# 项目设计文档

## 1 网络拓扑结构

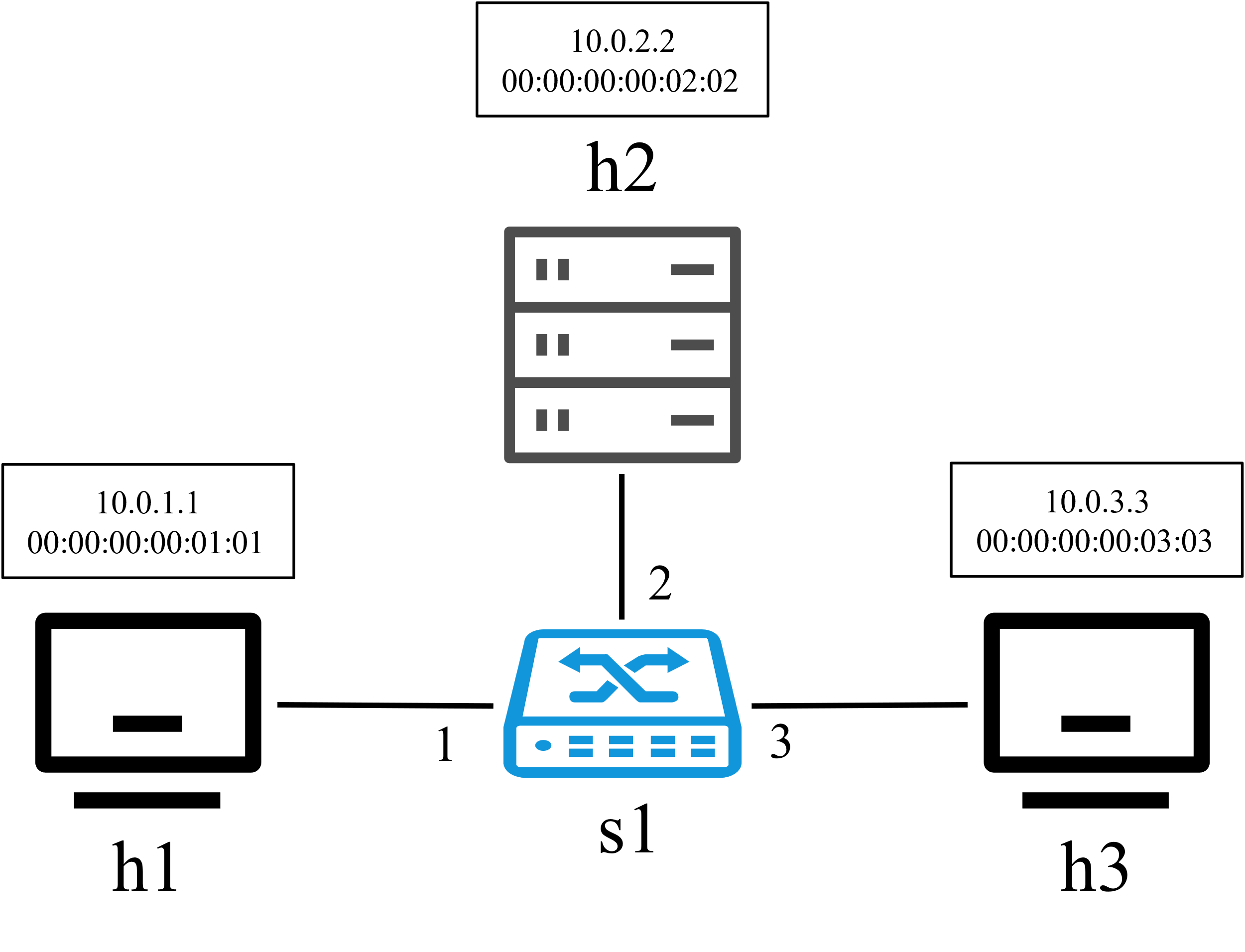


图1 网络拓扑结构图

如图1所示，这个拓扑中有1个switch，3个host，其中h2作为存储端，h1和h3分别作为发送端和接收端。

## 2 功能需求

用户通过使用编程接口来编写网络算法并运行在编程环境之中，然后加载运行时系统程序，来实现所需功能。该系统能运行实现以下三个功能：

转发功能：运行时系统程序根据用户编程的指定转发规则，自动下发流表项，完成自动化配置，例如：h1能将数据包能正确转发至h3。

存储功能：运行时系统程序根据用户编程的算法，选择是否打开存储功能，例如：若打开存储功能，运行时系统程序将会对转发的数据包另外存储一份在h2中。

计算/安全功能：交换机内部通过实现加解密功能来实现计算/安全功能，用户通过编程可以选择对交换机的某些ingress端口进行加密或解密，以及某些egress端口进行加密或解密，从而实现对流经交换机的数据包进行加密或解密功能。

## 3 功能相关表

一共包含三种功能：

1. 转发功能

* 该功能交换机内部存在一张表：table\_forward。
* 所需参数：[src\_ip, dst\_ip, action\_name, port]

运行时系统程序载入用户编程后的算法，将转发功能翻译转化为一个操作：srcIP，dstIP->修改转发port为3，对表table\_forward下发流表项：

|  |
| --- |
| "table\_entries": [  {  "table": "MyIngress.table\_forward",  "match": {  "hdr.ipv4.srcAddr": "10.0.1.1",  "hdr.ipv4.dstAddr": "10.0.3.3"  },  "action\_name": "MyIngress.set\_port",  "action\_params": {  "port": 3  }  }] |

2. 存储功能

* 该功能交换机内部存在两张表：table\_save、table\_redirect。
* 所需参数：[src\_ip, dst\_ip, ingress\_port, action\_name, port]

运行时系统程序载入用户编程后的算法，将存储功能翻译转化为两步操作，第一步是：srcIP，dstIP->修改转发port为2，对表table\_save下发流表项：

|  |
| --- |
| "table\_entries": [  {  "table": "MyIngress.table\_save",  "match": {  "hdr.ipv4.srcAddr": "10.0.1.1",  "hdr.ipv4.dstAddr": "10.0.3.3"  },  "action\_name": "MyIngress.set\_port",  "action\_params": {  "port": 2  }  }] |

第二步操作是：ingress\_port,destIP->修改转发port为3，对表table\_redirect下发流表项：

|  |
| --- |
| "table\_entries": [  {  "table": " MyIngress.table\_redirect",  "match": {  "standard\_metadata.ingress\_port": 2,  "hdr.ipv4.dstAddr": "10.0.3.3"  },  "action\_name": " save\_forward ",  "action\_params": {  "port": 3  }  }] |

3. 加密解密功能

* 该功能交换机内部存在四张表：table\_encr\_ingress、table\_encr\_egress、table\_decr\_ingress、table\_decr\_egress。
* 所需参数：[ingress\_port, action\_name]

运行时系统程序载入用户编程后的算法，根据是encr还是decr，以及是ingress\_port还是egress\_port，将加密功能翻译转化为一个操作，例如encr ingress\_port=1，翻译为ingress\_port ->指定端口加密，对相应表下发流表项：

|  |
| --- |
| "table\_entries": [  {  "table": " table\_encr\_ingress",  "match": {  "standard\_metadata.ingress\_port": 1  },  "action\_name": "encr\_ packet",  "action\_params": { }  }] |

## 4 运行时系统

运行时系统包括两部分：编程环境和控制器。结构如图2所示。

* 编程环境: 编程环境提供了编程接口，用户可以使用这些编程接口来编写网络算法并运行在编程环境之中。通过 P4 控制器来向交换机下发命令，来开启相关功能。
* P4 控制器: 负责和底层 P4 交换机的通信和交互。

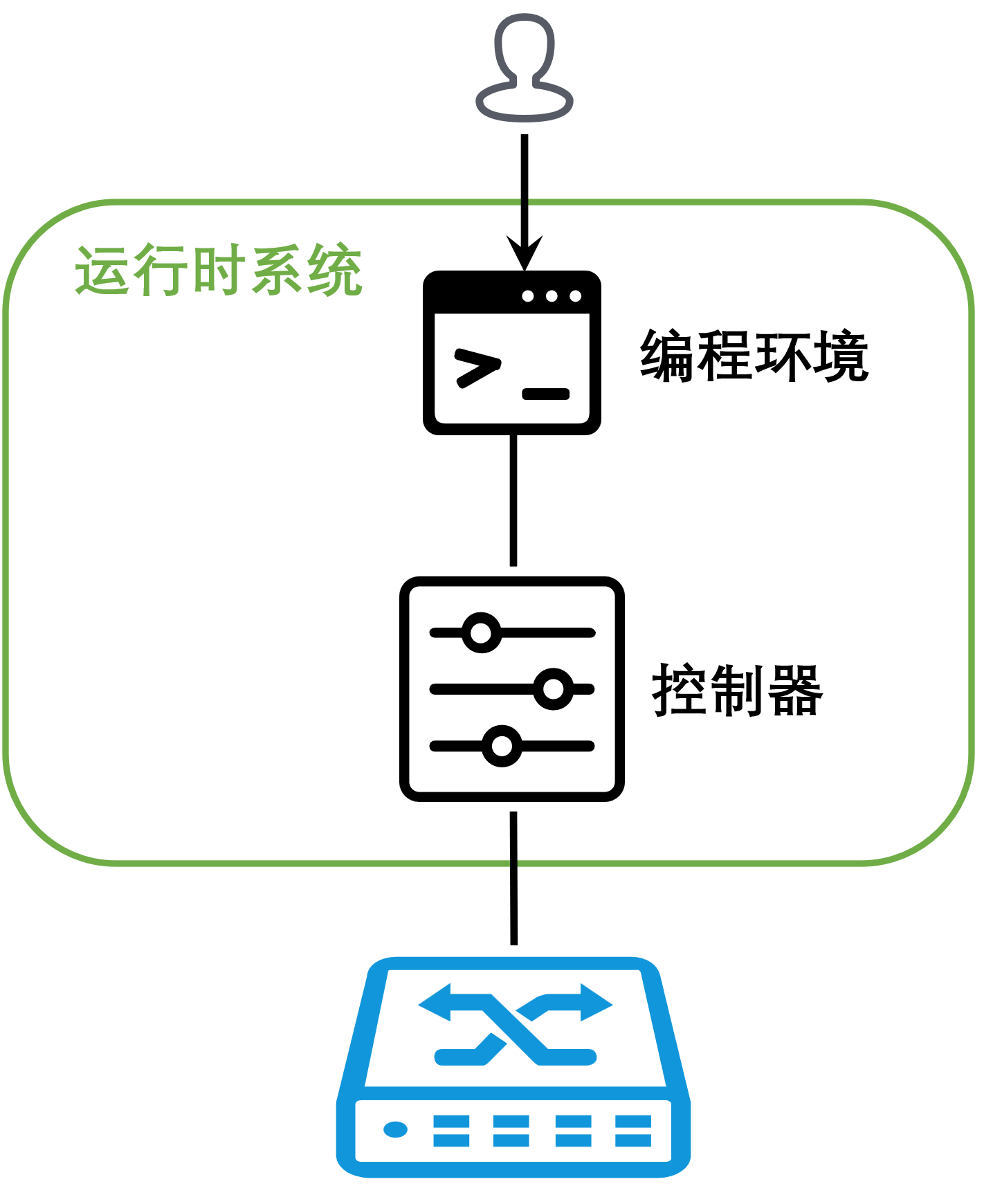


图2 运行时系统结构图

编程环境提供的具体编程接口以及接口接受的参数和返回值：

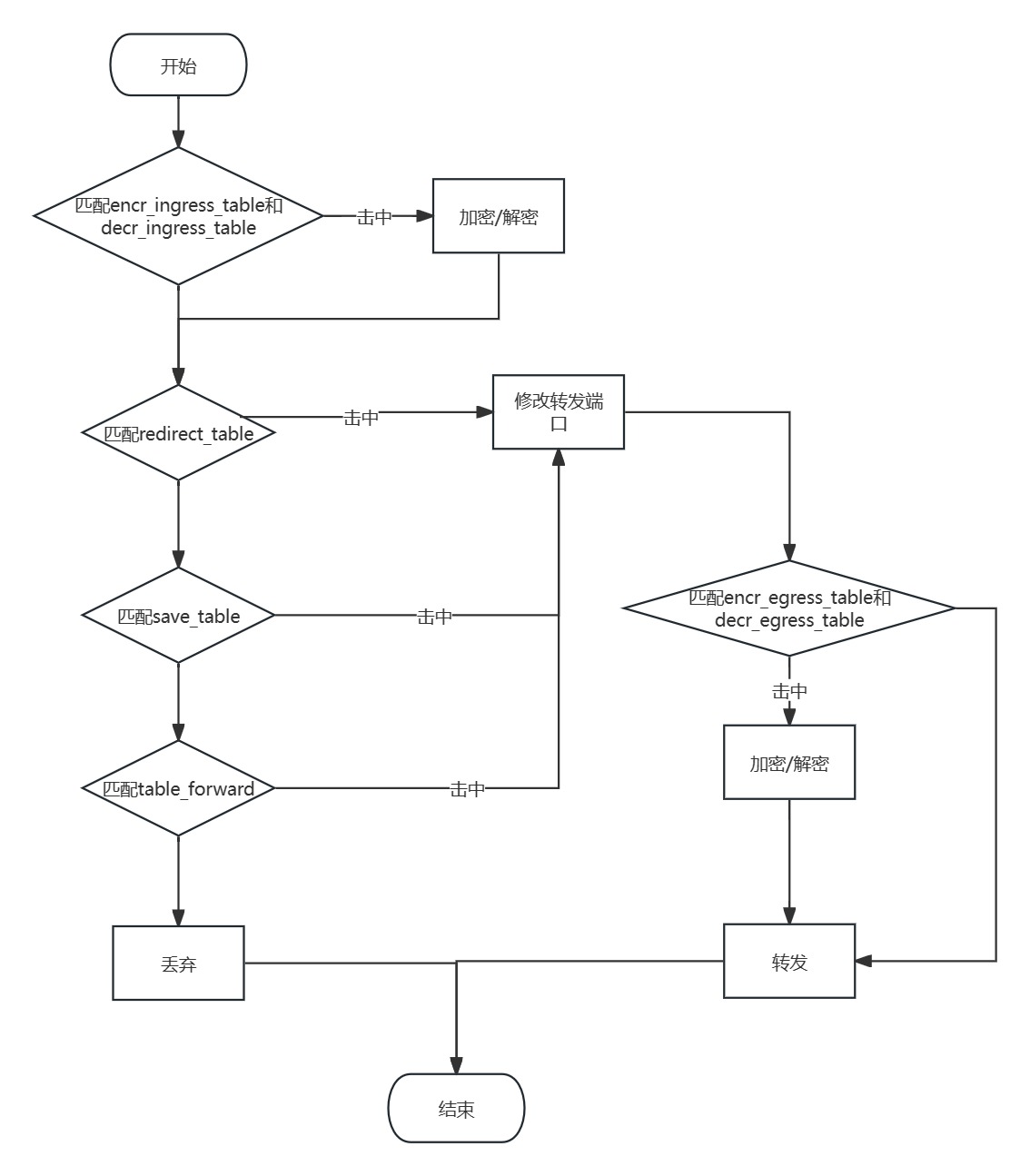
* bool comparer(string packetAdr, string matchAdr, int type)：根据type确定是源地址还是目的地址，比较传入数据包对应地址和用户想匹配的相应地址，如匹配返回true，否则返回false。
* void forward(int port)：根据用户编程流水线的源地址和目的地址，实现对参数port的转发功能。参数port指明转发输出端口
* void save(int savePort, int redirectPort)：根据用户编程流水线的源地址和目的地址，实现对参数savePort和参数redirectPort的存储功能。参数savePort指明数据包需转发存储的输出端口，参数redirectPort指明存储器重定向到目的地的转发输出端口。
* void encr(int type, int port)：根据用户编程流水线的源地址和目的地址，根据参数type实现对参数port加密功能。参数type指明ingress或egress。
* void decr(int type, int port)：根据用户编程流水线的源地址和目的地址，根据参数type实现对参数port解密功能。参数type指明ingress或egress。
* void endif()：结束一个if分支语句所必须调用的结束API。

以图1所示网络拓扑结构实现转发、存储、加解密功能为例示范网络算法的编写：

|  |
| --- |
| from config import \*  def pipeline(packet, rt):  if rt.comparer(packet.src, "10.0.1.1", SRCTYPE):  if rt.comparer(packet.dst, "10.0.3.3", DSTTYPE):  rt.forward(3)  rt.endif()  if rt.comparer(packet.dst, "10.0.3.3", DSTTYPE):  rt.save(2, 3)  rt.endif()  if rt.comparer(packet.dst, "10.0.3.3", DSTTYPE):  rt.encr(INGRESS, 1)  rt.decr(EGRESS, 3)  rt.endif()  rt.endif() |

## 5 工作流程

交换机处理数据包的具体工作流程如下：



## 6 项目目前进度

（1）实现转发功能API和存储功能API；

（2）实现编译用户编程代码，并产生对应流表项功能；

（3）完成mininet上环境搭建；

下一步计划完成转发和存储功能的p4软件代码和测试。