正式的中文名称

子标题如冬令营

正式的简称

网络流量分析与处理

时间：2019 年 5 月 26 日 18:00 ~ 21:00

题目名称	PCAP 文件解析与读写	常见网络校验和计算	数据链路层协议数据处理	网络层协议数据处理	RIP 路由协议实现	期待你的声音
题目类型	传统型	传统型	传统型	传统型	传统型	提交答案型
目录	pcap	checksum	layer2	layer3	routing	feedback
可执行文件名	pcap	checksum	layer2	layer3	routing	N/A
输入文件名	pcap.in	checksum.in	layer2.in	layer3.in	routing.in	feedback*.in
输出文件名	pcap.out	checksum.out	layer2.out	layer3.out	routing.out	feedback*.out
每个测试点时限	1.0 秒	2.0 秒	2.0 秒	2.0 秒	8.0 秒	N/A
内存限制	512 MB	512 MB	512 MB	512 MB	512 MB	N/A
测试点数目	5	5	5	5	10	1
测试点是否等分	否	否	否	否	否	否
预测试点数目	5	5	5	5	10	0

提交源程序文件名

对于 C++ 语言	pcap.cpp	checksum.cpp	layer2.cpp	layer3.cpp	routing.cpp	N/A

编译选项

对于 C++ 语言	-O3 -march=native -std=c++14	-O3 -march=native -std=c++14	-O3 -march=native -std=c++14	-O3 -march=native -std=c++14	-O3 -march=native -std=c++14	N/A

PCAP 文件解析与读写（pcap）

【题目背景】

芃芃是华清大学计算机系的一名学生，马上读大二上学期。由于他精通 OI，觉得这个学期的唯一一门专业课《数据结构》实在是太简单了。在学长杰哥的盛情推荐下，他提前选了大三的专业课《计算机网络原理》。然而，这门课程的大作业是自己动手实现一个软件路由器，这可难倒了没有工程经验的芃芃。为了不让杰哥失望，他决定在暑假里就提前完成这个任务。

本题一共分为 6 个小题，难度递增。你将与芃芃一起学习网络的基本原理、网络设备的工作方式，并亲手实现一个个功能，最终完成一个简化版的路由器软件。麻雀虽小，五脏俱全，它将能对多种基本的网络协议进行处理，并担任起路由器最根本的使命——转发。

在本题**所有的小题**中，我们都有以下的约定：

- 你的计算机上有 16 个网络接口，编号从 1 到 16。
- 你的计算机上默认的路由表为空，并且只能从 RIP 报文获得路由表的更新。
- 每个接口对应**一个** IP 地址 `10.2.n.1/24`，其中 n 代表接口编号，范围是 1 到 16。
- 每个接口对应一个 MAC 地址 `98:01:29:00:00:xx`，其中 `xx` 是接口编号的 16 进制表示，范围是 01_{16} 到 10_{16}。
- 所有的流量片段，包括读取和写入的，都是以太网帧（包括帧头和帧尾的校验），包含完整的协议格式。
- 除特殊说明外，输入的 PCAP 文件中的流量片段都是按照时间顺序排列的（时间戳单调递增），只需要进行依次处理即可。

如果你不能理解上述的约定，请参见《学习手册》网络基础相关部分。

【题目描述】

所有题目中，网络流量都将以 PCAP 文件格式保存，具体格式说明可参见《学习手册》。本题需要对 PCAP 文件进行基本的读写与解析。

在本题中，你需要完成的任务是依次遍历输入文件中时间乱序的流量片段，在对它们进行一些操作后，组装为 PCAP 格式写入输出。关于文件的读写方式，可以参照《学习手册》。

需要进行的操作为：只保留长度**不大于** 1000 的片段（只考虑完整的以太网帧长度，不包括 PCAP 格式的片段头），并按照时间升序排序这些片段（即首先比较秒，而后比较微秒），保证没有两个片段的时间是相同的。需要注意，你的输出文件需要是一个合法的 PCAP 文件。为了测试文件格式的合法性，你可以使用 Wireshark 等工具打开你的输出文件进行检查。

与手册中指示的不同的是，本题中，你应该直接完整复制每个流量片段的头，包括时间戳字段。

【输入格式】

从文件 ***pcap.in*** 中读入数据。

输入包含不超过 n 个流量片段，总大小不超过 m 字节。

【输出格式】

输出到文件 ***pcap.out*** 中。

你的输出将与答案文件进行逐字节对比。请不要输出任何多余信息，以免导致不必要的失分。

【子任务】

测试点	n	m
1	$= 10^2$	$= 5 \times 10^4$
2		$= 1.5 \times 10^5$
3	$= 10^3$	$= 1.5 \times 10^6$
4	$= 10^4$	$= 1.5 \times 10^7$
5	$= 10^5$	$= 1.5 \times 10^8$

【样例 1】

见选手目录下的 ***pcap/pcap1.in*** 与 ***pcap/pcap1.ans***。

【样例 1 解释】

你可以用你电脑上安装的 Wireshark 软件打开样例数据的输入和输出。

用 Wireshark 打开样例输入文件，应该可以看到，界面上部记录了这个 PCAP 文件中记录的所有以太网帧，界面中部显示的是当前这个以太网帧的解析，下部则是它的十六进制数据。对于这个软件的详细使用方法，除了部分题目给出的提示外，不会提供更多帮助，请选手自行熟悉并摸索。

样例输入文件记录了两个以太网帧，都是从 `10.2.254.100` 发往 `10.2.12.82` 地址的 ARP 请求，它们区别在于 ARP 请求后多余的数据长度不同。由于第一个帧的长度不大于 1000 ，第二个帧的长度大于 1000，按照题目意思，只有第一个帧需要输出，排序后输出到了样例输出文件中，你可以同样地用 Wireshark 打开样例输出文件来验证这一点。

常见网络校验和计算（checksum）

【题目背景】

芃芃发现，华清大学的玉兰学生公寓的网络质量似乎不是很好。由于网络线材质量的原因，也可能是有同学在进行一些奇怪的尝试，他总是会收到一些不太正常的内容。好在，网络协议的设计者们都考虑到了这一点，在协议中添加了校验和来检查一些明显的问题。作为鲁棒的路由器软件，必须只接收校验和检查通过的流量，而丢弃那些坏的。听说你已经能读到流量的内容了，下一步就是揪出这些不正常的流量。

【题目描述】

你需要根据《学习手册》中的相关知识，对给定的流量片段进行校验。请注意，由于所有片段类型**都是以太网帧**，所以你总是需要计算 `CRC32`；而对于承载了 IP 分组的流量，你还需要对 IP 分组头的检验和进行检查。作为简化，我们不考虑除了以太网和 IP 以外的校验字段。只有当所有的校验都通过时，才认为该流量片段是正确的。

【输入格式】

从文件 ***checksum.in*** 中读入数据。

输入保证格式合法，并且每个流量片段都是以太网帧。其中包含不超过 n 个流量片段，总大小不超过 m 字节。部分子任务的数据中保证 IP 分组头的校验是正确的，即只需要计算以太网帧的 `CRC32`。

【输出格式】

输出到文件 ***checksum.out*** 中。

输出由 n 行文本构成，依次对应每一个片段的校验结果。如果是正确的，输出 `Yes`，否则输出 `No`。

【子任务】

测试点	n	m	仅需检查以太网帧
1	$= 10^2$	$= 5 \times 10^4$	是
2		$= 1.5 \times 10^5$	否
3	$= 10^3$	$= 1.5 \times 10^6$	
4	$= 10^4$	$= 1.5 \times 10^7$	
5	$= 10^5$	$= 1.5 \times 10^8$	

【样例 1】

见选手目录下的 ***checksum/checksum1.in*** 与 ***checksum/checksum1.ans***。

【样例 1 解释】

和 `pcap` 题一样，你可以用你电脑上安装的 Wireshark 软件打开样例数据的输入和输出。

样例输入文件应该包含四个以太网帧，其中第一个以太网帧中 FCS 和 IP 的校验都是错误的，第二个以太网帧是 FCS 校验正确的 ARP 帧，第三个以太网帧仅 FCS 校验错误，第四个以太网帧仅 IP 校验错误。在页面中部点开当前帧的解析可以看到更多细节，包括是哪个校验错误和正确值等等，这可能对你有一定的帮助。

Wireshark 默认情况下可能不会检查 FCS 的正确性。对于这种情况，你可以在页面中部找到并右键点击 `Ethernet II` 所在的一行，在弹出的菜单中选择 `Protocol Preferences`，勾上选项中的 `Validate the Ethernet checksum if possible` 和 `Assume packets have FCS`，此时你应该可以在页面上部看到 FCS 错误的帧都被明显地标记出来。

需要注意的是，Wireshark 还会计算并显示 UDP 的 Checksum 正确与否。在本题中，我们不考虑 UDP 的 Checksum，因此你可以忽略这些错误。

数据链路层协议数据处理（layer2）

【题目背景】

路由器软件需要处理的最底层是数据链路层，这上面运行着一系列复杂的协议，比如各种神秘的滑动窗口协议。虽然芃芃对它们始终摸不着头脑，但他知道这一层最重要的协议就属 ARP 了。它起到了跨上下两层的桥梁作用，只有通过 ARP 查询，路由器才能知道究竟应该把 IP 分组发给哪一个邻居。你的路由器软件需要热情地回应每一个询问，才能正确地收到接下来的流量。

【题目描述】

你需要根据《学习手册》中的相关知识，处理数据链路层的协议数据：在给定的流量片段中，有一些是对你的 ARP 询问（也就是说，询问的 IP（TPA 字段）是你拥有的，请在 pcap 题面中找到你拥有的 IP 地址和与其对应的 MAC 地址），请你正确地回复它们。数据保证 FCS 校验正确的 ARP 请求中发送端的 MAC 地址（SHA 字段）与以太网帧中的发送端 MAC 地址一致。注意，ARP 请求中以太网帧的目标地址可能与你拥有的 MAC 地址不同（如 `FF:FF:FF:FF:FF:FF` 广播地址），这种情况下依然需要回应 ARP 请求。你需要构造正确的 ARP 协议数据，以及外层的以太网帧格式（包括头部和校验和，并注意以太网帧的长度要求），并将这些以太网帧帧按照应答顺序按照《学习手册》的要求依次写入 PCAP 格式的输出文件中。

在**下列所有题目（包括本题）**中，如果某个流量片段发生校验错误，则应当直接丢弃，片段的数据可能会出现不符合《学习手册》中的叙述的情况，所以不能有任何假设，不进行任何处理。

【输入格式】

从文件 ***layer2.in*** 中读入数据。

输入保证格式合法，其中包含不超过 n 个流量片段，总大小不超过 m 字节。你需要回复 n_1 个流量片段。

【输出格式】

输出到文件 ***layer2.out*** 中。

其将与答案文件进行**逐字节**对比。

【子任务】

测试点	n	m	$n_1 = n$
1	$= 10^2$	$= 5 \times 10^4$	是
2		$= 1.5 \times 10^5$	否
3	$= 10^3$	$= 1.5 \times 10^6$	
4	$= 10^4$	$= 1.5 \times 10^7$	
5	$= 10^5$	$= 1.5 \times 10^8$	

【样例 1 解释】

由于提供样例会大幅降低本题难度，故不提供样例。

网络层协议数据处理（layer3）

【题目背景】

百度对程序员来说最大的作用当然是测试网络的联通性，在处理完 ARP 包以后，芃芃兴奋地输入了 `ping baidu.com`，然而并没有得到任何结果。咨询过杰哥以后，芃芃懂得了 ping 工具的工作原理，原来它使用了 ICMP 协议，工作在比数据链路层更高的网络层。他还发现，网络中也存在许多给他发送的 ping 报文，看来的确有很多人在关心他。为了不让他们失望，你决定帮芃芃回复这些请求。

【题目描述】

你需要根据《学习手册》中的相关知识，处理网络层的协议数据：在给定的流量片段中，有一些是对你的 ICMP Echo Request 报文（也就是说，请求的 IP 和 MAC 地址都是是你拥有的，并为 `pcap` 题面中的对应关系），请你正确地回复它们。你需要构造正确的 ICMP Echo Reply 报文，以及外层的 IP 分组头、以太网帧头尾，并将这些帧按照应答顺序依次写入输出文件中。本题中只对**源地址与目的地址在同一个子网中的 `ICMP Echo Request` 报文**进行响应，而不响应其他情况（例如源地址与目的地址不在同一个网络中、ARP 请求）。

我们保证，所有 `ICMP Echo Request` 报文在数据链路层和网络层都是单播发送的（也就是说不用考虑广播地址等情况，广播地址的定义见《学习手册》），请注意正确填写回复报文在这两层的源、目标地址。如果接收到的报文中含有数据负载，则你回复的报文中也需要**原样携带**这些负载，但不要携带链路层帧中超过 IP 包尺寸后的多余数据（请忽略 Wireshark 对这部分数据的解析）。特别地，本题对输出数据中片段的时间戳有额外的要求，设 `ICMP Echo Request` 的时间戳为 $(\mathrm{sec}_0, \mathrm{us}_0)$，那么你回应的 `ICMP Echo Response` 的时间戳 $(\mathrm{sec}, \mathrm{us})$ 应该满足 $\mathrm{sec} \times 10^6 + \mathrm{us} = \mathrm{sec}_0 \times 10^6 + \mathrm{us}_0 + 1$ 并且 $0 \leq \mathrm{us} < 10^6$，数据保证 $\mathrm{us}_0 < 10^6, \mathrm{sec} < 2^{32}$，不保证上述加法不会溢出 32 位无符号整数。注意 Wireshark 界面上部显示的时间是相对于第一个以太网帧的时间戳的偏差。

【输入格式】

从文件 ***layer3.in*** 中读入数据。

输入包含不超过 n 个流量片段，总大小不超过 m 字节。

【输出格式】

输出到文件 ***layer3.out*** 中。

你的输出将与答案文件进行**逐字节**对比。请不要输出任何多余信息，以免导致不必要的失分。

【子任务】

本题的各子任务规模与上一题相同。

【样例 1】

见选手目录下的 ***layer3/layer31.in*** 与 ***layer3/layer31.ans***。

【样例 1 解释】

样例中输入数据是三个 `ICMP Echo Request`，输出数据是三个对应的 `ICMP Echo Reply`。

【提示】

请不要忘记前三题题面中与本题相关的内容，并注意 FCS 和 IP 校验和的计算和以太网帧的长度要求。

RIP 路由协议实现（routing）

【题目背景】

支持了链路层和网络层的协议以后，路由器立刻了收到无数的数据包。虽然芃芃被大量的数据淹没，不过镇定的他还是记起来教材上写的话：网络设备能够处理的报文有两种，分别是控制报文和数据报文。对于路由器来说，数据报文就是需要转发的内容；而控制报文则是用来管理它的转发流向（路由表）的。为了和其他真实的路由器进行交互，你需要实现简单的路由协议 RIP；而在建立了转发表之后，需要开始进行“抛热土豆”，正确地将其他主机的流量送到对应的下一台路由器。

【题目描述】

你需要根据《学习手册》中的相关知识，处理下列两类报文：

一类是 RIP 路由协议的控制报文，它被封装在 UDP 协议数据报中。你将只会收到 `RIP Response` 报文，且目标 IP 地址是 `224.0.0.9`，目标 MAC 地址为 `01:00:5E:00:00:09`，用于指示路由表的更新，你需要根据源 IP 地址来判断它与你拥有的哪一个 IP 在同一个子网中，从而得知路由表中对应的出端口编号。对于收到的每一条路由信息，你需要根据给定的规则（见《学习手册》）恰当维护自身的路由表，记录对于可达的每一个网络前缀的下一跳信息（包括 IP 和 MAC）。对于控制报文，总是无需进行回复。

另一类是数据报文，它将被封装在 IP 分组中，你需要对**目标 IP 地址非本机所拥有**的分组进行转发。如果路由表中能够查询到目标 IP 地址对应的下一跳，则你应当构造一个合法的 IP 分组，填写正确的源、目标信息，并更新相关字段（如 TTL、校验和）；如果无法查询到路由信息，则你应当构造一个正确的 `ICMP Destination Network Unreachable` 报文返回给源主机；如果分组的 TTL 减小到 0，则你应当构造一个正确的 `ICMP Time Exceeded` 报文返回给源主机，这两种错误信息都需要按照《学习手册》的要求填入相应的负载，从接收者 MAC 地址中推断出源地址 IP ，并且以太网帧中（接收者的 MAC 地址，发送者的 MAC 地址）为输入数据中对应数据报文的（发送者的 MAC 地址，接收者的 MAC 地址）；如果同时出现无法查询到路由信息并且 TTL 减到 0 的情况，应当构造一个 `ICMP Destination Network Unreachable`。无论何种情况，对于一个目标 IP 地址不属于本机的数据报文，你实现的路由器**总会产生**一个发出的以太网帧。

由于转发报文需要填写下一跳的 MAC 地址，而我们不提供主动的 ARP 查询接口；在本小题中，你需要并仅从 RIP 报文的**以太网帧头**中获得相应目的路由器的 MAC 地址，以**转发前最后一次获得到的为准** 。

我们保证，在此问题给出的正常流量片段（也就是通过了校验的片段）中，包含且只包含上述两类报文；并且对于所有需要转发的包，你都能通过上述方法从 RIP 报文中获得对应目的路由器的 MAC 地址。和上题类似，Wireshark 会显示 UDP 校验值错误，忽略即可。

【输入格式】

从文件 ***routing.in*** 中读入数据。

输入包含不超过 n 个流量片段，总大小不超过 m 字节。根据输入，你需要构造的路由表的项目不超过 k 条，需要查询路由表进行转发的报文占总报文的数量约为 $p_{query}\%$。

【输出格式】

输出到文件 ***routing.out*** 中。

你的输出将与答案文件进行逐字节对比。请不要输出任何多余信息，以免导致不必要的失分。

【子任务】

<table>
<tr><th>测试点</th><th>n</th><th>m</th><th>k</th><th>p_{query}</th></tr>
<tr><td>1</td><td>$= 10^2$</td><td>$= 1.5 \times 10^5$</td><td>$= 10$</td><td rowspan="2">$= 50\%$</td></tr>
<tr><td>2</td><td rowspan="2">$= 10^3$</td><td rowspan="2">$= 1.5 \times 10^6$</td><td rowspan="2">$= 10^2$</td></tr>
<tr><td>3</td><td>$= 95\%$</td></tr>
<tr><td>4</td><td rowspan="2">$= 10^4$</td><td rowspan="2">$= 1.5 \times 10^7$</td><td rowspan="2">$= 10^3$</td><td>$= 50\%$</td></tr>
<tr><td>5</td><td rowspan="5">$= 95\%$</td></tr>
<tr><td>6</td><td rowspan="2">$= 10^5$</td><td rowspan="2">$= 2 \times 10^7$</td><td rowspan="2">$= 10^4$</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td><td rowspan="3">$= 10^6$</td><td rowspan="3">$= 2 \times 10^8$</td><td rowspan="3">$= 10^5$</td></tr>
<tr><td>9</td></tr>
<tr><td>10</td></tr>
</table>

【样例 1】

见选手目录下的 ***routing/routing1.in*** 与 ***routing/routing1.ans***。

【样例 1 解释】

样例输入有十个以太网帧，包括八个 RIP 包和两个需要转发的 IP 包。

对于前四个 `RIP Response`，可以得到如下的路由表：

收到第一个 Response 后：

```
104.0.0.0/6 via 10.2.7.2 if 7 metric 7
13.115.192.0/10 via 10.2.7.2 if 7 metric 9
224.0.0.0/3 via 10.2.7.2 if metric 15
```

收到第二个 Response 后：

```
104.0.0.0/6 via 10.2.7.2 if 7 metric 7
13.115.192.0/10 via 10.2.7.2 if 7 metric 9
```

收到第三个 Response 后：

```
104.0.0.0/6 via 10.2.7.2 if 7 metric 7
13.115.192.0/18 via 10.2.7.2 if 7 metric 9
88.128.0.0/10 via 10.2.6.2 if 6 metric 13
```

收到第四个 Response 后：

```
104.0.0.0/6 via 10.2.7.2 if 7 metric 7
88.128.0.0/10 via 10.2.6.2 if 6 metric 13
```

接着是两个 UDP 包，对于 `77.147.142.166` 这个地址，在路由表中没有对应的项，于是回应了 `ICMP Destination unreachable` 。对于 `107.28.70.129` 地址，它在 `104.0.0.0/6` 范围中，于是 TTL 减一并且转发到 `10.2.7.2` ，它的 MAC 地址可以从第一个 `RIP Response` 得到。

收到第五个 Response 后：

```
104.0.0.0/6 via 10.2.7.2 if 7 metric 7
88.128.0.0/10 via 10.2.6.2 if 6 metric 13
130.127.0.0/17 via 10.2.4.2 if 4 metric 3
```

收到第六个 Response 后：

```
130.127.0.0/17 via 10.2.4.2 if 4 metric 3
```

收到第七个 Response 后：

```
130.127.0.0/17 via 10.2.4.2 if 4 metric 3
160.245.176.0/20 via 10.2.5.2 if 5 metric 9
```

收到第八个 Response 后：

```
130.127.0.0/17 via 10.2.4.2 if 4 metric 3
160.245.176.0/20 via 10.2.5.2 if 5 metric 9
90.32.0.0/15 via 10.2.5.2 if 5 metric 8
```

【提示】

由于每个数据报文的转发都需要查询路由表，因此需要用高效的数据结构存储路由表中的关键信息，同时也要注意空间的使用效率。关于算法的高效实现，你可以参考《学习手册》附带的相关论文和资料。

在转发 IP 分组时，分组头内很少的字段会被修改。因此 IP 校验值也未必需要重新计算，很多情况下可以只进行最小限度的改动。但如果你想进行任何形式的简化，务必保证你的操作与我们叙述的操作是**完全等价**的。直观的想法往往会遗漏一些边界情况，请一定注意。

期待你的声音（feedback）

这是一道提交答案题。

【题目背景】

经过千辛万苦的优化，芃芃终于让自己的路由器跑得和隔壁班的 e-n 同学一样快了。现在，最激动人心的时候就到了：他需要把自己路由器各个部件组合起来，放到真实的网络环境中进行测试。让人高兴的是，它成功经受了考验，芃芃也在《计算机网络原理》课程中取得了 A+ 的好成绩。这一切，都要感谢你的帮助。恭喜你通过自己的努力，帮助芃芃造出了年轻人的第一台路由器。

【题目描述】

当你看到这里的时候，这一题也已经到了尾声。或许你会奇怪，为什么会有这样的一题“大模拟”，来自一个你们或许从未接触过的领域，充斥着新名词和奇奇怪怪的细节。诚然，算法与数据结构是计算机科学中极其重要的一部分，但如果不真实应用到工程中，它们只能是论文中的神话。这一点在网络的世界中，得到了非常好的体现：无数从规模到技术都不同的网络，在有限的几种协议与算法的协调下，最终成功连接了整个世界。我们命制这一题，旨在于让你体会，前人的智慧如何被真正地应用于现实世界中，看似简单的设计思想又是如何给世界带来了巨大的改变。

事实上，这一题的灵感来自于真实存在的《计算机网络原理》课程的大实验项目，最终目标是让每个人都写出一个路由器。在清华大学计算机系，这样的课程还有很多：你可以亲手造一个 CPU、写一个操作系统、做一个路由器、设计一个编译器。你甚至可以完成它们完整地结合起来的壮举：在自己造的 CPU 上运行自己写的操作系统，在自己写的操作系统上运行自己编写的路由器和编译器，用编译器再完成对操作系统的编译。它们无疑都是艰巨的挑战，但也会带来巨大的收获。我们相信，只有亲自动手，才能从根本上理解它们的工作原理，才是学得最快、最好、最透彻的方式。

这一题与传统的 OI 题截然不同，对出题人、对选手来说都是一次全新的探索。如果愿意，希望你能够留下你的一些想法（使用纯文本格式书写并作为答案文件提交即可）。无论是建议还是批评、赞同或者吐槽，我们都会认真对待，并用来改进我们今后的工作。当然，不管你提交与否，也不管你提交什么内容，都**不会以任何形式**影响你的最终成绩。

无论这题做得是否顺利，无论你之后去向何方，我们都诚挚地希望你能够保持对计算机科学的热情。祝你成功，我们在清华等你来！

【输入格式】

本题没有任何输入。