南京航空航天大学

计算机科学与技术学院/人工智能学院

计算机网络实验设计报告

课　　题：三种交换方式的模拟

姓　　名：

学　　号：

班　　级：

时　　间：2019/4/15

1 模拟实验的内容和要求

本实验要求学生掌握利用 Socket 编程的能力，并可以使用 Socket 编程来模拟出三种交换方式的工作原理，以增强对三种交换方式的理解。

网络拓扑结构如下所示（其中的设备都以进程代替，数据链路通过 Socket 来模拟）：

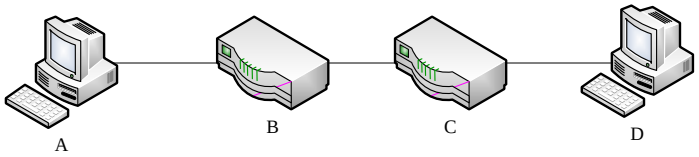

图 2 网络拓扑图

实验要求如下：

- 系统至少有 2 个主机（A 和 D）和 2 个交换设备（B 和 C，可以根据交换技术分别充当电话交换机、报文交换机、分组交换机），均使用进程来代替，它们之间的连接使用 Socket 进行模拟。
- 发送内容要足够大（比如 1M～10M，可以事先选择一个文件），并且假定已经知道目的方（不需要查表就可以进行发送方向的选择）。
- 对于电路交换，要能够体现出连接建立的过程（建立连接-发送数据-拆除连接）。
- 对于分组交换，分组数据的大小可以通过界面让人工选择（1000、1500、2000）。
- 报文标识在 0~255 中随机生成一个数字。
- 采用图形界面，查看处理的过程。
- 统计出整个过程的时延。

2 实验分析和设计

下面以分组交换方式为例，首先进行分析和设计，然后给出实现的部分描述。

2.1 用例图

一、发送端（A）用例图

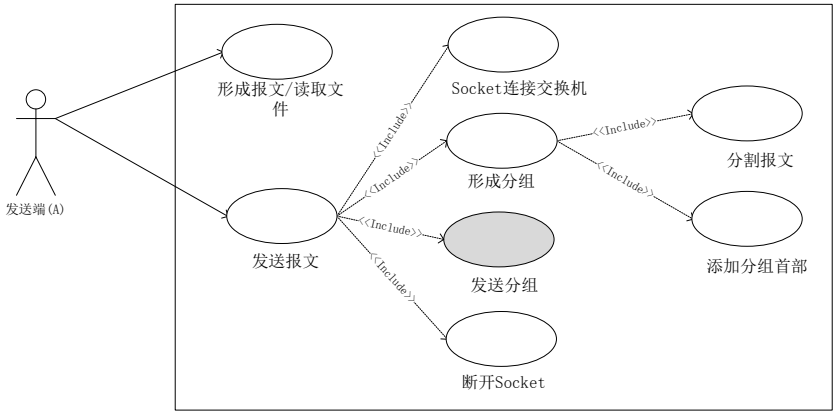

图 3 发送端用例图

对于发送端（A），以发送分组为例，其用例描述如表 1 所示。

表 1 发送分组用例描述

名称	分组交换仿真实验之发送分组。
标识	UC010203
描述	程序接收/形成一个独立的分组，利用 Socket 将分组发送出去。
前提	与分组交换机建立了连接，并且接收到一个独立分组。
结果	分组发送成功，或者失败。
注意	该方法可能会被循环调用，完成一个报文所包含的全部分组的发送。

由于是模仿实验，所以进行数据传送的时候，实际连接交换机的操作是使用 Socket 进行模拟的。如果采用流模式的 Socket 的话，这样会有建立 Socket 连接的过程，与分组交换模式相对比看上去是多余的。但实际上，真正的主机向交换机发送数据时，也是需要有握手/同步信号的交互过程的。这里可以把建立 Socket 连接的过程考虑成握手的过程。当然，如果采用用户数据报模式的话，则不用建立连接。下面考虑采用流模式的 Socket 通信。

针对分组交换，这种连接可以根据不同设计有不同的实现方法：

1) 整个过程只连接交换机一次，后面的分组全部通过该 Socket 进行发送。
2) 针对每一次的分组发送，都建立一次 Socket 连接，即对于一个报文被拆成 n 个分组的情况，使用 Socket 连接交换机 n 次，每次只发一个分组。

假设：为了仿真过程更加真实，建议采用第二种方法，下面同样处理。

二、分组交换机用例图

模拟分组交换机（B 或 C），需要处理的第一个工作就是对收到分组进行缓存。如同“特别说明”一章所述，强烈建议采用多线程技术来进行分组的接收/处理。

交换机的第二个工作是处理分组并继续向目的主机方向进行转发。在此强烈建议采用另外一个独立的线程来处理分组。

真实的分组处理过程应该包括：

(1) 从缓存中抓取一个分组；
(2) 获取并分析分组的首部；
(3) 找出首部中的目的地址；
(4) 从交换表中查出分组发送方向/接口；
(5) 从指定的发送方向/接口发送分组。

但是因为这里是软件模拟，不必要完成全部的工作，只需要处理步骤（1）和（5）即可。用例图如下所示。

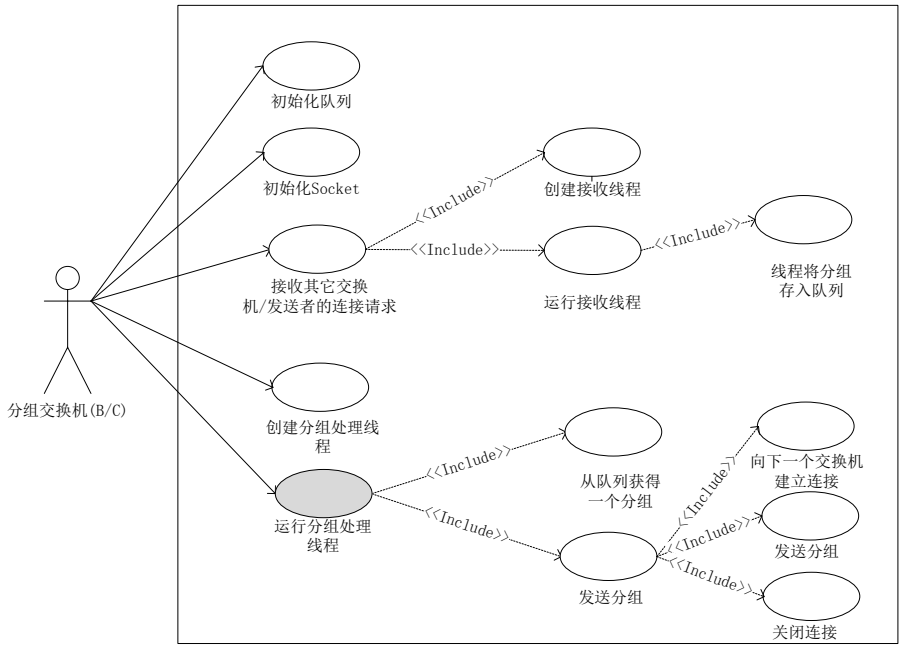

图 4 交换机用例图

队列的实现，可以很多种，可以参考数据结构课程相关内容，这里不再赘述。

以运行分组处理线程为例，其用例描述如表 2 所示。

表 2 交换机运行分组处理线程用例描述

名称	分组交换仿真实验之运行分组处理线程。
标识	UC0205
描述	循环执行（可以是死循环），每次从队列中获取队首的一个分组，向下一个分组交换机 C/接收端 D 建立 Socket 连接，并且发送分组，最后关闭 Socket 连接。
前提	队列中存在分组。
结果	分组发送成功，或者失败。
注意	线程一直执行，直到用户关闭整个程序。

三、接收端（D）用例图

模拟的接收端（D）需要处理的一个重要工作也是对收到的分组进行缓存，但在这里，缓存的目的和交换机有所不同，是为了对报文进行组合恢复。

同样，这里也建议采用多线程技术来处理，每收到一个 Socket 连接请求，就启动一个接收线程来进行处理，后续可以立即转入下一个连接请求的处理。

用例图如下所示。

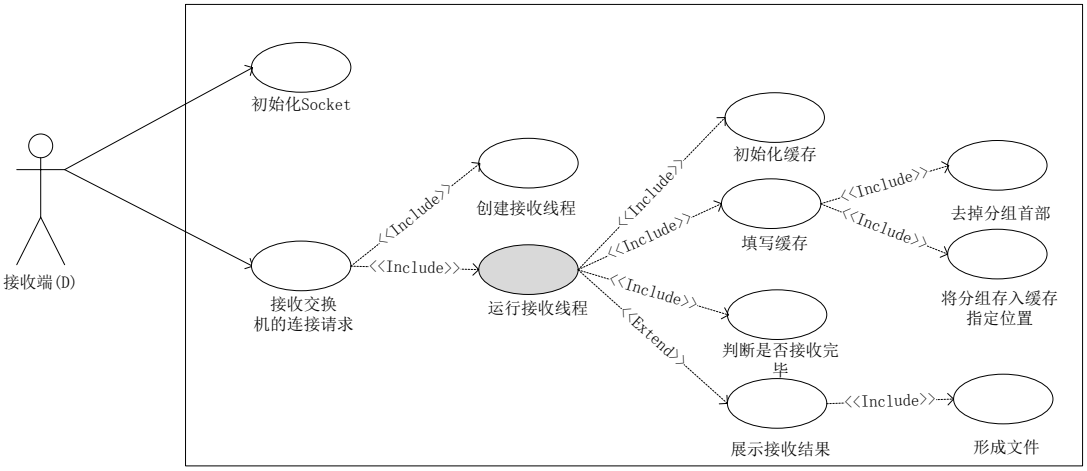

图 5 接收端用例图

注意，这里的初始化缓存应该是根据报文大小来进行申请的。在模拟分组交换的试验中，不建议使用文件追加的方式，因为实际的分组交换过程中，分组到达可能是乱序的，不是按照顺序到达的。

以运行分组接收线程为例，其用例描述如表 3 所示。

表 3 接收端运行分组接收线程用例描述

名称	分组交换仿真实验之运行分组接收线程。
标识	UC030202
描述	接收一个分组； 如果是报文的第一个分组则申请缓冲区； 将分组写入缓存的指定位置； 判断是否是需要接收的最后一个分组，如果是则把缓存数据写入文件，并展示接收结果。
前提	接收一个分组。
结果	把分组数据填充进入缓存。
注意	线程执行完毕后自动消亡。

2.2 时序图

在分组交换模式的实验中，针对一个报文的交换过程（注意不是分组），时序图如图 6 所示。

用户发送一个报文（从文件中读取），调用发送端 A 的发送功能，A 将报文拆分成多个分组后，循环（Loop）发送所有分组给 B。B、C 分别执行存储转发的过程，将每一个分组分别向 D 方向进行转发。D 收到分组后进行组合。

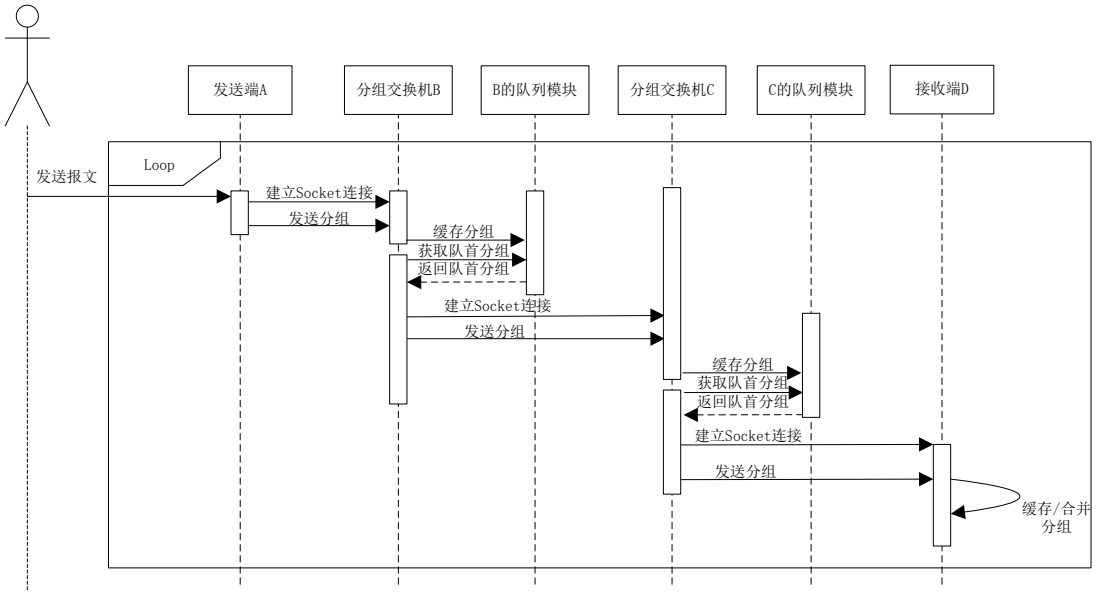

图 6 分组交换实验时序图

2.3 部署图

对于我们的模拟实验，可以部署在多台电脑上，也可以部署在一台电脑上。这里我们假设部署在多台电脑上，这样各个进程可以具有相同的端口号。如果因为条件所限，确实需要把所有进程都部署在一台电脑上，因为它们的 IP 地址相同，则需要为这些进程设定不同的端口号。

实验的部署图相对简单，如下图所示。

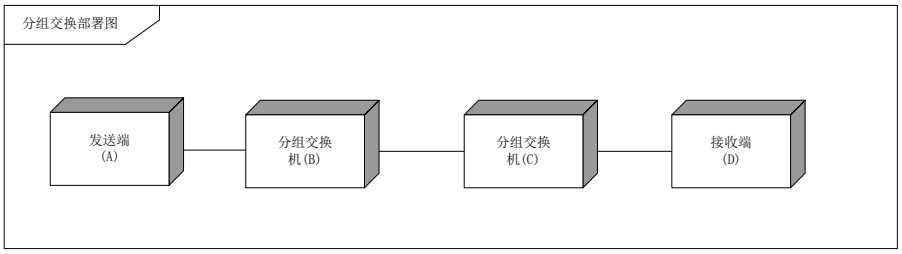

图 7 系统部署图

2.4 系统体系结构设计

针对分组交换模拟实验，系统可以分为 3 个部分，其体系结构如下所示。

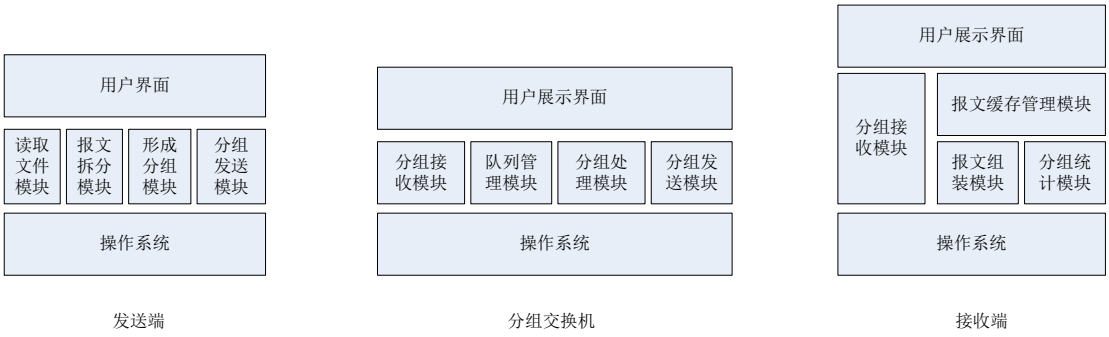

图 8 系统体系结构图

各个部分的功能浅显易懂，这里不再赘述。

2.5 报文格式设计

报文格式，放在本试验场景下，即分组格式。

分组格式如下图所示，当然，读者可以根据自己的设计需要进行修改。

接收端/D	发送端/A	报文标识	报文大小	分片总数	本分组在其中的位置	本分组长度	分组内容
1字节	1字节	1字节	4字节	2字节	2字节	2字节	

图 9 报文设计格式

其中：

- 接收端在本实验中固定为 D。
- 发送端固定为 A。
- 报文标识用于区别若干分组是否属于同一个报文。
- 报文大小指发送的整个报文大小，方便接收端进行缓存的申请，考虑到实验需要传输 1～10M 的数据，实际只需要 3 字节即可。
- 分片总数用于接收端对接收过程进行计数，以判断报文是否已经接收完毕。当片数接收计数器达到分片总数时，认为报文已经接收完毕，可以结束接收过程，进行提交（在本实验中即显示发送结果）。
- 本分组在其中的位置方便接收端把收到的分组保存进入缓存。
- 本分组长度主要是针对最后一个分组。

2.6 类图

一、发送端相关类

发送端的相关类如图 10 所示。

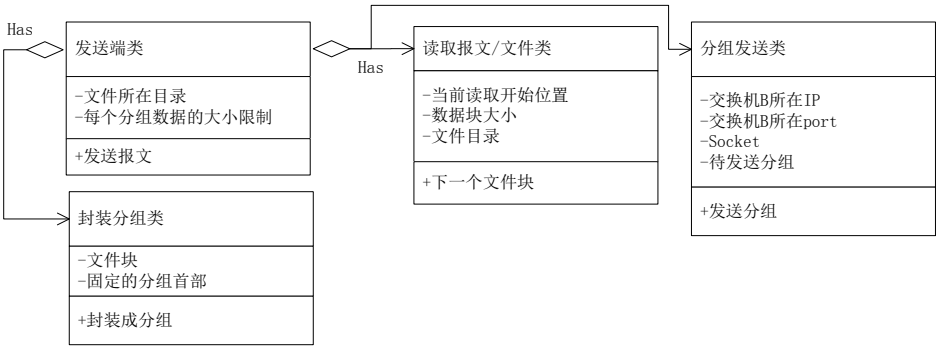

图 10 发送端相关类

其中，发送端类，接到发送的指令后，循环执行下面操作：调用读取报文/文件类循环读取报文数据块、使用封装成分组类进行封装、使用分组发送类发送分组，直到完成整个报文/文件的发送。

读取报文/文件类，既充当了体系结构中的读取文件模块，又充当了报文拆分模块，也就是每次只读取报文的一部分，不需要为了完全读取整个报文而设置较大的缓存。

在封装成分组类中，当前采用的方法是固定的分组首部，但是可以很容易地扩展成动态的分组首部，即根据不同的目的地址、控制信息等形成符合实际应用需求的分组首部。

针对分组发送类，因为本实验是固定的下一跳（即第一个分组交换机），所以可以在生成该类的时候，写死下一跳的网络信息，当然也可以作为参数进行接

受。

二、分组交换机相关类

分组交换机的相关类如图 11 所示。

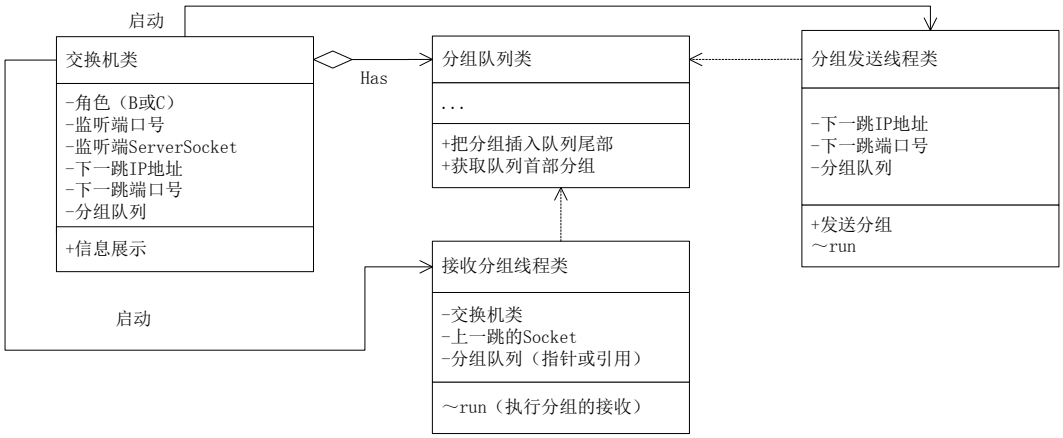

图 11 分组交换机相关类

其中，交换机类作为主类进行程序的运行。在准备完毕后，就进入等待状态，在监听端口等待上一跳进程（发送端 A 或交换机 B）发来的 Socket 连接请求。如果 Socket 连接请求到来，交换机类启动一个接收分组线程类进行分组的接收，并把自己（p，方便线程回填统计信息）、针对上一跳的接收端 Socket（s，用于接收数据）、分组队列（l，便于缓存分组）传给该线程。

接收分组线程类使用 s 读取分组，插入队列 l 的队尾，并把统计信息回填给交换机类 p。

分组发送线程类是一个单独的类，一直运行，如果能够从分组队列的队首获得分组，就发送给下一跳，否则略过。

分组队列类，具体参考数据结构，这里只给出了两个方法：把接收到的分组插入队列尾部、从队列首部获得一个分组。

三、接收端相关类

接收端的相关类如图 12 所示。

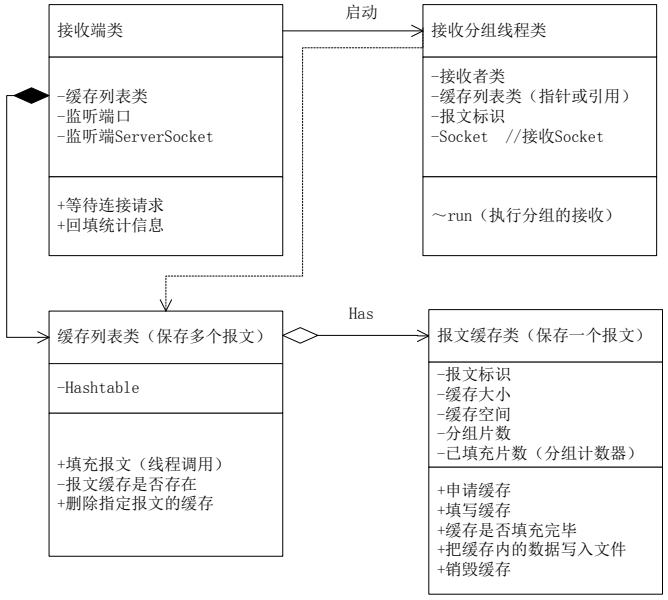

图 12 接收端类图

接收端类，是本进程的主类，它包含了所有报文（代表分组）的缓存（缓存使用缓存类进行实现）。

接收端准备完毕后，就进入等待状态，在监听端口等待其它进程（交换机 C）的 Socket 连接请求。如果 Socket 连接请求到来，接收端类启动一个接收分组线程类进行分组的接收，并把自己（p，方便线程回填统计信息）、针对 C 的接收端 Socket（s）、缓存 hash 列表（h）传给该线程。

接收分组线程类使用 s 读取分组，分离出分组首部，获得报文的标识，根据该标识，到 h 中查找报文的缓存，如果没有找到，则进行添加。其后，可以进行缓存的填充。直到报文缓存被填充完毕，然后调用 p 的回填方法显示统计信息。

3 实验实现

3.1 发送端处理流程

发送端的算法较为简单，可以不使用多线程即可满足要求。这里如前所述，假设每一次的分组发送，都建立一次 Socket 连接。

发送端的算法步骤如下：

(1) 发送端接受命令，对某文件进行处理，获得文件所在目录/文件夹、分组数据大小。
(2) 当前读取开始位置设为 0。剩余文件大小设为文件本身大小。
(3) 打开指定文件。
(4) 读取下一块数据。
(5) 将数据封装成分组。
(6) 发送分组给交换机 B。
(7) 文件是否读取完毕，如果完毕则结束，否则转向（4）。

发送端处理流程的活动图见图 13。

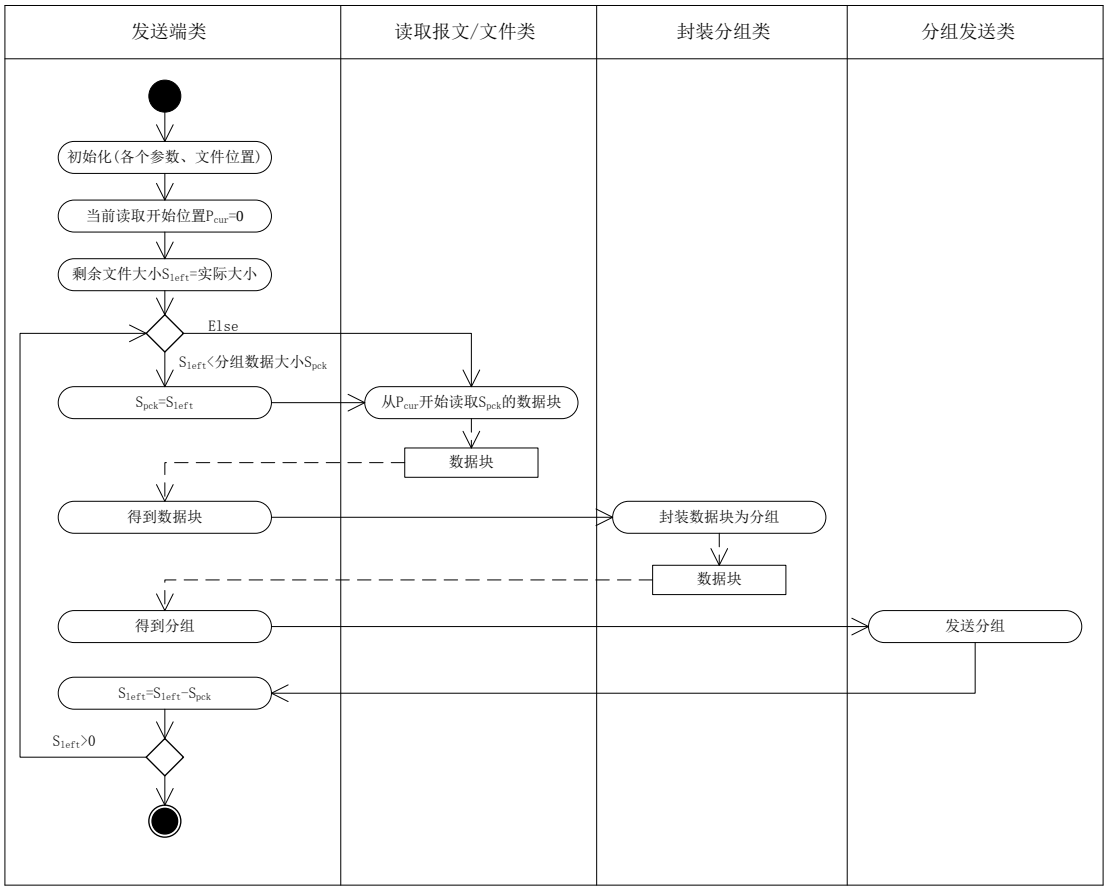

图 13 发送端活动图

3.2 交换机处理流程

交换机的算法分为两个部分：

- 交换机类通过多线程（接收分组线程类）来接收分组，每接收一个分组就启动一个线程，线程接收完毕即消亡。
- 交换机类产生一个独立的线程（分组发送线程类）用来转发分组，这个处理过程是一个死循环，即线程不会自动消亡。

接收分组线程算法步骤如下：

(1) 从 Socket 处获得输入缓冲。
(2) 从输入缓冲中读取分组。
(3) 检查分组队列是否已满。如果分组队列未满，则将分组保存入队列。
(4) 线程执行完毕，自己结束。

分组发送线程算法步骤如下：

(1) 分组队列中是否有分组，如果没有，则 sleep，本次循环结束。
(2) 从分组队列处获得队首分组。
(3) 删除分组队列的队首分组。
(4) 将分组发给下一跳。
(5) 转向（1）。

这两个处理过程较为简单，这里不再给出活动图。

3.3 接收端处理流程

在缓存报文的时候需要注意：申请的缓冲区应该属于进程，而不应该属于线程，因为线程执行完毕即自行消亡了，缓冲区将不复存在。如此做的前提是前面发送端的发送模式假设为每发一个分组就建立一个连接。

算法步骤如下：

(1) 接收一个分组，拆分成首部和数据两个部分。
(2) 查看报文标识，如果不存在该标识所对应的缓冲区，则表明这是报文的第一个分组。否则转到（4）。
(3) 获取首部写入的报文大小，申请缓冲区，设置分组计数器等于 0。
(4) 获取分组在报文中的位置。
(5) 将分组写入缓存的指定位置。
(6) 分组计数器加一。
(7) 检查分组计数器是否等于报文中分组的片数，如果等于则终止，否则转（1）。

接收端处理流程的活动图见图 14。

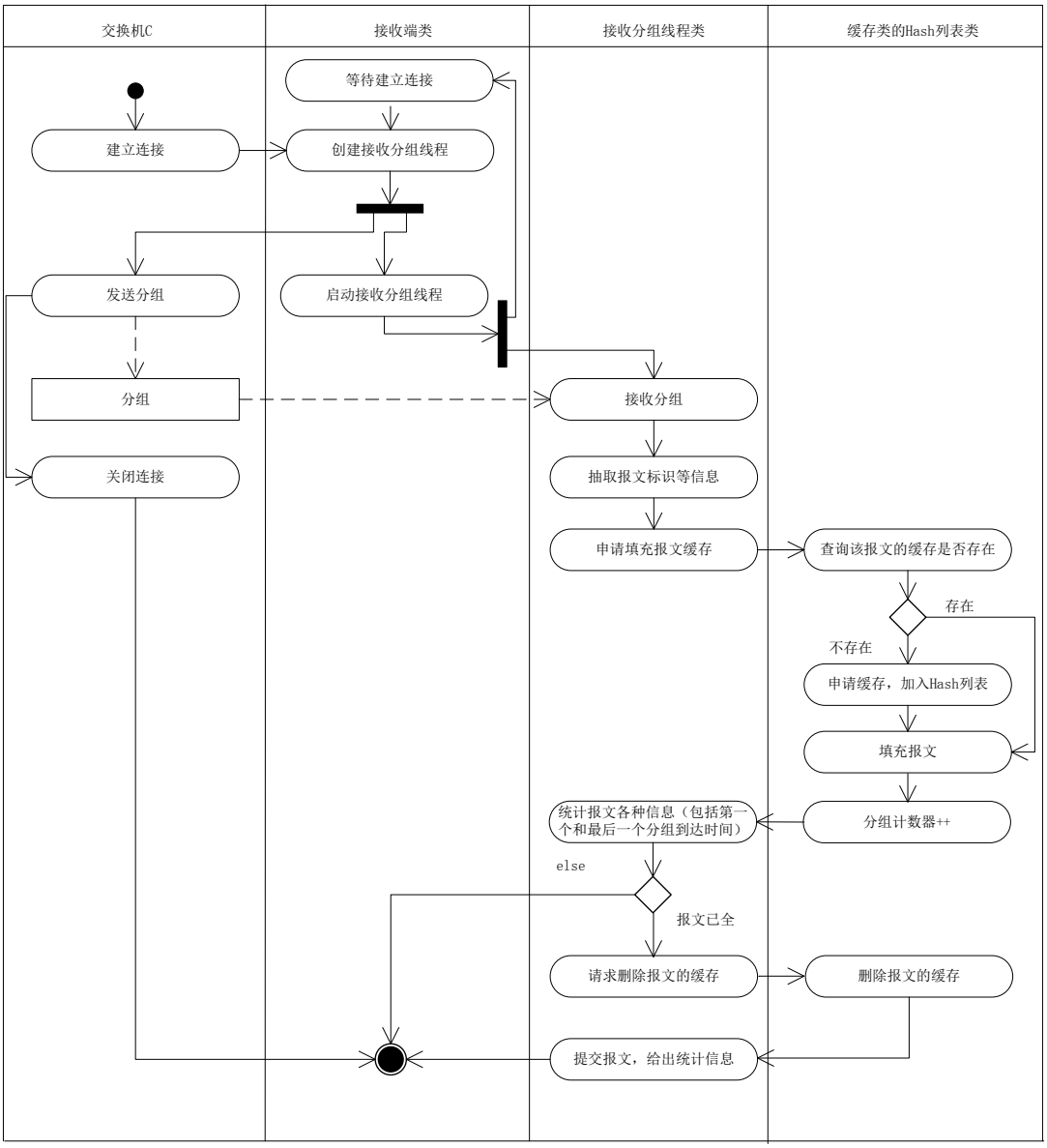

图 14 接收端活动图

3.4 界面样例

发送端 A 界面如下图所示。

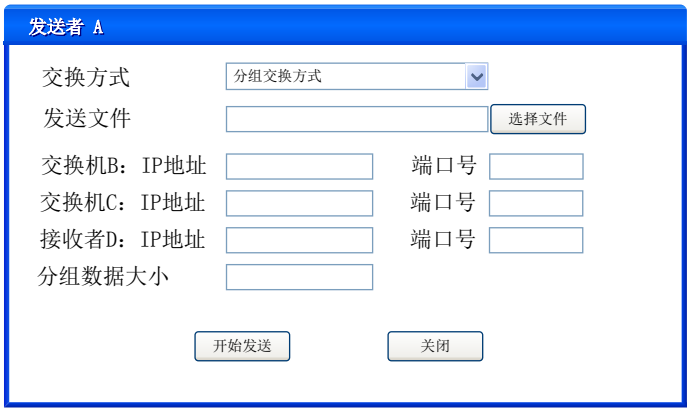

图 15 发送端界面样例

交换机界面如下图所示。

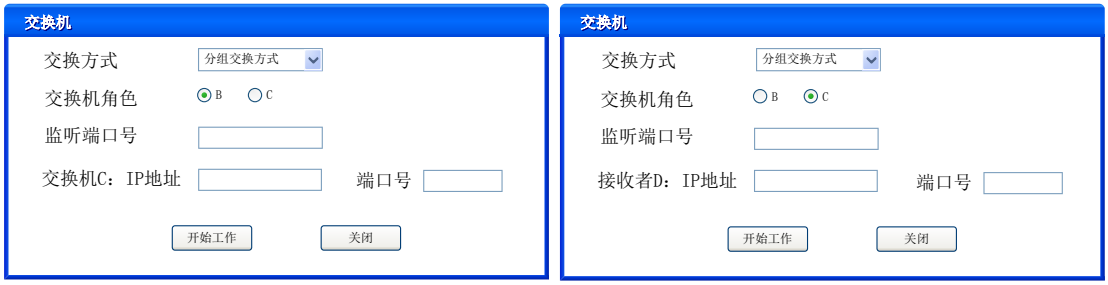

图 16 交换机界面样例

交换机程序应该能够做到，让用户选择交换机的角色（B 或者 C），从而让交换机知道自己的下一跳是谁。

接收端 D 界面如下图所示。

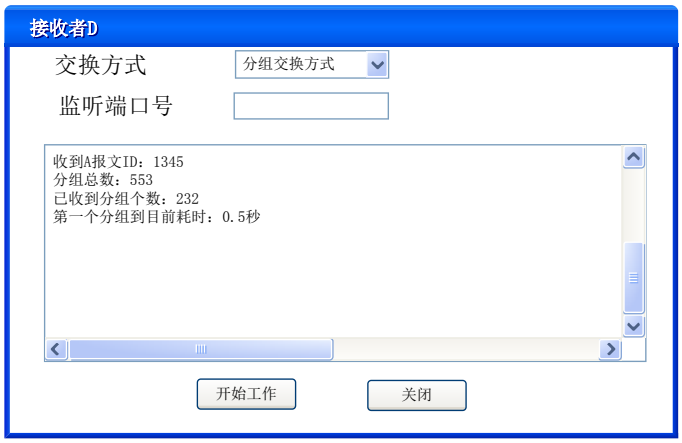

图 17 接收端界面样例