TTL和应用层数据：

背景图案

低可信度描述已自动生成

### 2.4.2 任务二：组流

在控制台打印了流的总数、每条流的五元组、源、目的MAC、报文长度序列、方差：

图片包含 背景图案

描述已自动生成

### 2.4.3 任务三：流特征分析

已经包含在2.4.2的输出中。

### 2.4.4 任务四：实现简单的数据可视化

分别对单向流和双向流的报文长度方差绘制直方图：

图表, 直方图

描述已自动生成

通过观察流数-方差分布图，可以看出对于单向流而言，报文长度方差为0的流占了大多数，这类流可能为传输的数据包很少，也可能是是发送的一串数据包都是相同的内容，如心跳包、网络探测等，在观察流的详细信息输出时发现两种都有，第一种（只有一个数据包的流）更多一些；除去方差为零的流，其余的流的方差大多位于100至10^5间。

对于双向流来说，方差为零的的流不再占比最多，与单向流相对比，发现少了很多只有一个数据包的流，其余的流的方差大多集中在10^3附近。

# 实验总结

在本次实验中，更清楚了流的归类、划分，并且学习到了单向流、双向流的概念，通过代码实现的过程理解了在归类上有何不同之处。在作图过程中学习了对于为了避免图中部分值过大，导致其他值的相对关系不明显，需要对最大数据计算对数，并根据对数值生成一组对数均匀分布的直方柱的方法。