**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： OSPF路由算法模拟实验**

**姓 名： 王康**

**学 号： 162010312**

**班 级： 1620103**

**时 间： 2023.06.14**

**目录**

[1 实验分析和设计 3](#_Toc21822)

[1.1 需求分析 3](#_Toc652)

[1.2 概要设计 3](#_Toc11605)

[1.3 用例图和用例分析 3](#_Toc3084)

[1.4 时序图 5](#_Toc11548)

[1.5 部署图 6](#_Toc17153)

[1.6 报文格式设计 7](#_Toc17418)

[1.6.1 链路状态信息报文格式 7](#_Toc12023)

[1.6.2 握手报文格式 8](#_Toc16775)

[1.7 类图 8](#_Toc30596)

[1.7.1发送链路状态信息相关类 10](#_Toc5493)

[1.7.2洪泛相关类 10](#_Toc8475)

[1.7.3计算最短路径相关类 10](#_Toc7724)

[1.7.4公共类 11](#_Toc5534)

[1.8 系统体系结构设计 11](#_Toc14153)

[2 实验实现 12](#_Toc32366)

[2.1 洪泛法发送信息流程 12](#_Toc6142)

[2.2 计算最短路径算法流程 12](#_Toc25604)

[3 界面样例 14](#_Toc17598)

**1 实验分析和设计**

**1.1 需求分析**

路由算法有很多分类方法，一个很重要的分类方法是分为内部网关协议(Interior Gateway Protocol, IGP)和外部网关协议(External Gateway Protocol,  
EGP)。对于内部网关协议，计算机网络教材一般会介绍两个：一个是RIP，另一个是OSPF。前者是互联网中最先得到广泛使用的内部网关协议；然而，因为RIP具有一些局限性，所以目前OSPF得到了广泛使用。

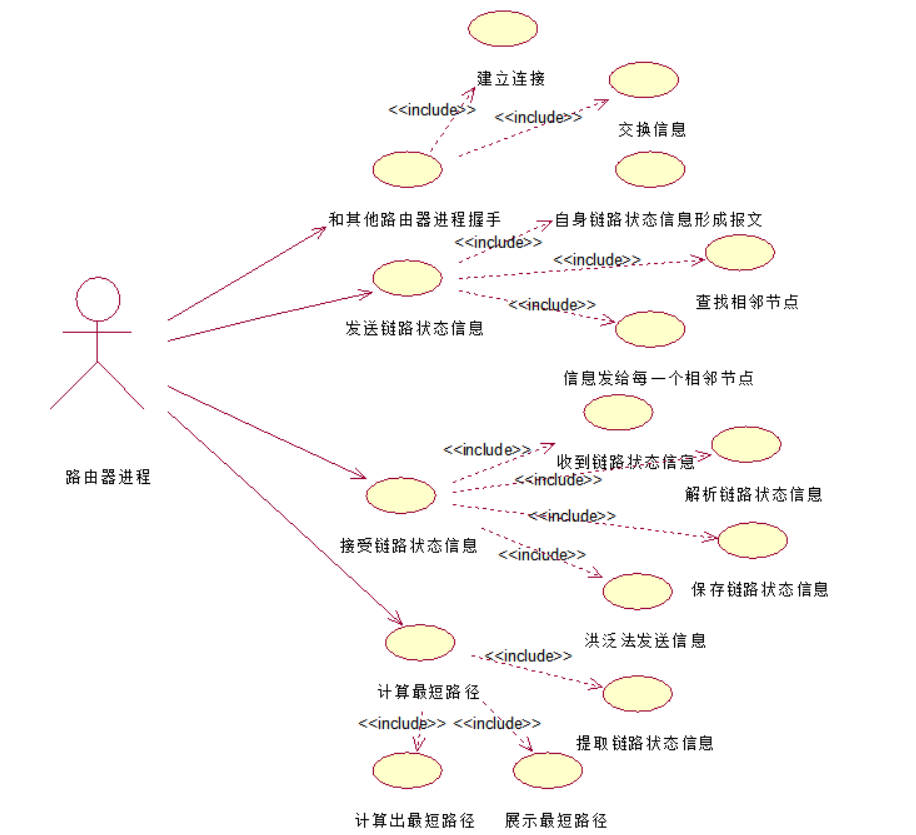
OSPF的全称为开放最短路径优先(Open Shortest Path First)协议。“开放”表明OSPF不是受某一家厂商所控制的，是公开发表的。“最短路径优先”是因为该协议使用了Dijkstra提出的最短路径算法。而且OSPF只是一个协议的名字，它并不意味着其他的路由选择协议不是最短路径优先的。  
 OSPF采用分布式的链路状态协议(link state protocol)。链路状态是指本路由器和哪些路由器相邻(即具有链路)以及该链路的度量(metric,即状态)是什么。  
 OSPF具有3个要点:  
 （1）每一个路由器都向本区域内的所有其他路由器发送链路信息，发送信息使用的方法是洪泛法，最终整个区域中所有的路由器都得到了这些信息。  
 （2）每一个路由器发送的信息是链路状态信息的集合，这些信息表示了自己与相邻路由器的连接情况。  
 （3）在一段时间内，只有当链路状态发生变化时，路由器才会使用洪泛法向所有路由器发送此信息。

因此，由于OSPF算法的先进性和有效性，并且其模拟实验的可实现性，对OSPF路由算法的模拟实验实现是有必要的。

**1.2 概要设计**

由于各路由器之间交换链路状态信息，因此利用所有路由器的链路状态信息能建立一个链路状态数据库。这个数据库实际上就是全网的拓扑结构图，它在全网范围内是一致的。这样,每一个路由器都知道全网共有多少个路由器，哪些路由器是相连的，相连路由器之间的代价是多少，等等。  
 在建立链路状态数据库的基础之上，OSPF算法就可以进行最短路径的计算了，从而构造出自己的路由表。当然，OSPF算法最终计算出来的结果是从自已到某个网络的路径，交换的链路状态信息也较为复杂。为了便于理解，下面关于算法的介绍和模拟实验都只考虑从一个路由器到另一个路由器的最短路径。  
 OSPF的链路状态数据库能较快地进行更新，使各个路由器能及时更新其路由表。并且该数据库具有一次收集、一次计算即可收敛的特点，使得OSPF本身具有良好的快速收敛性。相比之下，RIP需要多次收集、多次比对后才能够完成收敛。  
 本实验忽略OSPF分级的特性，只模拟一个区域内的工作情况。另外,OSPF采用IP分组进行工作，而本实验采用TCP(流式Socket工作模式)来模拟。

**1.3 用例图和用例分析**



**图1.1 路由器进程的用例图**

路由器进程有4个主要的工作:和相邻路由器进程建立物理连接，向相邻路由器进程传送链路状态信息，接收相邻路由器进程的链路状态信息并通过洪泛法继续传送，根据收集到的链路状态信息计算从自身到所有其他路由器进程的最短路径。  
 路由器进程的用例图如图1.1所示。  
 在实验过程中,路由器之间进行握手(这在实际情况中是不存在的,这里是为了实现实验而人为添加的过程，可以视为用物理线缆连接路由器)后，双方可以保留对方的Socket连接，也可以在交换完握手信息后就断开Socket连接。在本实验中，因为每一个路由器进程都必须具有ServerSocket，所以可以不保留对方的Socket,而且第二种方案在实现上更加方便,因此本实验采取第二种方案。

图1.1中接收链路状态信息用例和计算最短路径用例分别如表1.1和1.2所示。

**表1.1 接收链路状态信息用例描述**

|  |  |
| --- | --- |
| 名称 | 接收链路状态信息 |
| 描述 | 1.收到相邻节点X发送的链路状态信息报文。 2.解析报文。 3根据报文标识判断自己是否已经收到过该报文。如果收到过该报文,则抛弃该报文不予处理,结束。 4.获得链路状态信息。 5.更新链路状态数据库。 6.对于自己的每一个相邻节点，如果该节点不是报文的发出节点，则将报文发送给该节点 |
| 前提 | 和相邻路由器进行了握手过程 |
| 结果 | 更新链路状态数据库 继续发送链路状态信息 |
| 注意 | 路由器需要事先指定自己的相邻节点。 链路状态信息需要包含那些曾经连接过，但是现在断开的链路的信息，以方便接收者更器自己的链路状态数据库。 一定要注意非常重要的一点，用洪泛法发送报文的过程一定要有结束判断条件，否则报文会在网络上无休止地循环发送。本用例的判断条件是：解析该报文，判断自己是否已经收到过该报文，如果是，则抛弃该报文并结束。 发送链路状态信息时，只发送与自己相关的信息。 收到其他路由器进程的链路状态信息并对自己的链路状态数据库进行更新后,不要和自己已有的其他链路状态信息合并后再进行洪泛 |

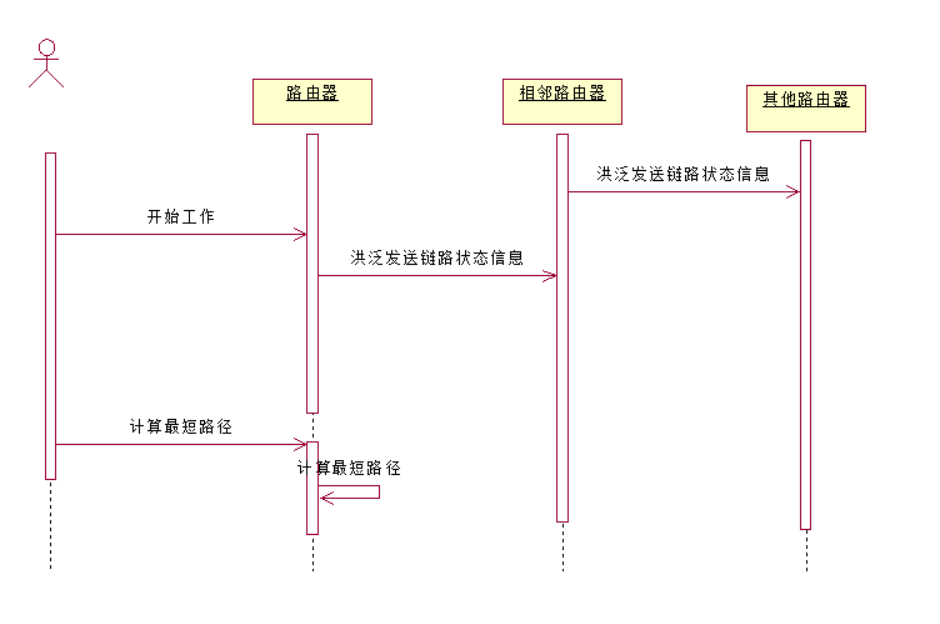
**表1.2 计算最短路径用例描述**

|  |  |
| --- | --- |
| 名称 | 计算最短路径 |
| 描述 | 1.获取路由节点的拓扑结构。 2.由Dijkstra算法计算最短路径。 3根据计算结果存入数组输出表格。 4.根据数组输出图像。 |
| 前提 | 建立了路由器之间的拓扑结构 |
| 结果 | 更新最短路径数组  输出结果并由此画出图像 |

由于用户单击按钮的过程有先后，所以必然有一些路由器进程会先得到其他进程发来的链路状态信息，而后才被用户操作以发送自己的链路状态信息，因此无法判断所有路由器进程发送的链路状态信息是否已经收集齐全。为此，本实验设定路由器进程开始计算的过程也是人工操作的。  
 路由器进程发送链路状态信息报文的过程并不复杂，时间也不是很长，所以不必借助于多线程技术。但是接收链路状态信息报文的过程需要不断监听其他路由器的连接请求，需要并行处理，所以应该采用多线程技术实现。

**1.4 时序图**

OSPF模拟实验的时序图如图1.2所示，总体过程是非常简单的，路由器进程的两个主要工作（利用洪泛法发送自己的链路状态信息和计算最短路径）都是由用户启动的，只有当收到相邻节点的链路状态信息时，才自动处理并向外继续洪泛。



**图1.2 OSPF模拟实验的时序图**

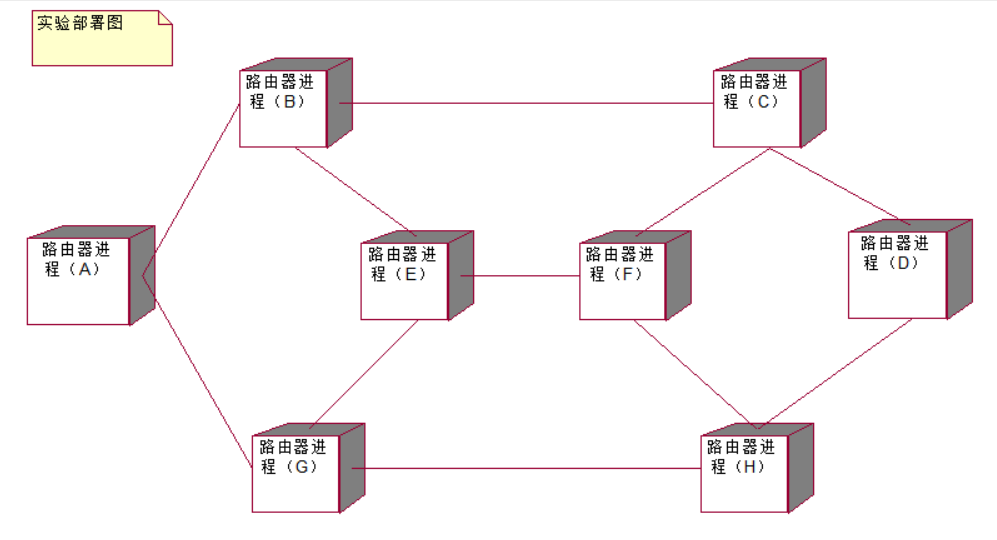
以上主要工作涉及的函数设计如表1.3所示：

**表1.3 主要函数设计**

|  |  |
| --- | --- |
| 进行握手进程 | SendHandShakeMessage() |
| 解析报文 | AnalysisMessage(ArrayList<Integer> message, Router r) |
| 洪泛发送链路状态信息 | OrganizeLinkStateMessage() |
| 计算最短路径 | Dijkstra() |
| 输出最短路径树 | ChartAfterDijkstra() |
| 输出最短路径 | PrintDijkstra() |

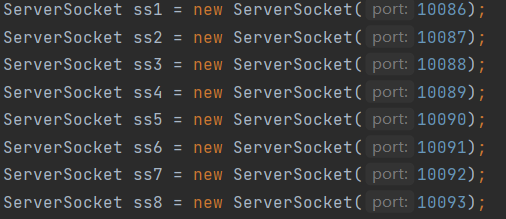
**1.5 部署图**

在本实验中，多个进程可以部署在多台计算机上，也可以部署在同一台计算机上。这里假设实验是部署在多台计算机上的，这样，各个进程可以具有随意的端口号。如果因为实验条件所限，确实需要把所有进程部署在同台计算机上，因为各个进程所需的IP地址相同，所以需要为这些进程设置不同的端口号，以防止冲突。  
 本实验的部署图如图1.3所示。其中的连线表示双方需要进行通信。



**图1.3 实验部署图**

路由器进程之间建立通信，通过调用java.net.ServerSocket，由如图1.4所示初始化连接，之后在需要接受和发送的地方调用thread.start方法创建进程。



**图1.4 socket进程初始化**

**1.6 报文格式设计**

本实验涉及的报文有两个:链路状态信息报文和握手报文。为了方便各个进程之间的通信处理，特定义了报文类型字段,其中包含的报文类型如表1.4所示。针对这些报文类型以及后面定义的报文格式，可以根据自己的设计需要进行修改和完善。

**表1.4 报文类型**

|  |  |  |
| --- | --- | --- |
| **类型号** | **类型** | **备注** |
| 0 | 链路状态信息报文 | 用于传输链路状态信息的报文 |
| 1 | 握手报文 | 完成路由器进程之间的初始化握手过程 |

**1.6.1 链路状态信息报文格式**

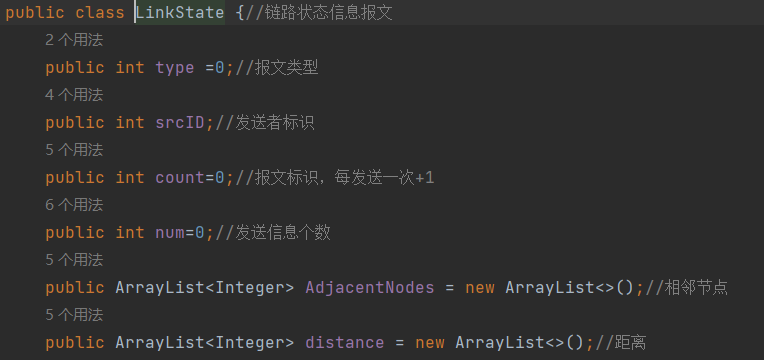
有了链路状态信息报文的相互交流，才能够实现路由器进程计算最短路径的目的。链路状态信息报文的格式如图1.5所示。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 报文  类型 | 发送者标识 | 报文  标识 | 信息个数 | 邻节点1标识 | 距离1 | 邻节点2标识 | 距离2 | ... |
| 1字节 | 1字节 | 1字节 | 1字节 | 1字节 | 1字节 | 1字节 | 1字节 | 1字节 |

**图1.5 链路状态信息报文格式**

报文类型固定为0。  
 发送者标识是最初发送本链路状态信息报文的路由器进程(下称源路由器进程)的标识。  
 报文标识是源路由器进程发出报文的“身份证”，可以用简单累加计数的方式加以实现，用于唯一地标识报文。  
 将发送者标识和报文标识结合，可以使得接收者对报文的判断(判断自己是否已经接收过该报文)变得非常简单。只有接收者判断出已经接收过某链路状态信息报文,才能终止该链路状态信息报文在网络中无休止的传送。  
 信息个数是后面的(相邻节点标识，距离)信息对的个数，每一个信息对代表了源路由器进程和相邻路由器进程之间的链路状态。  
 邻节点标识是和源路由器进程相邻的路由器进程的标识。  
 距离是源路由器进程和相邻路由器进程之间的距离度量，其值大于0，小于或等于100。如果距离为-1,则表示两个路由器进程曾经连接,但是后来断开了。增加一1这个值只是为了让用户显示的信息更加全面。

链路状态信息报文格式设计如图1.6所示：



**图1.6 链路状态信息报文类**

**1.6.2 握手报文格式**

握手报文的格式如图1.7所示，此报文是为了实现路由器进程之间交互初始信息的握手过程,在Socket连接建立后，路由器进程需互相发送此报文。

|  |  |  |
| --- | --- | --- |
| 报文类型 | 本路由器标识 | 距离 |
| 1字节 | 1字节 | 1字节 |

**图1.7 握手报文格式**

报文类型固定为1。  
 本路由器标识是指发出此报文的路由器进程的标识(A~H)。  
 距离是指路由器进程之间的距离度量，其值大于0，小于或等于100。如果距离为-1，则表示两个路由器进程曾经连接，但是后来断开了。

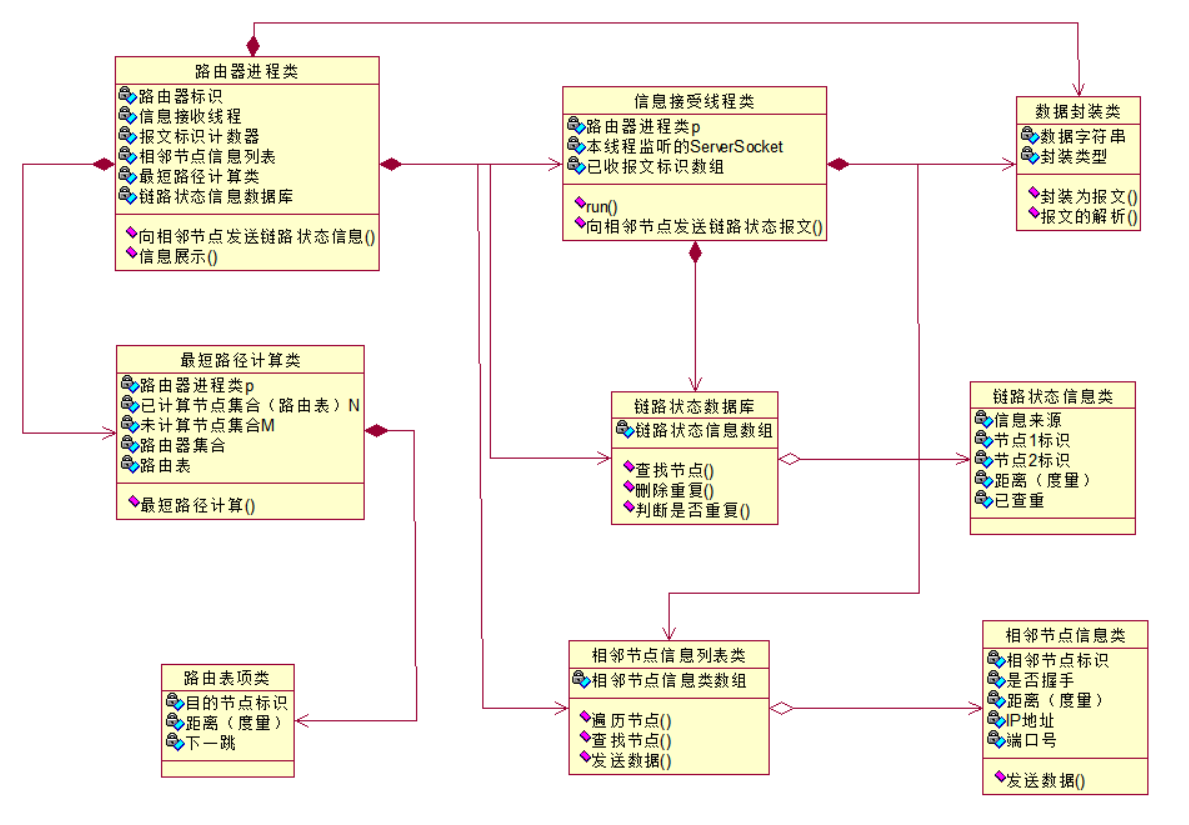
握手报文格式设计如图1.8所示：



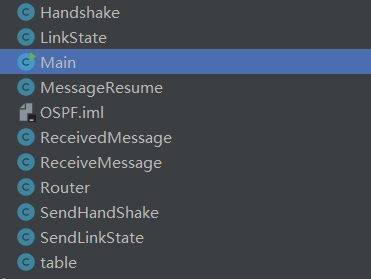
**图1.8 握手报文类**

**1.7 类图**

路由器进程的相关类图如图1.9所示。具体来说，这些类又可以按照功能分为4类。



**图1.9 路由器进程的相关类图**



**图1.10 程序代码设计的所有类**

由此类图具体实现到代码中，建立了如图1.10所示的类，表1.5解释了它们的功能。

**表1.5 代码设计的类的功能**

|  |  |
| --- | --- |
| Handshake | 握手报文类 |
| LinkState | 链路信息状态报文类 |
| MessageResume | 接收报文类 |
| ReceivedMessage | 已接收信息类 |
| ReceiveMessage | 接收信息类 |
| Router | 路由器类 |
| SendHandShake | 发送握手报文类 |
| SendLinkState | 发送链路状态信息类 |
| table | 界面类 |

**1.7.1发送链路状态信息相关类**

路由器进程类是主进程，也充当了多个类的纽带，并提供了信息展示的方法。  
 在本实验中，路由器进程还需要根据用户单击按钮的指令发送自身的链路状态信息给相邻路由器进程，该路由器首先查询自身的相邻节点信息列表，搜集自身所保存的、与相邻节点之间的链路状态信息，然后调用数据封装类将这些链路状态信息封装成报文,最后调用向相邻节点发送链路状态信息方法完成洪泛发送的第一步。而向相邻节点发送链路状态信息方法则通过相邻节点信息类的发送数据方法具体完成发送过程。  
 相邻节点信息列表类是在实验系统中各个路由器进程相互握手之后形成的，保存了路由器进程与相邻节点之间的链路状态信息(主要是距离度量,另外还添加了一些辅助通信的信息)。为了方便实现，当两个路由器进程断开连接(如实验要求那样，E、F路由器之间断开连接)后，重新握手建立新的信息列表之前，距离度量为-1。

**1.7.2洪泛相关类**

信息接收线程类是完成洪泛发送信息的主要类,该类收到其他路由器进程发来的Socket连接请求后,可以自己进行处理，也可以生成新的线程进行处理。因为业务并不复杂，所以本实验采用前一种方法。

信息接收线程类时刻监听相邻节点发送的报文,依据报文解析后得到报文标识等信息，判断是继续洪泛发送还是抛弃不理。判断的依据是该类中保存的已收报文标识数组(<发送者标识，最大报文标识数>的数组)。如果新接收的报文标识小于或等于最大报文标识数(前提是发送者标识相同)，则认为是已经收到的报文；否则认为是新的报文，此时需要更新已收报文标识数组。  
 如果信息接收线程类判断出需要将收到的链路状态信息报文继续洪泛发送，说明该报文中的信息对自身也是有用的，此时将解析出来的链路状态信息更新到路由器的链路状态数据库中(包括添加、删除等操作)。  
 注意:信息接收线程在利用洪泛法继续发送链路状态信息报文的过程中，需要排除来源方向的路由器进程。  
 链路状态数据库保存了整个网络拓扑中所有路由器与相邻节点之间的链路状态信息，这个信息较为简单，只包括了距离信息和起始节点。

**1.7.3计算最短路径相关类**

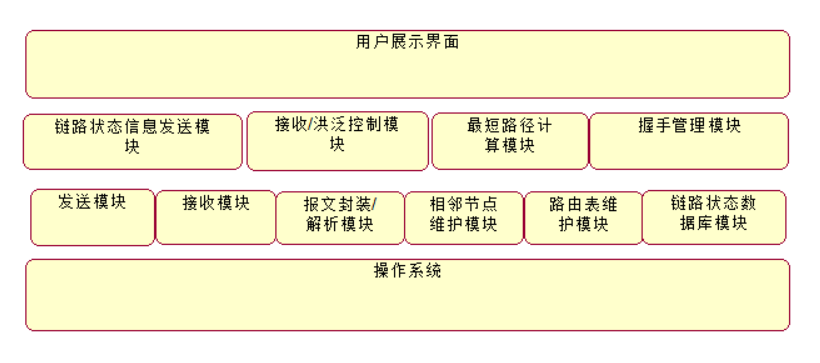
最短路径计算类负责在用户单击“开始计算”按钮后，依据OSPF算法进行最短路径的计算。计算的输入是链路状态数据库，输出是已计算节点集合N。  
 计算的第一步,需要对链路状态数据库中重复的信息进行清除，因为从X到Y的链路和从Y到X的链路实际上是同一条链路，在数据库中无须保存重复的链路信息。为此，链路状态数据库需要提供判断重复信息的方法。  
 为了方便计算，链路状态数据库还需要提供一个根据路由器标识查找链路的方法，这个查找的方法稍显特殊：无论是链路的节点1还是节点2,只要其中一个标识等于传进的路由器进程的标识，就算是找到链路了。  
 最短路径计算类中包含了两个数组：  
 **·**已计算节点的集合N,记录了那些已经计算出最短路径的节点的信息(包括距离、上一 跳等)。  
 **·**未计算节点集合M，记录了那些尚未计算出最短路径的节点的信息(包括距离、上一 跳)。该集合为空时，表示计算过程的结束。  
 这两个节点的集合是最短路径计算类在计算过程中必需的。在M中的节点，如果与源节点的距离是无穷大(可能是根本不相连，也可能是暂时不相连)则设距离为120，上一跳为“-”。

**1.7.4公共类**

数据封装类完成链路状态信息报文、握手报文的封装和解析。

**1.8 系统体系结构设计**

针对OSPF模拟实验系统只需要一个路由器角色即可,其体系结构如图1.11所示，很显然，在实验中，每一个路由器进程既是链路状态信息的发送者，也是链路状态信息的接收者和使用者。



**图1.11 系统体系结构图**

接收模块的主要工作是完成基本的数据接收功能，负责读取网络上传送到本进程的二进制数据流。而发送模块的主要工作是将上层的二进制数据流通过网络发送出去。

报文封装模块的主要工作是将相关信息封装成报文。

报文解析模块的主要工作是对报文进行解析，得到相关信息，以方便进行后续处理，还可以获得些必要的控制信息。

路由器进程的工作是由用户启动的，具体完成则是由链路状态信息发送模块和接收/洪泛控制模块完成的，前者负责发送路由器进程自己的链路状态信息，后者接收其他路由器进程的链路状态信息并进行后续的洪泛发送。通过这两个模块，可以实现在所有路由器进程之间交流链路状态信息的目的。

在各路由器进程交流了链路状态信息后，由用户启动最短路径计算模块,完成最短路径计算的工作。

相邻节点维护模块和链路状态数据库模块是辅助模块，前者帮助完成链路状态信息的洪泛发送，后者负责完成链路状态信息的保存和维护。

路由表维护实际上是对最终计算结果的维护。

握手管理模块负责在系统运行的初期完成各个路由器进程之间的握手过程，以模拟物理连接的效果。

**2 实验实现**

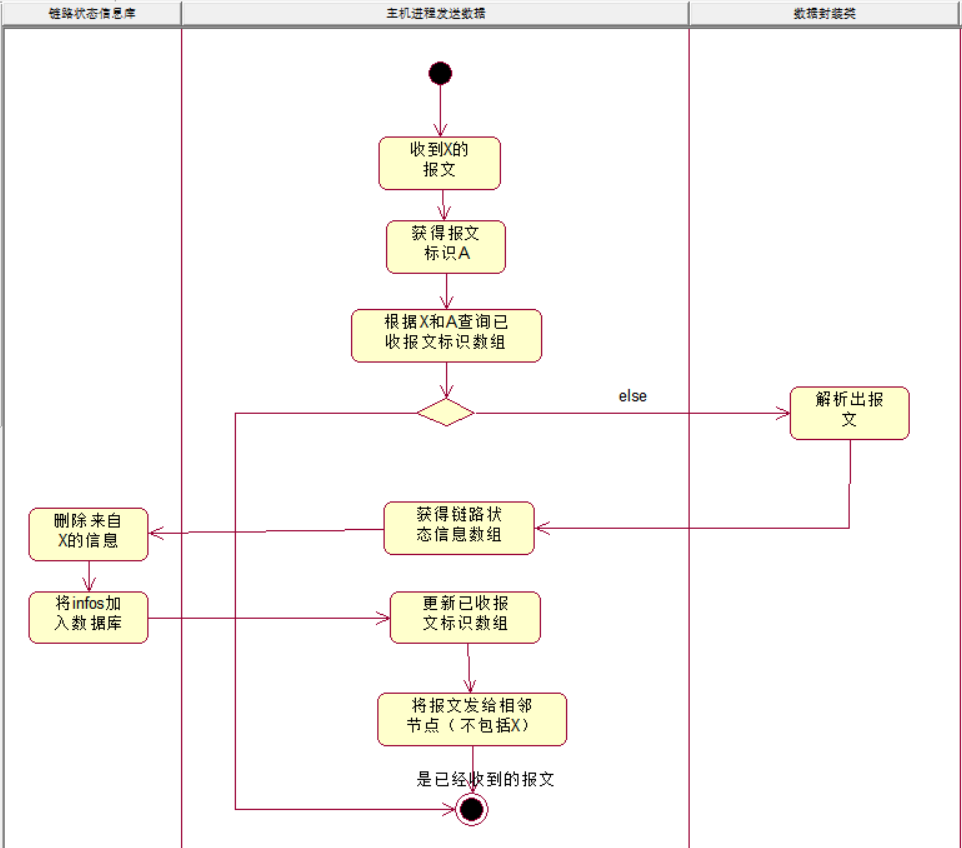
**2.1 洪泛法发送信息流程**

洪泛法发送链路状态信息是一个非常重要的功能，是实现OSPF的基础之一。不仅要实现类似于广播的效果,还要避免报文在网络中不断发送。

当一个路由器进程收到其他路由器进程发来的链路状态信息后，处理算法的步骤如下：

(1)收到相邻节点x发送的链路状态信息报文。  
 (2)解析报文标识。  
 (3)根据发送者标识和报文标识判断自己是否已经收到过该报文。如果收到过该报文，则抛弃该报文,转(9)；否则转(4)。  
 (4)解析报文，获得链路状态信息。  
 (5)在链路状态数据库中，根据链路状态信息类的信息来源删除所有来自X的链路状态信息。  
 (6)将新的链路状态信息插人链路状态数据库。  
 (7)修改已收报文标识数组中发送者的最大报文标识数。  
 (8)针对自己的每一个相邻节点Y,如果Y不是X，则将报文发送给Y。

(9)调用界面展示函数进行展示。  
  
 洪泛法发送信息的活动图见图2.1。



**图2.1 洪泛法发送信息的活动图**

**2.2 计算最短路径算法流程**

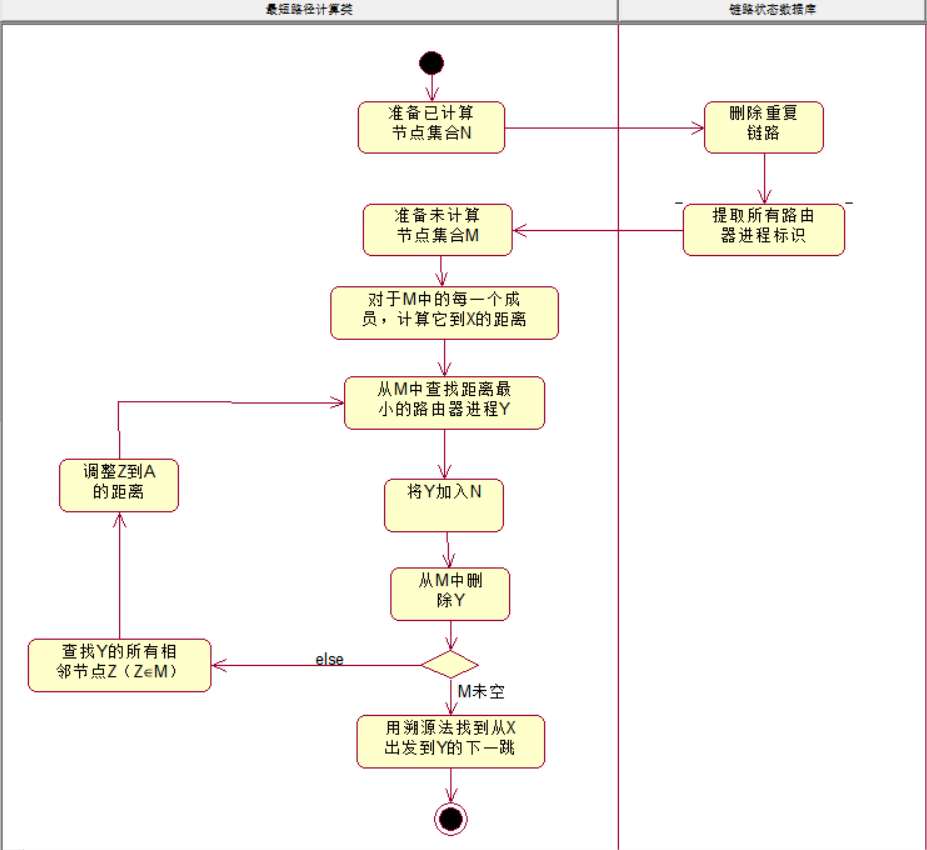
用户在路由器进程X的界面上单击“开始计算”按钮,X进程即启动计算最短路径的算法，对链路状态数据库中的信息进行处理，计算出从自己到其他任意一个路由器进程的最短路径以及下一跳路由器进程的标识，从而构成简单的路由表。

计算最短路径算法的步骤如下:  
 (1)准备已计算节点集合N，初始成员只有本路由器进程的标识X。

(2)清空路由表项类。  
 (3)对链路状态数据库进行清理，删除重复信息，即I→J和J→I两条链路中只需要保留一条即可。具体做法是:对于每一条链路(I→J,即节点1为I,节点2为J),标记其为已查重,遍历所有未查重的其他链路，如果存在链路J→I,则删除链路J→I。

(4)提取所有链路的起止路由器进程的标识，将它们加入到路由器进程集合中。这个过程需要避免重复。  
 (5)准备未计算节点集合M,初始成员包括路由器进程集合中除了X以外的其他所有路由器进程的标识。  
 (6)对于M中每一个成员Y,计算Y到X的距离L:如果Y和X相邻，则L(X,Y)为X、Y的距离；否则L(X，Y)为无穷大(设定为数值120)。  
 注意:如果距离是一1，则也要令其等于120。

(7)从M中查找距离最小的路由器进程Y。  
 (8)将Y加人N(记录上一跳是哪- -个路由器进程)。从M中删除Y。  
 (9)如果M为空,转向(12)；否则转向(10)。  
 (10)在链路状态数据库中查找Y的所有相邻节点Z(Z∈M)。  
 (11)对每一个Z，调整Z到X的距离：如果L(X,Z)>L(X,Y)+ L(Y,Z),则令L(X,Z)=L(X,Y)+L(Y,Z)，且记Z的上一跳为Y。转向(7)。  
 (12)使用溯源法，从每一个节点Y向X溯源，从而找到从X出发到Y的下一跳。具体做法是:根据Y的上一跳找到Z1，根据Z1的上一跳找到Z2....一直溯源找到X为止，如果Zn=X,则从X出发到Y的下一跳(为Zn-1)。  
 (13)调用界面展示函数展示路由表和拓扑图。  
  
 以上算法的活动图见图2.2。



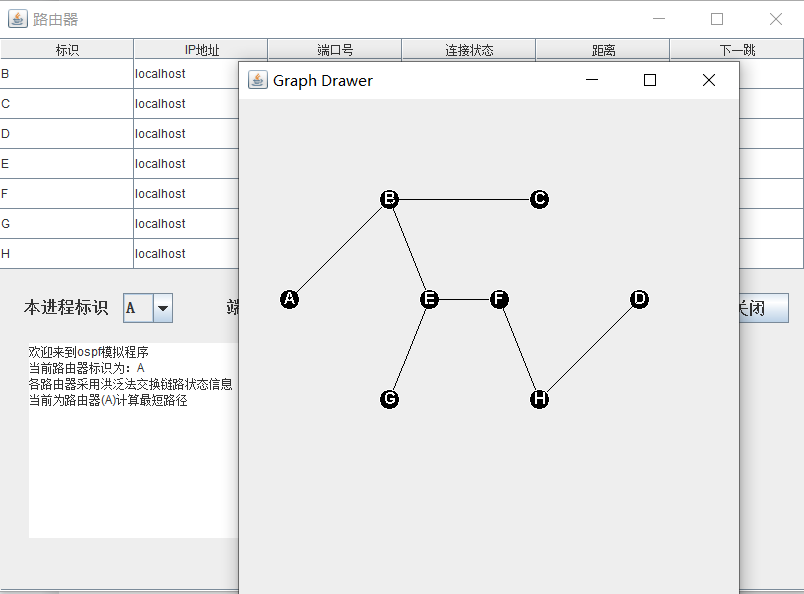
**图2.2 计算最短路径算法的活动图**

**3 界面样例**

针对本实验，路由器进程界面样例如图3.1 -- 3.3所示。



**图3.1 进程主界面样例**



**图3.2 最短路径生成树界面样例**



**图3.3 连接/断开路由连接界面样例**

界面中的表格主要用于显示各个节点的计算过程以及最后的计算结果，文本框主要用于显示各操作的功能。本实验中连接/断开路由连接实现了与用户的交互式指令，还给出了网络的拓扑图以及最后的最短路径生成树的图形展示，可以改善用户的使用体验。