实验四　网络层协议实验(简化版)

上次编辑时间	@2025年5月30日 08:23
创建人	陈 陈洑
创建日期	@2025年5月25日

实验目的：

1.理解网络层的作用和工作机制。

2.掌握路由转发的作用、原理和算法。

3.理解路由选择的优先原则，能够编程设计实现路由选择匹配算法。

4.理解数据转发的过程，能编程实现在给定的模拟网络中检索到达目标网络的传输路径。

5.思考如何优化路由

实验原理

一、网络层的作用

网络层是 OSI 参考模型的第三层，主要解决**跨网络通信**的问题，实现数据在不同网络之间的传输。其核心作用包括：

1. **逻辑编址与寻址**
   - 为设备分配唯一的**逻辑地址**（如 IPv4/IPv6 地址），用于标识网络中的节点，解决 “如何找到目标设备” 的问题。
   - 例如：IP 地址 “192.168.1.100” 由网络部分和主机部分组成，网络层通过解析地址确定数据转发方向。

1. **路由选择与路径规划**

- 当数据需要跨多个网络传输时，网络层通过**路由协议**（如 RIP、OSPF、BGP）计算最佳路径，避免环路并优化传输效率。
- 例如：从北京到上海的数据包可能经过多个路由器，网络层负责选择最优路径。

2. **分组转发与分段**
   - 将上层（传输层）的报文分割为适合传输的**数据包**（Packet），并在必要时重组（如 MTU 不匹配时）。
   - 例如：传输层的 TCP 报文长度为 15000 字节，而链路层 MTU 为 1500 字节，网络层会将其拆分为 10 个数据包传输。
3. **流量控制与拥塞管理**
   - 通过丢弃冗余数据包、调整传输速率等方式避免网络拥塞，确保数据有序传输。

二、网络层工作原理

网络层的核心是基于**分组交换**和**路由转发**，其工作流程如下：

1. 数据封装与解封装

wireshark

- **发送端**：
  - 传输层报文（如 TCP/UDP 段）进入网络层，添加**IP 头部**（包含源 IP、目标 IP、协议号等），封装为 IP 数据包。
  - IP 头部示例：

版本	首部长度	服务类型	总长度	标识	标志	片偏移
4	20	0	500	1234	0	0

- **接收端**：
  - 剥离 IP 头部，将数据传递给传输层（根据协议号识别，如 TCP=6，UDP=17）。

2. 路由转发逻辑

- 路由器通过**路由表**决定数据包的转发路径。路由表包含以下关键信息：
  - **目标网络地址**：如 192.168.2.0/24（表示一个子网）。
  - **下一跳地址**：数据包应转发到的相邻路由器接口 IP。
  - **出接口**：数据包从当前路由器的哪个物理接口发出。
  - **度量值**：路径的 “代价”（如跳数、带宽、延迟等，用于路由协议计算最优路径）。
- **最长匹配原则**：当路由表中有多个匹配项时，选择**子网掩码最长**（即网络前缀最长）的条目。
  - 例如：路由表中有 192.168.1.0/24 和 192.168.1.128/25，目标 IP 为 192.168.1.150 时，匹配后者（/25 更精确）。

3. 跨网络传输的关键步骤

1. **主机发送数据**：主机 A（IP: 192.168.1.100）向主机 B（IP: 10.0.0.50）发送数据，先检查目标 IP 是否在同一子网。
2. **判断是否需要路由**：若目标 IP 不在同一子网，主机 A 将数据包发送给默认网关（路由器 R1 的接口 192.168.1.1）。
3. **路由器转发**：
   - R1 接收数据包，查看目标 IP（10.0.0.50），查询路由表，发现通过接口 GigabitEthernet 0/1 转发，下一跳为 172.16.0.2（路由器 R2）。
   - R1 剥离原数据链路层头部（如以太网帧头），重新封装为适合下一段链路的帧格式（如 PPP 或新以太网帧），并更新 TTL（生存时间，每经过一个路由器减 1，防止环路）。
4. **重复路由过程**：数据包经多个路由器转发，最终到达目标网络的路由器 R3，由 R3 交付给主机 B。

三、路由转发的工作原理（详细流程）

1. 路由表的构建方式

- **静态路由**：手动配置路由条目，适用于小型网络。

```
ip route 10.0.0.0 255.255.255.0 172.16.0.2  # 目标网络、掩码、下一跳
```

- **动态路由**：通过路由协议自动学习路由条目，适用于大型网络。
  - **内部网关协议（IGP）**：用于同一自治系统（AS）内，如 OSPF、EIGRP。
  - **外部网关协议（EGP）**：用于不同 AS 间，如 BGP。

2. 转发决策的核心逻辑

1. **查询路由表**：路由器收到数据包后，提取目标 IP 地址，在路由表中查找匹配的路由条目。
2. **最长匹配优先**：选择子网掩码最长的条目（更精确的路由）。
3. **递归查找下一跳**：若路由条目的下一跳是另一个路由器的 IP，需继续查询该 IP 对应的出接口（如通过 ARP 解析 MAC 地址）。
4. **数据链路层封装**：根据出接口的类型（如以太网、PPP），封装新的数据链路层头部（包含目标 MAC 地址）。
5. **转发数据包**：将封装后的数据包从指定接口发出。

3. 关键机制

- **生存时间（TTL）**：防止数据包在网络中无限循环。每经过一个路由器，TTL 减 1，当 TTL=0 时丢弃数据包并返回 ICMP 超时消息。
- **分片与重组**：若数据包大小超过链路 MTU，路由器将其分片（添加分片偏移字段），目标主机负责重组。
- **ICMP 协议**：用于网络层的错误报告和控制（如 ping 使用 ICMP Echo Request/Reply）。

实验内容

铺垫概念：

（一） 路由表的结构

一、路由表的核心字段

字段	说明
目标网络地址	数据包的目标网络或主机地址（如 `192.168.1.0/24`），表示路由匹配的范围。
子网掩码	与目标网络地址配合，标识网络前缀长度（如 `/24` 表示前 24 位为网络号）。
下一跳地址	数据包转发的下一个路由器接口的 IP 地址（若为直连网络，下一跳可为 `0.0.0.0`）。
出接口	数据包从当前路由器发出的物理接口或逻辑接口（如 `GigabitEthernet0/1`）。
管理距离（AD）	表示路由条目的可信度（数值越小越优先），用于不同路由协议间的优先级比较。
度量值（Metric）	路由协议计算的路径 “代价”（如跳数、带宽、延迟等），用于同协议内的路径优选。
路由类型	标识路由的来源（如直连、静态、动态协议）。
更新时间	路由条目最后一次更新的时间（动态路由协议使用，用于老化机制）。
路由标记	可选字段，用于路由策略或 BGP 等协议的路径标记（如社区属性）。

二、字段详解

1. 目标网络地址与子网掩码

- **作用**：确定路由匹配的数据包范围。例如：
  - `10.0.0.0/8` 表示匹配前 8 位为 `10` 的所有 IP 地址（覆盖 `10.0.0.0 ~ 10.255.255.255`）。
  - `192.168.1.100/32` 表示精确匹配单个主机 IP（`/32` 子网掩码为 `255.255.255.255`）。
- **最长匹配原则**：当多个路由条目匹配同一目标 IP 时，选择**子网掩码最长**（即网络前缀最长）的条目。

2. 下一跳地址（Next Hop）

- **非直连网络**：数据包需转发到相邻路由器的接口 IP。例如：

> 目标网络：172.16.0.0/16，下一跳：192.168.1.2（相邻路由器 R2 的接口 IP）

- **直连网络**：当目标网络与路由器接口直接相连时，下一跳为 `0.0.0.0`，数据包直接通过出接口发送（如同一子网内的主机）。

3. 出接口（Out Interface）

- 指明数据包从哪个接口发出，可能是物理接口（如 Ethernet0/0）或逻辑接口（如隧道接口、Loopback 接口）。
- **示例**：路由器通过接口 `GigabitEthernet0/1` 连接到子网 `192.168.2.0/24`，则该子网的路由条目出接口为 `GigabitEthernet0/1`。

4. 管理距离（Administrative Distance, AD）

- **作用**：衡量路由来源的可信度，用于比较不同路由协议的优先级。数值越小，优先级越高。
- **常见协议的 AD 值**：

路由类型	管理距离（AD）
直连路由	0
静态路由	1
EIGRP	90
OSPF	110
RIP	120
BGP	20（外部）/200（内部）
未知 / 不可达	255

- **示例**：若路由器同时通过静态路由和 RIP 学习到同一目标网络，由于静态路由 AD=1 小于 RIP AD=120，优先使用静态路由。

（二） 标准路由协议的编程架构

1. 核心组件

```
┌─────────────────────────────────────────────────────┐
│              路由协议守护进程                        │
├─────────────────────────────────────────────────────┤
│ 协议处理模块 │ 路由计算模块 │ 路由表管理模块 │
├─────────────────────────────────────────────────────┤
│ 网络接口监听 │ 定时器管理   │ 配置管理       │
└─────────────────────────────────────────────────────┘
```

- **协议处理**：解析和生成路由协议报文（如 OSPF 的 LSU、RIP 的 Response）。
- **路由计算**：实现 Dijkstra 或 Bellman-Ford 算法，生成最优路径。
- **路由表管理**：维护和更新路由表，处理路由优先级和冲突。

2. 关键数据结构

```
// 标准路由表项结构示例
typedef struct {
    uint32_t destination;    // 目标网络地址
    uint32_t mask;           // 子网掩码
    uint32_t next_hop;       // 下一跳地址
    int interface;           // 出接口索引
    int metric;              // 度量值
    int protocol;            // 路由协议类型 (RIP=1, OSPF=2)
    int admin_distance;      // 管理距离
    time_t last_update;      // 最后更新时间
} RouteEntry;

// 路由表结构（哈希表或树实现）
```

```
typedef struct {
    RouteEntry *entries;     // 路由条目数组
    int size;               // 当前大小
    int capacity;           // 最大容量
} RoutingTable;
```

实验要求

（一）根据路由表搜索到达目标地址的最佳路由

根据路由表搜索到达目标地址的最佳路由，使用掩码长度优先和度量值优先的策略搜索路由项。首先是掩码长度优先原则，掩码长度相同的度量值优先。

掩码长度策略：掩码越长网络规模越小，意味着定位越准确，优先级越高。

度量值策略：度量值越小，意味着传输代价越小，优先级越高。

- 给定路由器配置数据和一个目标ip地址，编写函数查找下一跳IP地址，函数定义如下：

  uint32_t find_best_route(RouterInfo *router, const uint32_t dest_ip)
  返回值为下一跳IP

实验要求为参考给出的范例代码，编写主函数，初始化运行环境，从文件中读入一个路由器配置，调用该函数得到下一跳地址，并以点分式字符串方式输出。

（二）传输路径搜索
在网络中搜索到达目标的传输路径

1、在给出的网络中依据给定路由算法，搜索到达目标的传输路径。
2、程序中用路由器数组模拟一个网络，指定从其中的一个路由器（通过IP指定）发起传输。
3、如果已到达目标网络，输出路由器的接口IP。
4、如果未到达目标网络，检索路由表找到下一跳IP并输出。
找不到路由项时，终止路由过程。
找到路由项时重复之上过程，直到到达目标网络。
5、为防止路由环路造成死循环，设置最大跳数，到达最大跳数结束。

- 给定一组路由器配置列表和一个IP地址，编写函数找出传输路径，输出节点途经的节点IP保存在整型数组中，变量及函数定义如下：

```
define MAX_HOP_NUM 100                  //最大节点跳数
uint32_t routepath[MAX_HOP_NUM]         //途径节点IP数组
Int find_route_path(RouterInfo *routerlist, const uint32_t dest_ip, uint32_t route_path[] )
```

途经的下一跳IP保存在route_path[]数组中，返回途经路由跳数。

实验要求为参考给出的范例代码，编写主函数初始化运行环境，从配置文件中读入多个路由器配置，生成routerlist[]，调用该函数查找路由转发的路径，在主程序中以点分式字符串方式输出途经的IP地址，一行一个IP。

（三）动态路由与路径优化（挑战）

静态路由中，每个路由器都只根据自身路由表检索最优路由项，但该路由项不一定是全局的最优路径。
路由器根据互相交换的邻接网络信息产生并优化路由表。
路由器根据链路状态、负载情况调整路由表。
路由器根据优先级策略满足特定传输等

实验指导

一、开发环境

windows平台下dev c++或其他类C开发工具

头文件：

```
#include <stdio.h>     // 标准输入输入
#include <stdint.h>    // uint8_t,uint32_t数据类型声明
#include <string.h>    // 字符串函数
```

二、编程实验数据结构

```
// 定义用于存储单个IP地址和掩码的结构体
typedef struct {
      uint32_t ip_address;         //IP地址
      uint32_t subnet_mask;        //网络掩码
           } IPAndMask;
// 定义用于存储单个路由条目的结构体
typedef struct {
      uint32_t destination_network; //目标网络
      uint32_t destination_mask;    //目标网络掩码
      uint32_t next_hop_ip;         //下一跳地址
      int metric;                   //度量值
           } RouteEntry;
// 定义用于存储路由器信息的结构体
typedef struct {
      IPAndMask ip_and_masks[MAX_IPS];   //IP地址列表
      int ip_and_masks_count;            //IP地址数量
      RouteEntry route_table[MAX_ROUTES];//路由表
      int route_table_count;             //路由项数量
      char router_name[32];              //路由器名称
          } RouterInfo;
```

三、输入输出格式定义

Router_Config.TXT 文件中记录了一个路由器的配置信息，包括拥有的IP地址和掩码和路由表，具体格式为:
以
ROUTER字符串开头的行表示路由器描述开始，ROUTER字符串后是路由器名称。接下来的每一行都是该路由器的配置信息。
以
ADDRESS开头的行描述该路由器拥有的IP地址，后面依次是接口编号、IP地址和掩码，用空格分隔。
以
ROUTE开头的行描述路由表中的一个路由项，后面依次是 网络地址、掩码、下一跳IP地址、度量值，用空格分隔。

```
ROUTER A
ADDRESS 0 192.168.1.1 255.255.255.0
ADDRESS 1 192.168.2.2 255.255.255.0
ADDRESS 2 192.168.101.1 255.255.255.0
ROUTE 192.168.3.0 255.255.255.0 192.168.1.2 10
ROUTE 192.168.5.0 255.255.255.0 192.168.1.2 10
ROUTE 192.168.3.0 255.255.255.0 192.168.2.1 9
ROUTE 192.168.0.0 255.255.0.0 192.168.2.1 9
ROUTE 0.0.0.0 0.0.0.0 192.168.1.2

ROUTER B
ADDRESS 0 192.168.2.1 255.255.255.0
ADDRESS 1 192.168.3.2 255.255.255.0
ADDRESS 2 192.168.102.1 255.255.255.0
ROUTE 192.168.4.0 255.255.255.0 192.168.3.1 9
ROUTE 192.168.0.0 255.255.0.0 192.168.3.1 9

ROUTER C
ADDRESS 0 192.168.3.1 255.255.255.0
ADDRESS 1 192.168.4.2 255.255.255.0
```

```
ADDRESS 2 192.168.103.1 255.255.255.0
ROUTE 192.168.5.0 255.255.255.0 192.168.4.1 9
ROUTE 192.168.0.0 255.255.0.0 192.168.4.1 9

ROUTER D
ADDRESS 0 192.168.4.1 255.255.255.0
ADDRESS 1 192.168.5.2 255.255.255.0
ADDRESS 2 192.168.104.1 255.255.255.0
ROUTE 192.168.1.0 255.255.255.0 192.168.5.1 9
ROUTE 192.168.0.0 255.255.0.0 192.168.5.1 9

ROUTER E
ADDRESS 0 192.168.5.1 255.255.255.0
ADDRESS 1 192.168.1.2 255.255.255.0
ADDRESS 2 192.168.105.1 255.255.255.0
ROUTE 192.168.2.0 255.255.255.0 192.168.1.1 9
ROUTE 192.168.0.0 255.255.0.0 192.168.1.1 9
ROUTE 0.0.0.0 0.0.0.0 192.168.1.1
```

四、函数定义

要编写的相关的函数有：

1、判断两个IP是否在同一网络：

```
bool are_same_net(uint32_t ip1, uint32_t ip2, uint32 mask)
```

2、在路由表中查找最佳路由：

```
uint32_t find_best_route(RouterInfo *router, const uint32_t dest_ip )
```

3、找出最优传输路径，输出节点途经的节点IP保存在整型数组中：

```
Int find_route_path(RouterInfo *routerlist, const uint32_t dest_ip, uint32_t route_path[] )
```