附加题——树莓派路由实验

提纲

- 实验背景介绍
- 实验目的
- 实验方案

实验背景介绍

- 网络中的数据包是如何到达自己想要到达的目的地的？
  - 首先要知道目的地的地址，一个可以互相通信的系统首先要能够对通信实体进行唯一的标识，在网络中这个标识就是IP地址。(Internet Protocol Address)
  - 知道了目的地的地址之后，有2种情况，一种是源和目的地是直接相连的，一种的源和目的地不直接相连。
  - 直接相连的不必多说
  - 源和目的地不直接相连的话，需要中间好几跳才能从源抵达目的地

实验背景介绍

- 网络中的数据包是如何到达自己想要到达的目的地的？
  - 源和目的地中间相隔好几跳：路由（routing）--通过互联的设备将数据包从源地址传输到目的地址

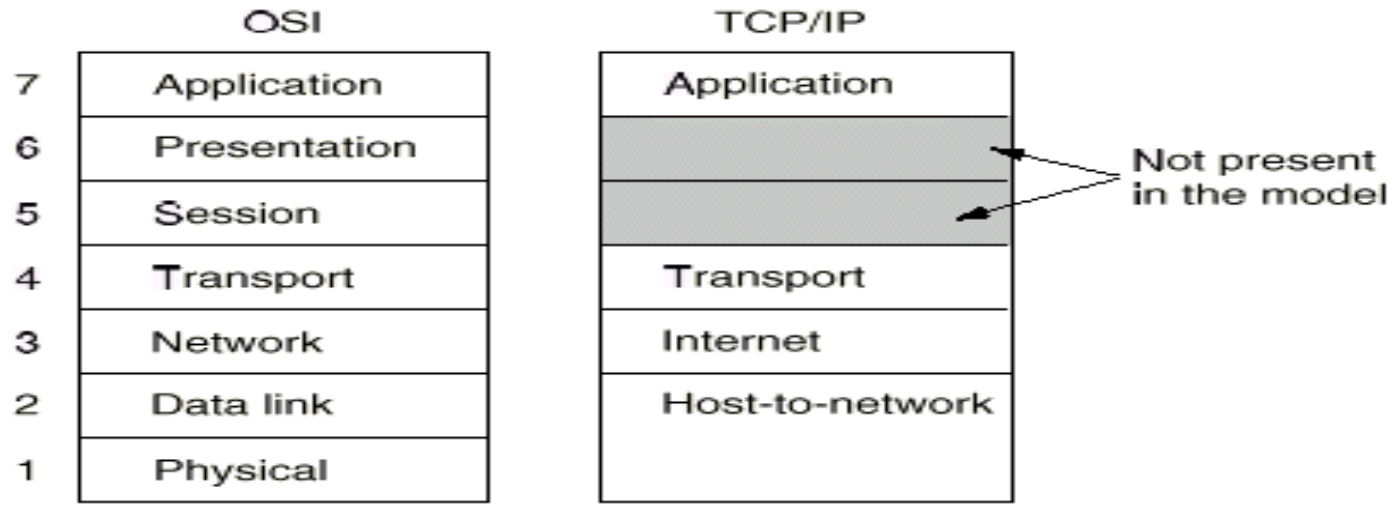
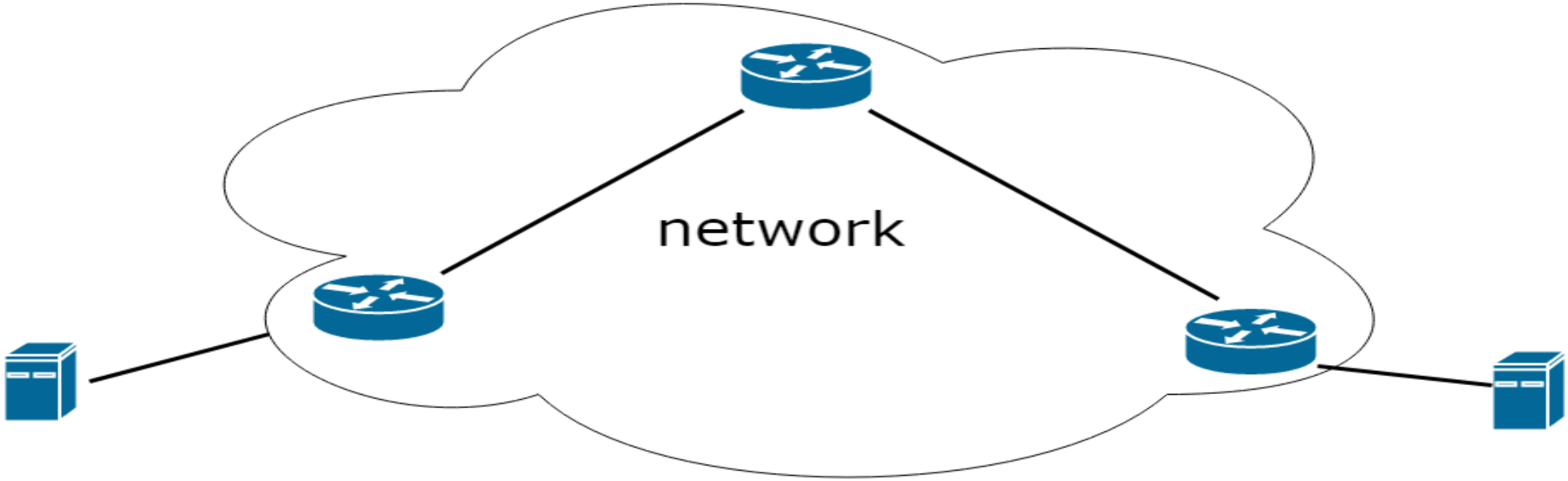


Fig. 1-18. The TCP/IP reference model.

实验背景介绍

路由的核心设备:路由器

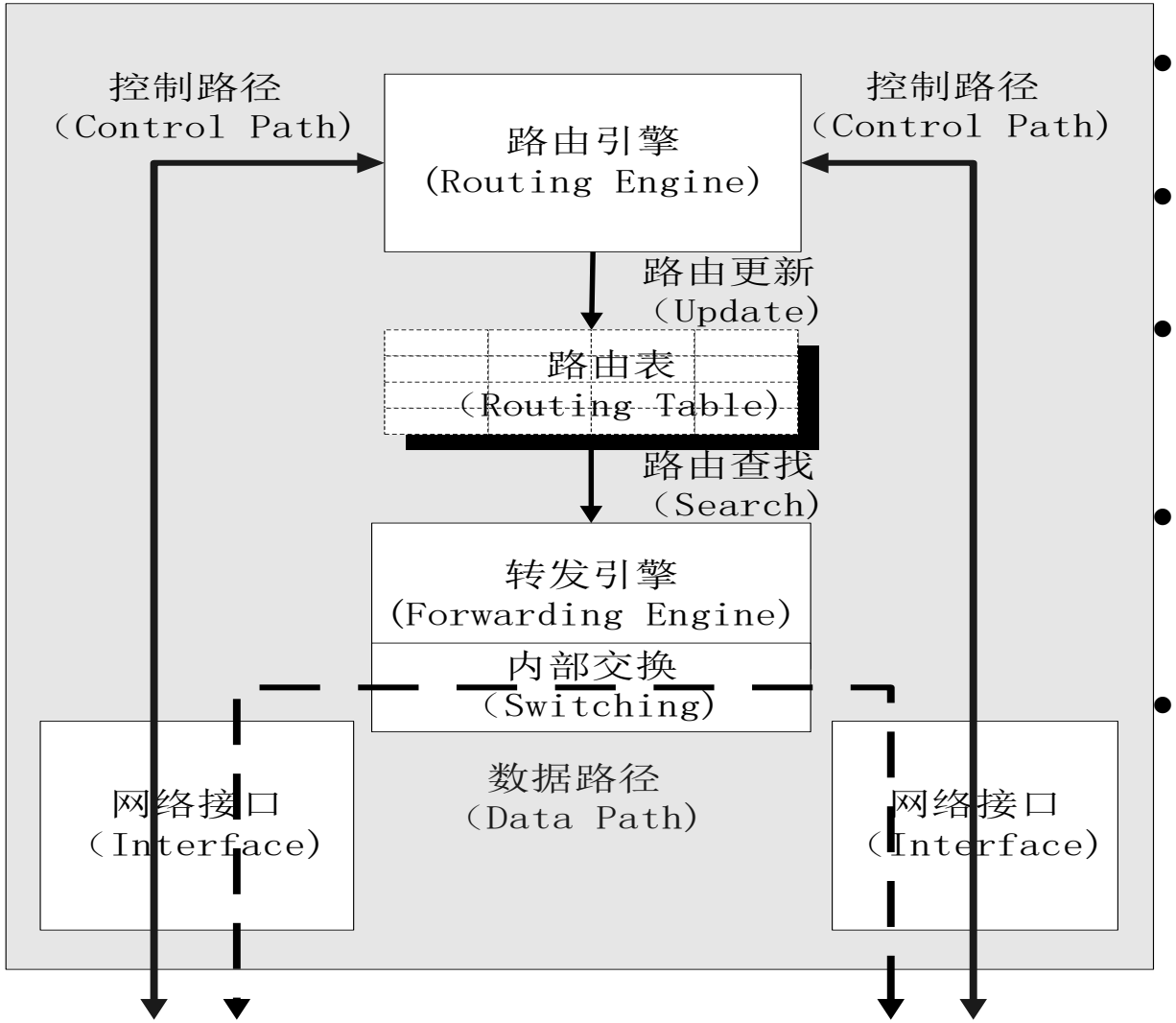
路由器的作用:寻路，给IP包寻找正确的通往目的地的路径



实验背景介绍

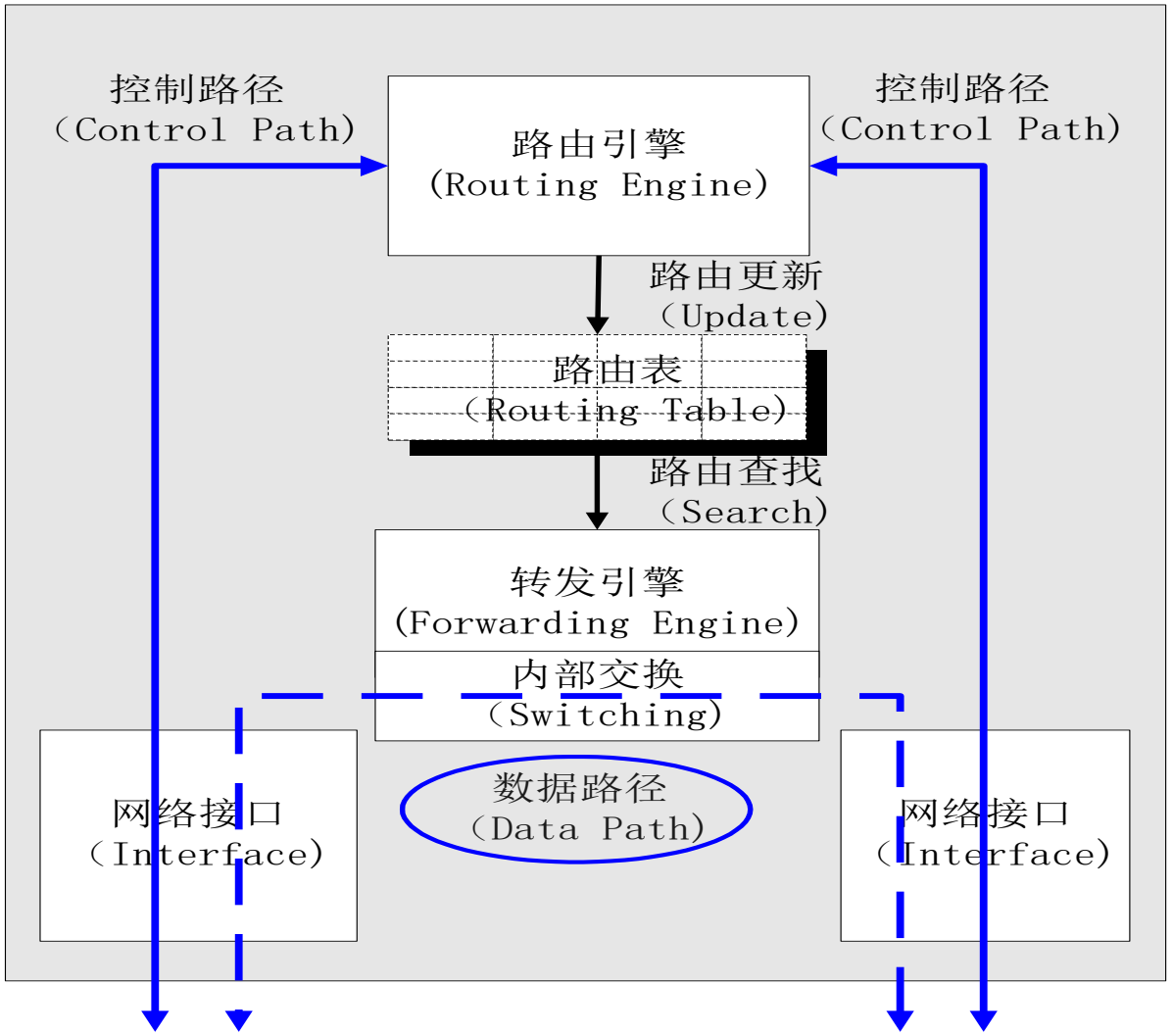
- 路由器中的2个重要功能：
  - 路由：做出路由决策，具体为处理路由协议消息，维护更新路由表
  - 转发：根据路由决策，将收到的特定数据包从相应的outgoing line发出去，具体为检查包头，实行查表，找到转发出口
- 路由的分类
  - 按照路由算法是否是适应性的。
    - 静态路由：提前对网络中的路由器做好配置，网络情况简单，拓扑相对静态
    - 动态路由：网络的路由器具备智能，能根据网络中发生的拓扑变化进行路由决策的改变
    - 动态路由根据使用的路由算法又可以分为距离向量路由协议和链路状态路由协议
  - 按照网络范围
    - 域内路由：在一个自治域（AS）内
    - 域间路由：自治域之间

实验背景介绍--路由器结构



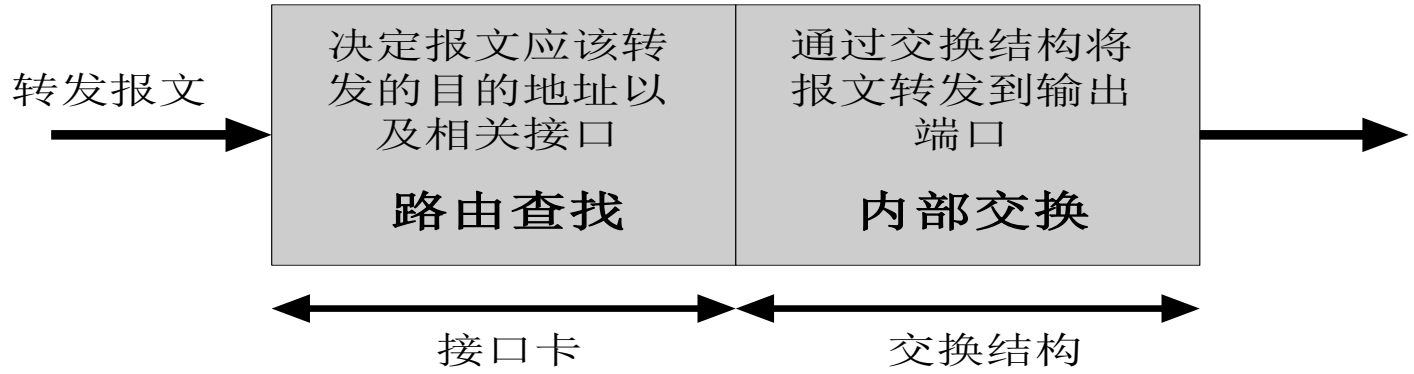
- 网络接口
  - 完成网络报文的接收和发送。
- 转发引擎
  - 负责决定报文的转发路径。
- 内部交换
  - 为多个网络接口以及路由引擎模块之间的报文数据传送提供高速的数据通路。
- 路由引擎
  - 由运行高层协议(特别是路由协议)的内部处理模块组成。
- 路由表
  - 路由表包含了能够完成网络报文正确转发的所有路由信息，它在整个路由器系统中起着承上启下的作用。

实验背景介绍—报文处理路径



- 路由器提供了两种不同的分组处理路径：
  - ❖数据路径：处理目的地址不是本路由器而需要转发的分组，因此数据路径是整个路由器的关键路径，它的实现好坏直接影响着路由器的整体性能。
  - ❖控制路径：处理目的地址是本路由器的高层协议分组，特别是各种路由协议分组。虽然控制路径不是路由器的关键路径，但是它负责完成路由信息的交互，从而保证了数据路径上的分组沿着最优的路径转发。

实验背景介绍—数据路径的工作流程

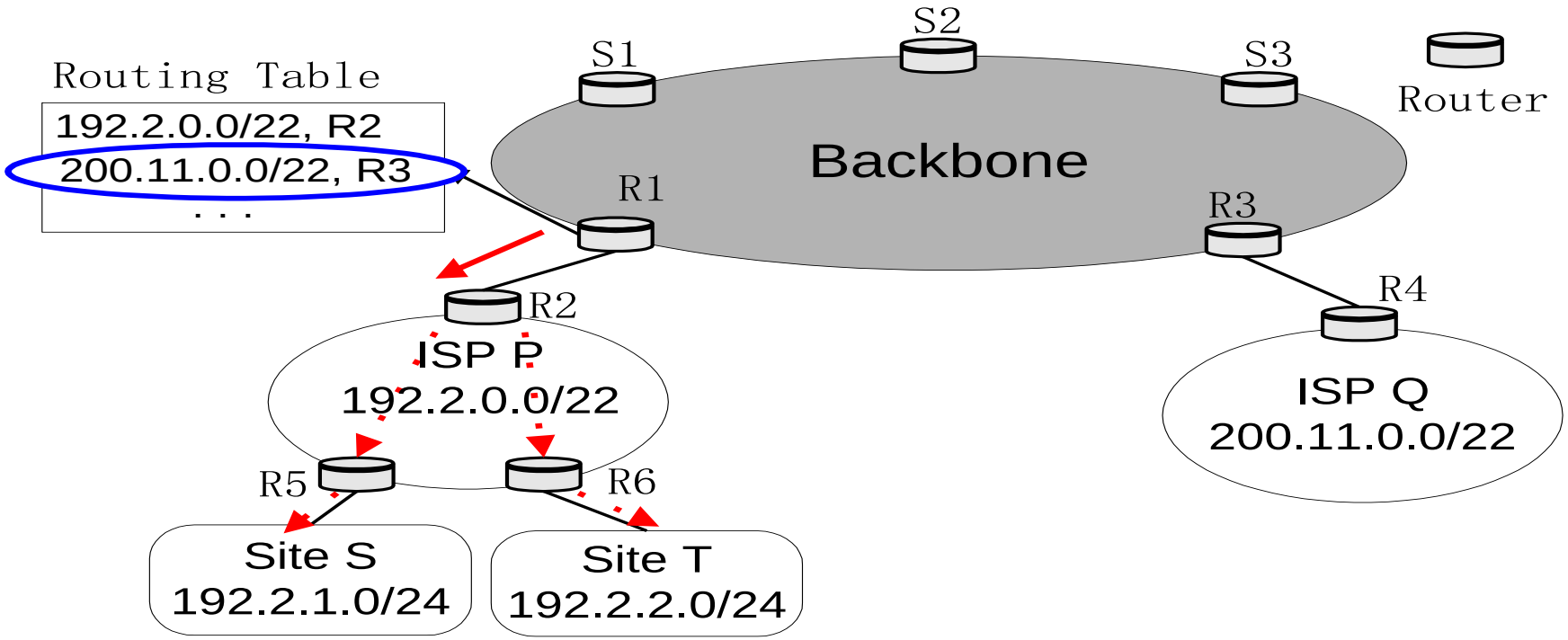


- RFC1812规定IP路由器必须完成两个基本功能：
  - 首先路由器必须能够对每个到达本路由器的报文做出正确的转发决策，决定报文向哪一个下一跳路由器转发。为了进行正确的转发决策，路由器需要在转发表中查找能够与转发报文目的地址最佳匹配的表项，这个查找过程被称为路由查找（Route Lookup）。
  - 其次路由器在得到了正确的转发决策之后必须能够将报文从输入接口向相应的输出接口传送，这个过程被称为内部交换过程（Switching）。
- 路由查找和内部交换是路由器设计的关键问题。

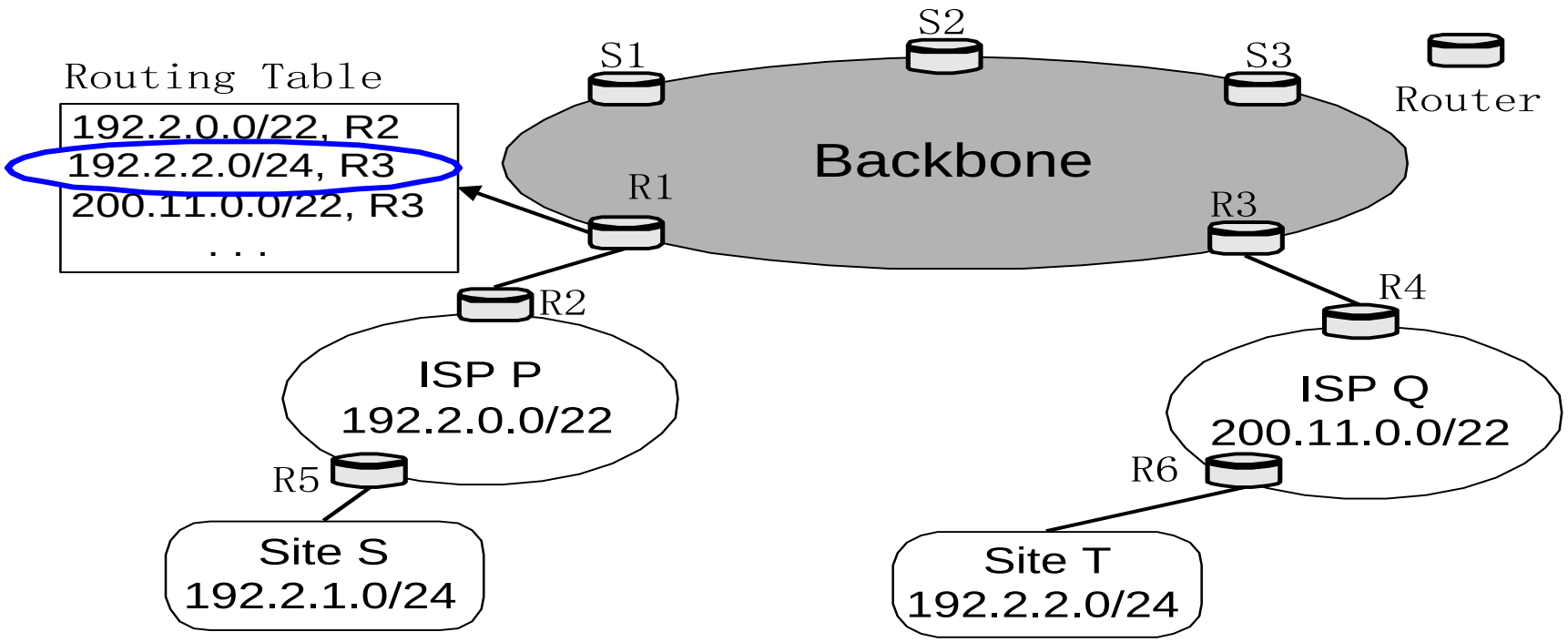
实验背景介绍--路由查找算法

- 传统的路由查找算法
  - 线性查找法
  - 缓存最近查找过的目的地址
  - 二分支trie树查找法
  - 路径压缩的二分支trie树查找法
- 近年提出的路由查找算法
  - 多分支trie树查找法
  - 地址前缀长度的二分查找法
  - 地址区间的二分查找法
  - TCAM硬件查找法

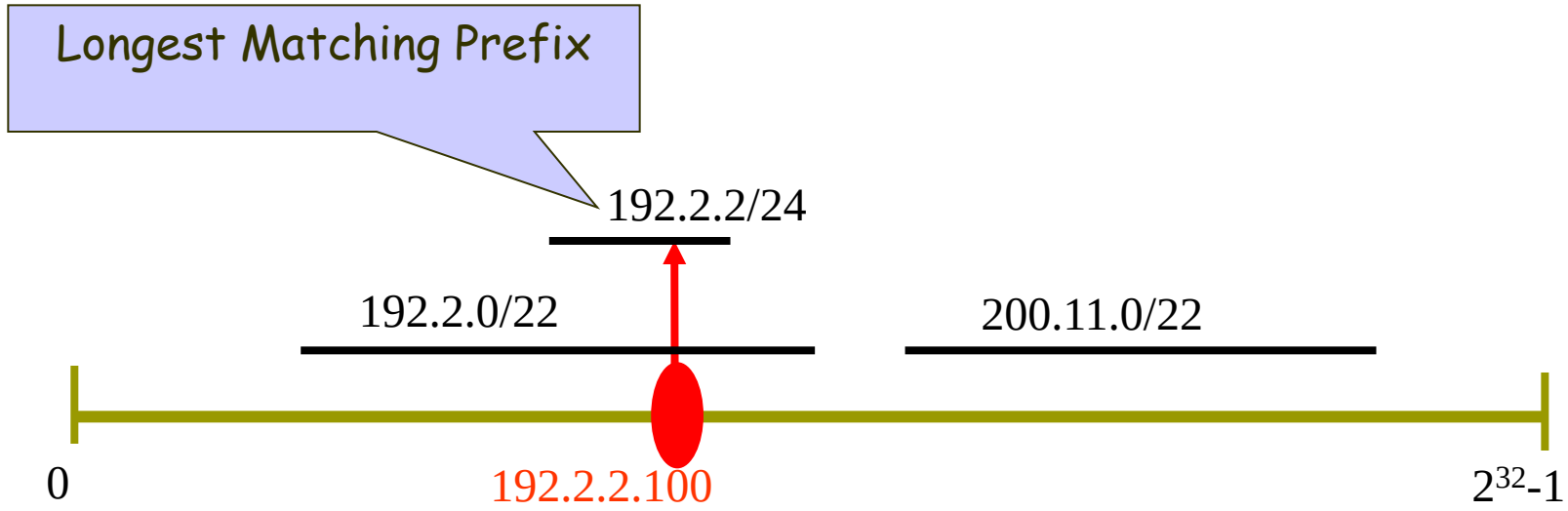
最长前缀匹配


Routing Table
192.2.0.0/22, R2
200.11.0.0/22, R3
. . .
S1
S2
S3
Router
Backbone
R1
R3
R2
R4
ISP P
192.2.0.0/22
ISP Q
200.11.0.0/22
R5
R6
Site S
192.2.1.0/24
Site T
192.2.2.0/24

最长前缀匹配(续)


Routing Table
192.2.0.0/22, R2
192.2.2.0/24, R3
200.11.0.0/22, R3
. . .
S1
S2
S3
Router
Backbone
R1
R3
R2
R4
ISP P
192.2.0.0/22
ISP Q
200.11.0.0/22
R5
R6
Site S
192.2.1.0/24
Site T
192.2.2.0/24

最长前缀匹配(续)



路由查找: 在所有前缀中寻找最长匹配

路由查找算法的分类

基于地址前缀值的路由查找算法

通过对整个地址前缀空间进行地址关键字穷举法来避免对地址前缀长度进行考虑。

线性查找法、地址区间的二分查找法、TCAM硬件查找法

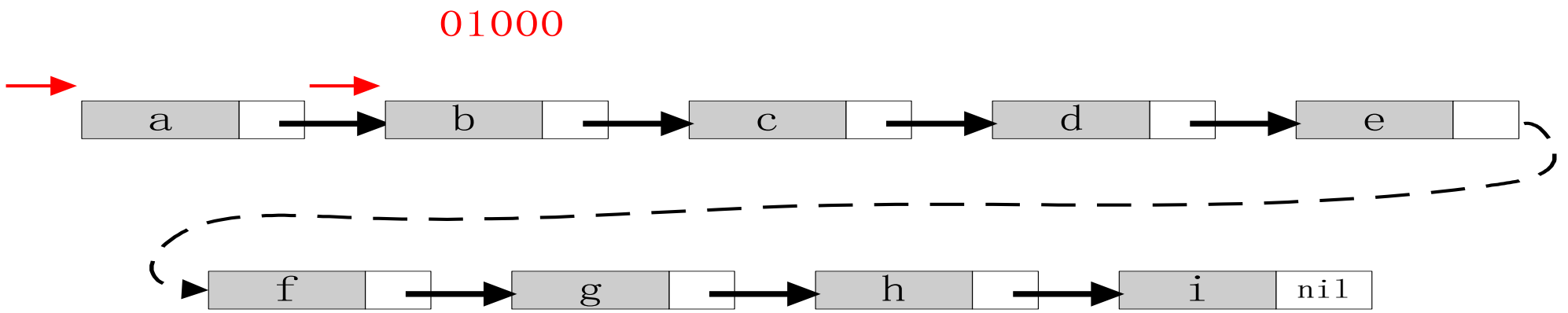
基于地址前缀长度的路由查找算法

从前缀长度的角度入手进行路由查找

trie树(包括二分支、多分支)、前缀长度空间的二分查找法

线性查找法

Prefixes
a 0*
b 01000*
c 011*
d 1*
e 100*
f 1100*
g 1101*
h 1110*
i 1111*



实现简单

查找效率低，查找过程的算法复杂度为O(N)

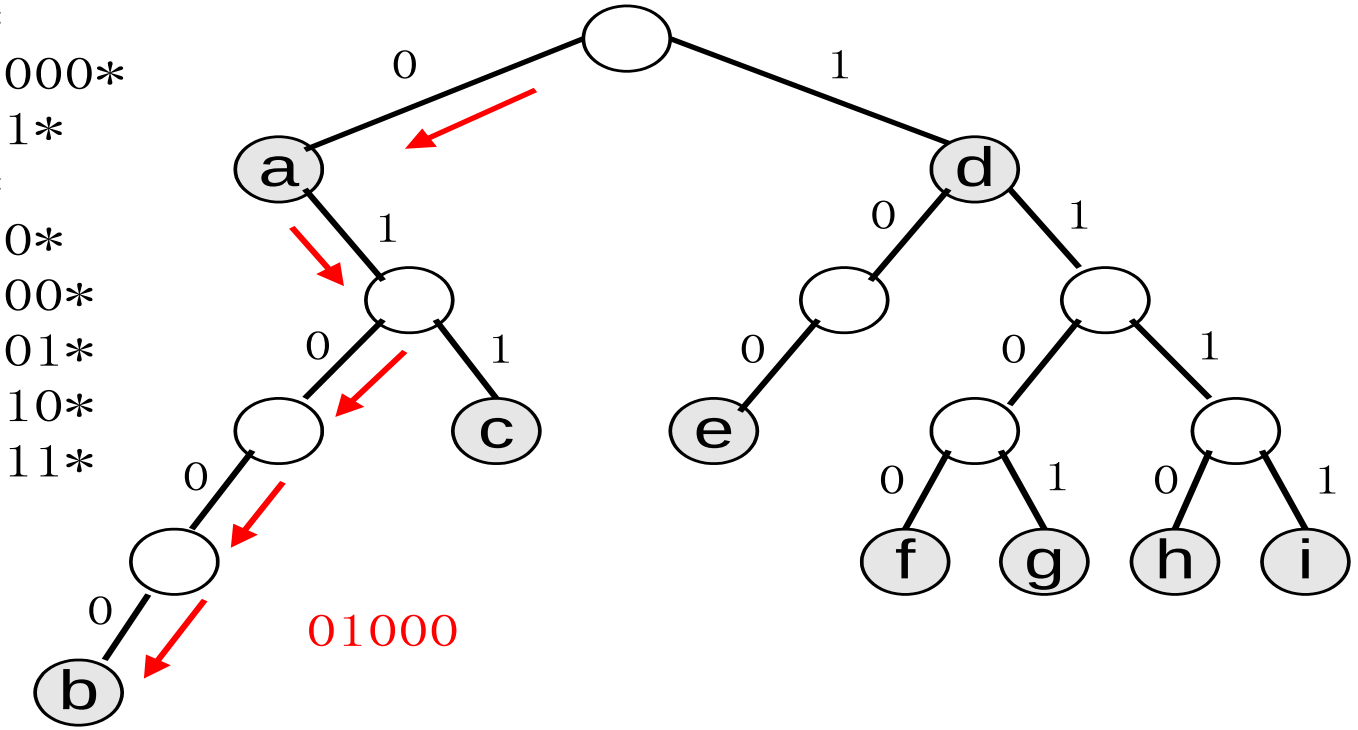
存储空间复杂度为O(N)，插入删除复杂度为O(1)

一种改进：将前缀长度大的路由表项放在链表的前列，平均查找性能会有改进，但是路由更新复杂度变为O(N)

只适用于路由表项比较少的早期低端路由器

二分支TRIE树查找法(A BINARY TRIE)

Prefixes
a 0*
b 01000*
c 011*
d 1*
e 100*
f 1100*
g 1101*
h 1110*
i 1111*



数字查找树的每个结点不是关键字，而是组成关键字的符号

实现简单

适用性好，可以用于任何长度关键字的查找

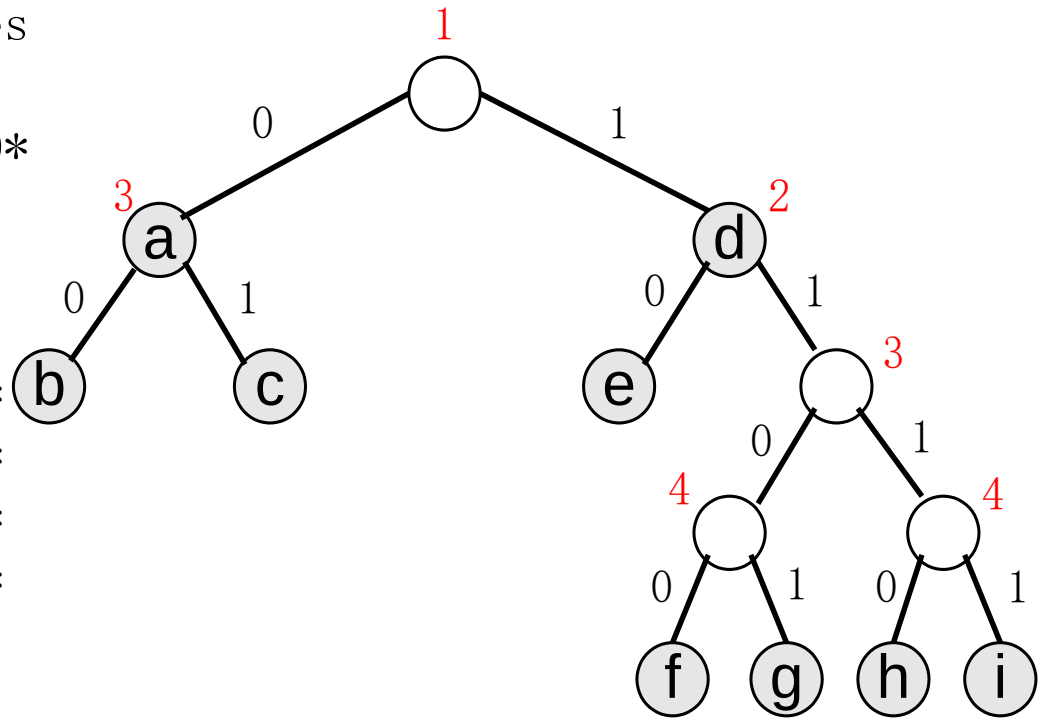
查找效率低，最差情况下**O(W)，W = logN**

存储效率较低

Trie树查找过程实际上就是在长度空间内的顺序查找操作

路径压缩trie树（Path-Compressed Trie）

Prefixes
a 0*
b 01000*
c 011*
d 1*
e 100*
f 1100*
g 1101*
h 1110*
i 1111*

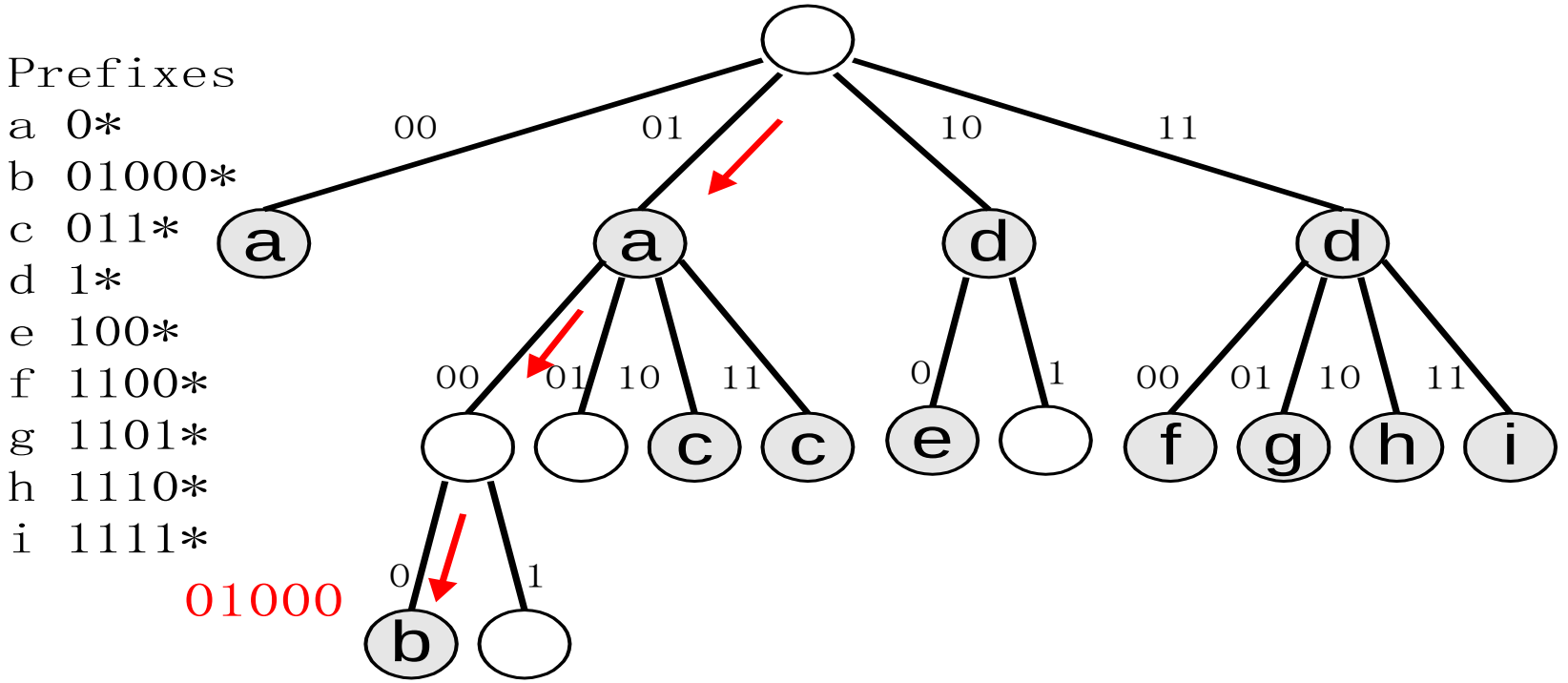


- Trie树中单分支结点的存在既增加了搜索的深度，又增加了存储空间
- 由于忽略了地址中某些位的匹配操作，结点处需要有一个变量指示下一个要检查的位
- 路径压缩trie树前缀结点保存的是地址前缀
- 当到达叶子结点或前缀匹配失败时，查找结束
- 最早在PATRICIA算法中提出，并在BSD UNIX中实现。

查找算法使用的辅助策略

- 前缀扩展(Prefix Expansion)
  - 将一条长度较短的前缀展开成多条长度较长的前缀集
  - 前缀扩展技术可以把包含各种前缀长度的前缀集转化为只包含较少前缀长度的前缀集
- 独立前缀转化(Disjoint Prefix Transformation)
  - 为了解决前缀地址空间的重叠和最长匹配问题，可以将地址前缀集转化为完全独立的前缀集
  - 根据独立前缀集构造的trie树中所有前缀结点都出现在叶子上，也称为叶子扩展(leaf pushing)技术
- 压缩技术(Compression Techniques)
- 优化技术(Optimization Techniques)
- 存储层次(Memory Hierarchy)

多分支trie树查找法



- 优点
  - 提高了查找效率，复杂度为O(W/k)，其中k为trie树步宽
- 缺点
  - 存储空间的需求大，利用率不高
  - 更新效率低
- 前缀扩展
  - 图中a(0*)扩展成a(00*)以及a(01*)

基于大容量RAM的快速路由查找算法

Table16表项结构

0	路由转发信息
1	TableNext表指针

TableNext表项结构

路由转发信息

表Table16

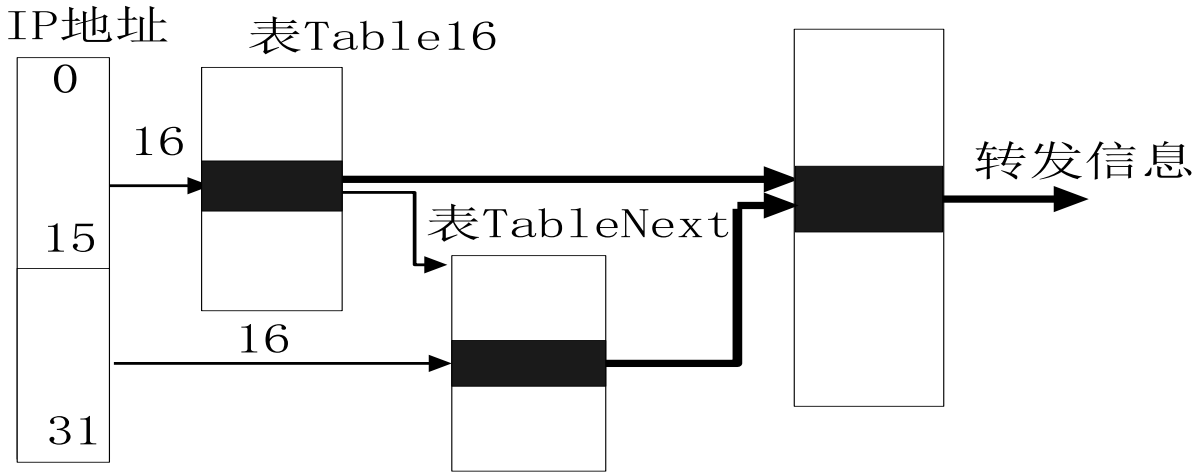
...		...
165.255	–	– –
166.0	0	A
166.1	0	A
...		...
166.110	0	A
166.111	1	
166.112	0	A
...		...
166.255	0	A
167.0	–	– –
...		...

表TableNext

0*256+0	B
...	..
0*256+255	B
...	..
68*256+0	C
...	..
68*256+255	C
69*256+0	B
...	..
69*256+255	B
...	..

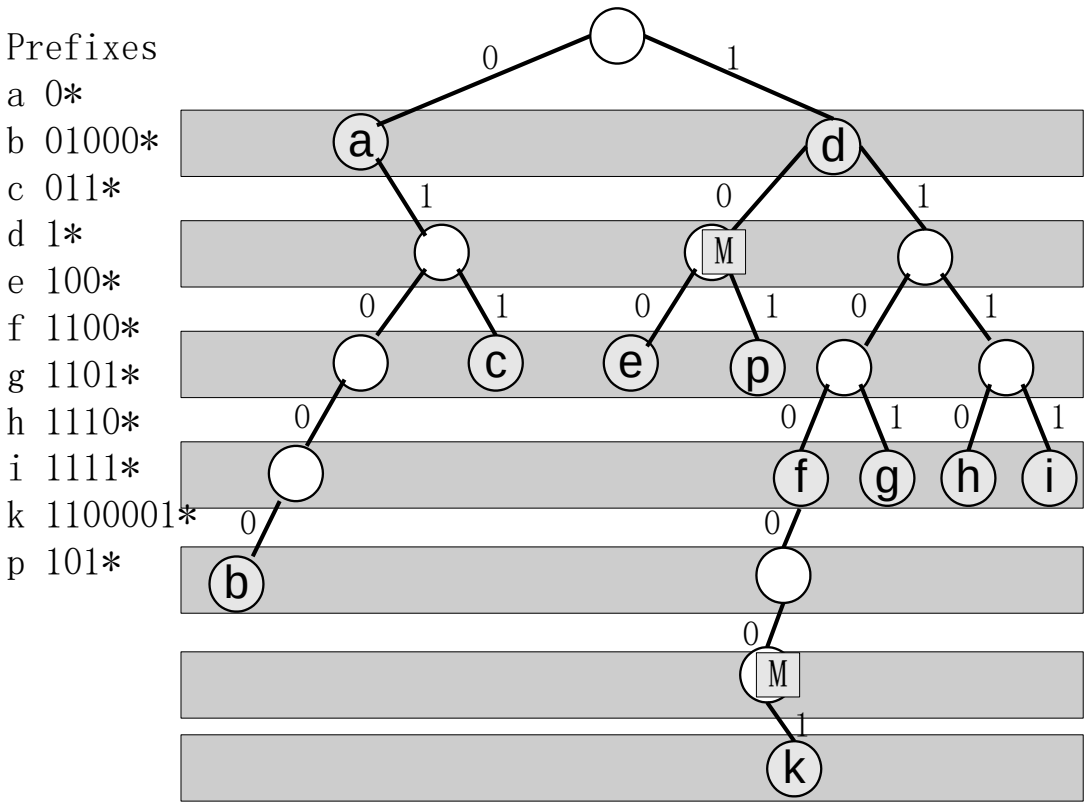
- 采用多分支trie树查找算法思想，trie树的深度为2，步宽分别为16/16（或24/8）。
- 算法采用两种查找表结构，分别为Table16和TableNext。Table16表相当于多分支trie树的第一层结点，它主要保存那些路由地址前缀小于等于16的表项；TableNext表相当于多分支trie树的第二层结点，主要保存那些路由地址前缀大于16的表项。
- 例如加入路由项166/8(A)，166.111/16(B)，166.111.68/24(C)，查找表如左图所示。
- 在查找表中查找地址166.1.2.3和166.111.1.2

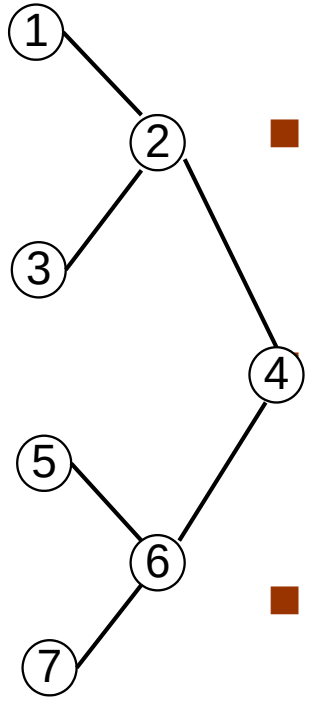
基于大容量RAM的快速路由查找算法



- 对于任何一个IP地址，查找过程最多只需要访问两次查找表（Table16和TableNext）就可以得到转发信息，大大加快了路由查找的速度。
- 如果我们在实现中将Table16和TableNext表从物理上分开（比如说采用两个不同的RAM），那么访问不同的表就可以同时进行，从而可以使用流水线查找。在查找过程完全满足流水线操作的条件下，查找只需要一次存储器访问，达到了查找算法的最高性能。

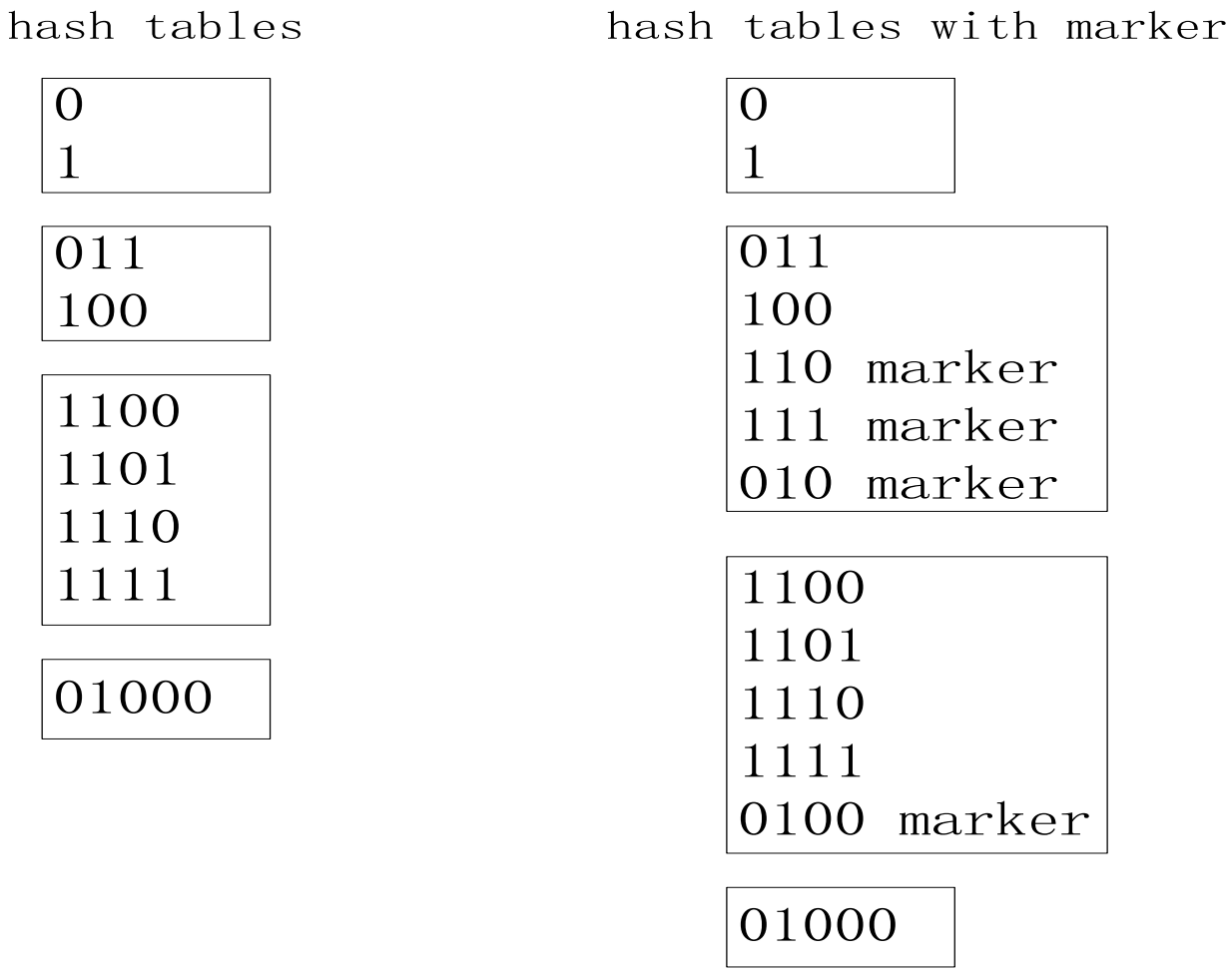
地址前缀长度的二分查找法



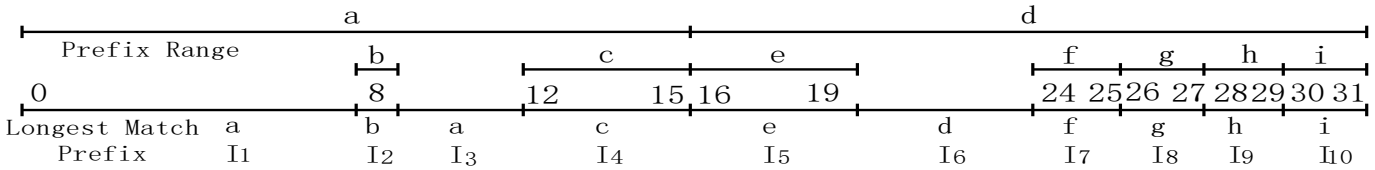


- 在地址前缀空间内顺序遍历前缀(前缀长度从小到大(binary trie)或从大到小)的算法复杂度为O(W)
- Waldgoel提出了在地址前缀长度空间内进行二分查找的思想，算法复杂度为O(logW)
- 所有地址前缀根据前缀长度分成32个(IPv4)不同的前缀集S = {S0, S1, …, S32}。
- 需要在二分查找树中增加称为Marker的表项

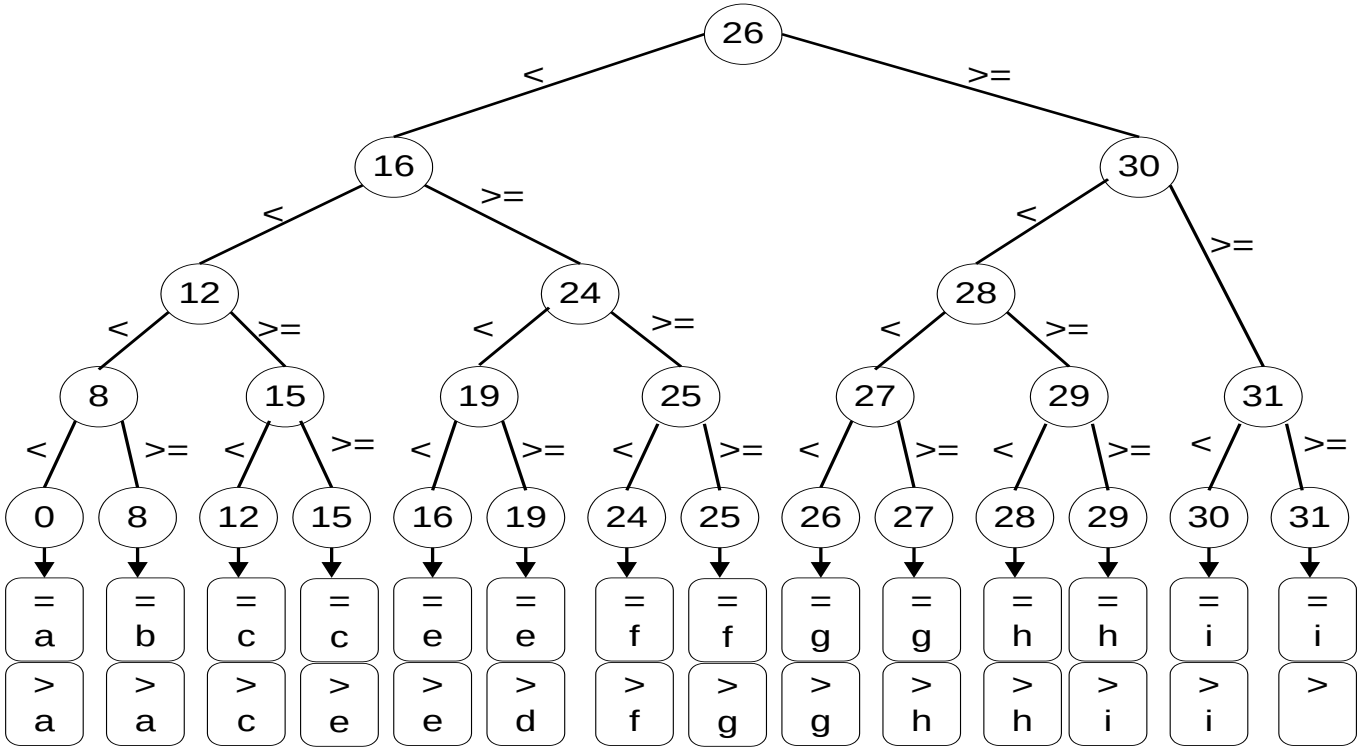
地址前缀长度的二分查找法



地址区间的二分查找法

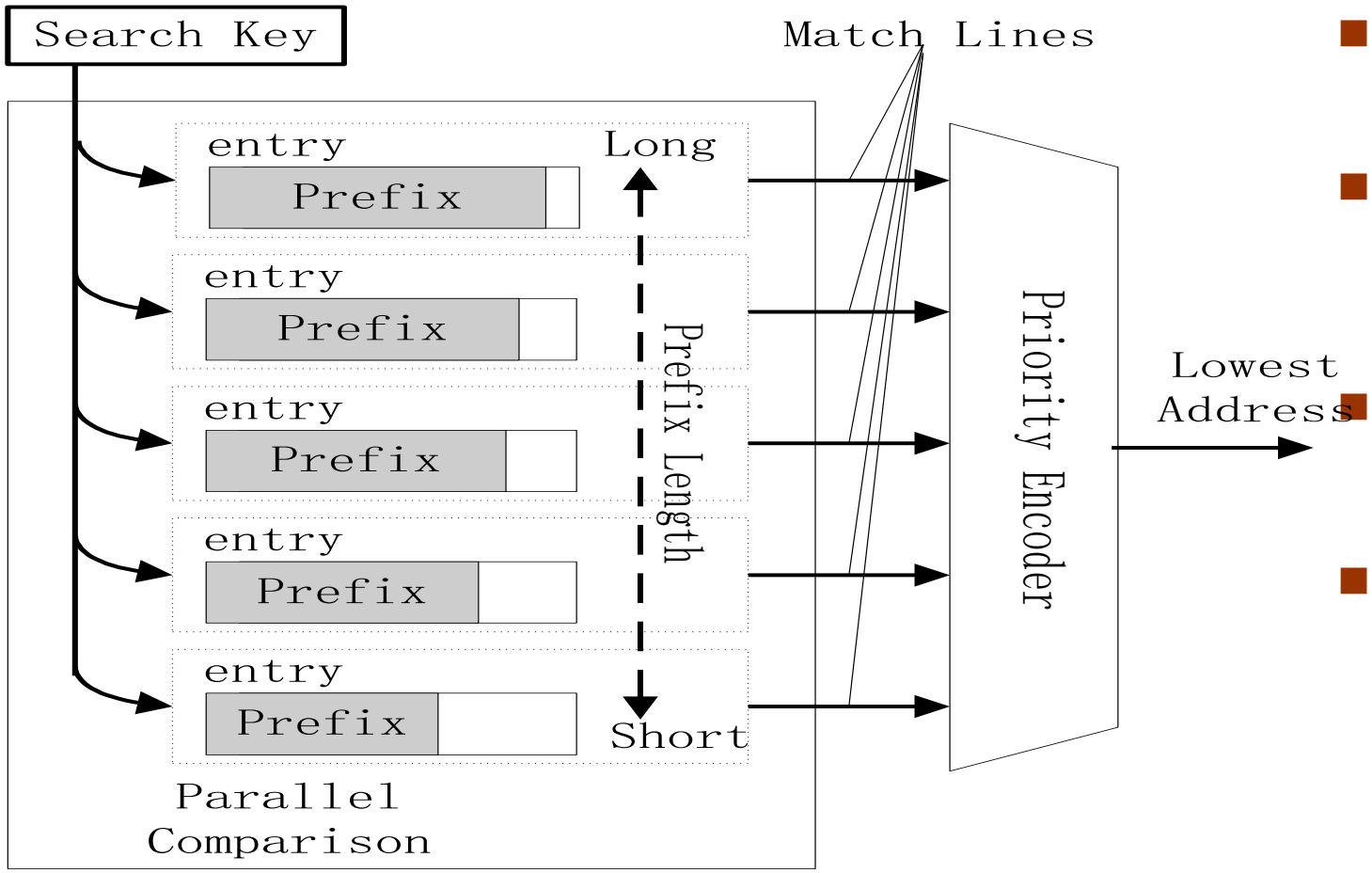


前缀对应的地址区间图



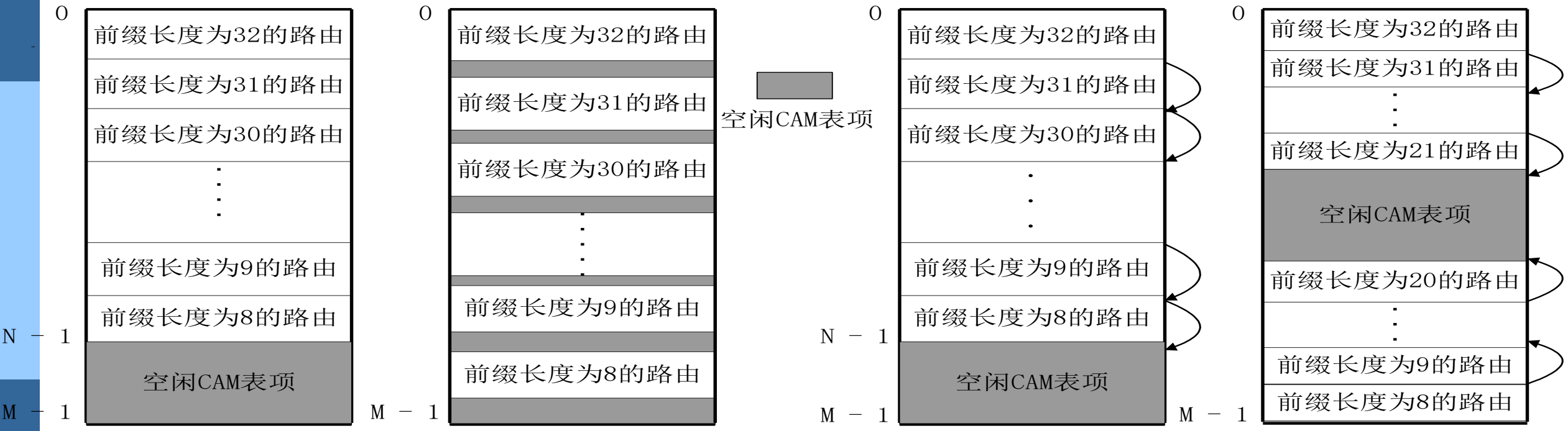
- 地址前缀在整个地址空间内代表了一段连续的地址区间
- 任何地址区间对应的最长前缀应该是包含此区间的前缀中地址范围最窄的一项
- 例：10110(22)
- N个地址前缀，查找算法复杂度为O($\log_2 2N$)
- 改进：多路查找算法，复杂度降为O($\log_k 2N$)
- 区间二分查找法的最大问题是地址前缀的更新

TCAM (Ternary Content Addressable Memory)



- 在一个硬件时钟周期内完成关键字的精确匹配查找
- TCAM规定在所有匹配的表项中选取地址最低的表项作为最后结果，因此需要保证在低地址存储前缀较长的前缀
- 优点
  - 查找速度快(15-20ns)
- 缺点
  - 单位容量的芯片价格高
  - 功耗较大
  - 更新效率低

TCAM路由更新



顺序移动法 -- O(N)　　预留表项空间的顺序移动法 -- O(N)　　选择移动法 -- O(W)　　改进的选择移动法 -- O(W/2)

N 为目前TCAM 中保存的表项数目
W为路由前缀长度

路由查找算法的评价标准

- 查找速度（Speed)
- 存储容量（Storage）
- 预处理和更新速度(Preprocessing and Update Speed
- 算法实现的灵活性（Flexibility in Implementation）
- 算法的可扩展性（Scalability）

几种路由查找算法性能比较

算法	查找过程	存储容量	更新过程	其他特点
二分支trie树	O(W)	O(NW)	O(W)	
路径压缩trie树	O(W)	O(N)	O(W)	
多分支trie树(步宽k)	O(W/k)	$O(2^kNW/k)$	$O(W/k+2^k)$	可以硬件实现
前缀长度的二分查找	O(logW)	O(NlogW)	O(NlogW)	适合IPv6查找
前缀区间的二分查找	O(log2N)	O(N)	O(N)	适合IPv6查找
理想的查找算法	O(1)	O(N)	O(1)	适合IPv6查找并且可以硬件实现

实验背景介绍

- 距离向量
  - 什么是向量？Vector—table，每个路由器维护一张表，表中的内容是：到达目的地的最佳距离以及使用哪条outgoing line的信息，距离一般使用跳数作为评价。
  - 向量（表）的维护依赖于邻居之间周期性交换信息
  - RIP
  - The Count-to-Infinity Problem
- 链路状态
  - 由于前者的收敛慢,出现了链路状态算法
  - IS-IS，OSPF
  - 每个路由器都知道完整的网络拓扑，Dijkstra’s algorithm 找自己到其他路由器的最短路径即可

实验背景介绍

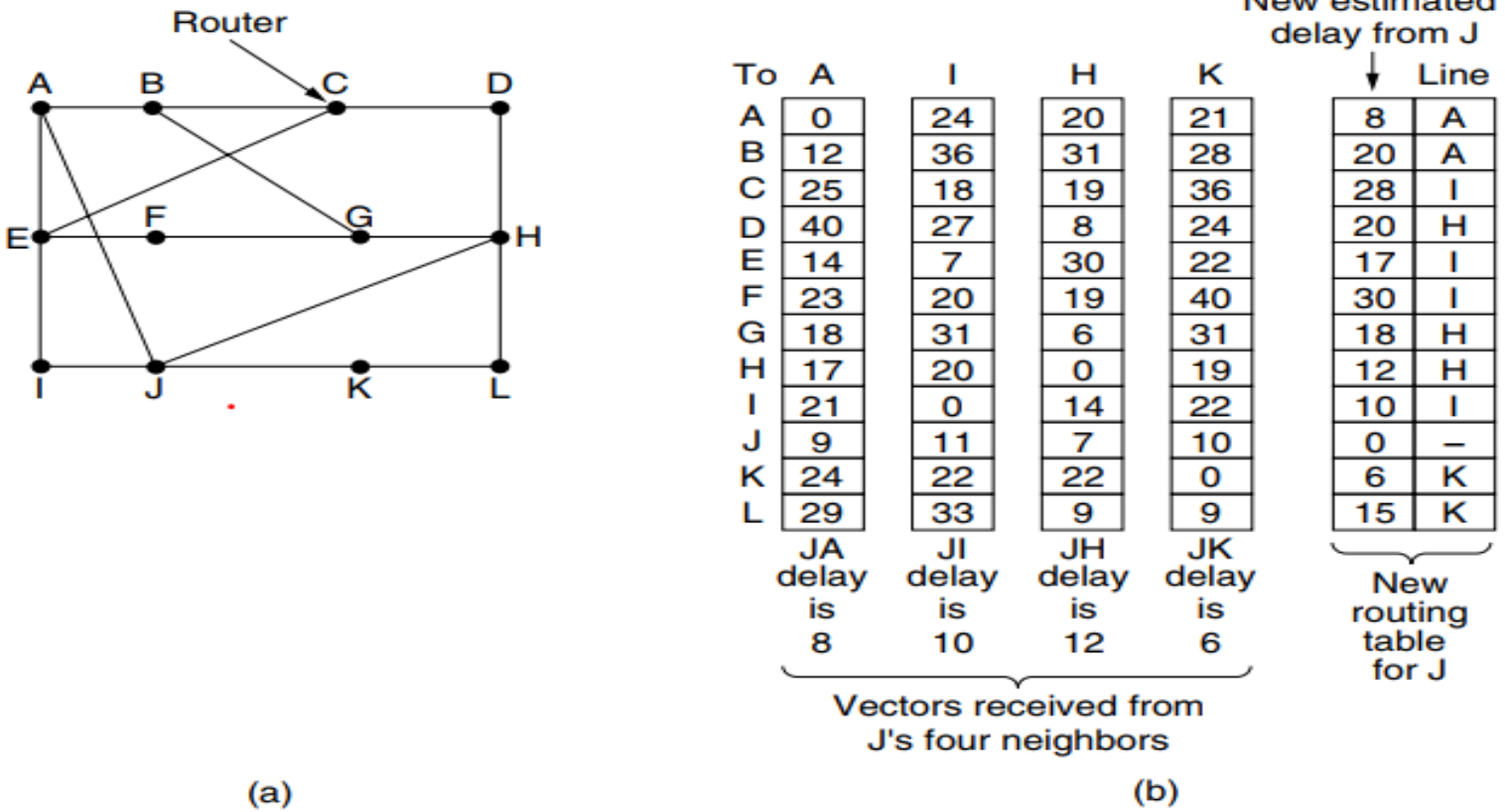
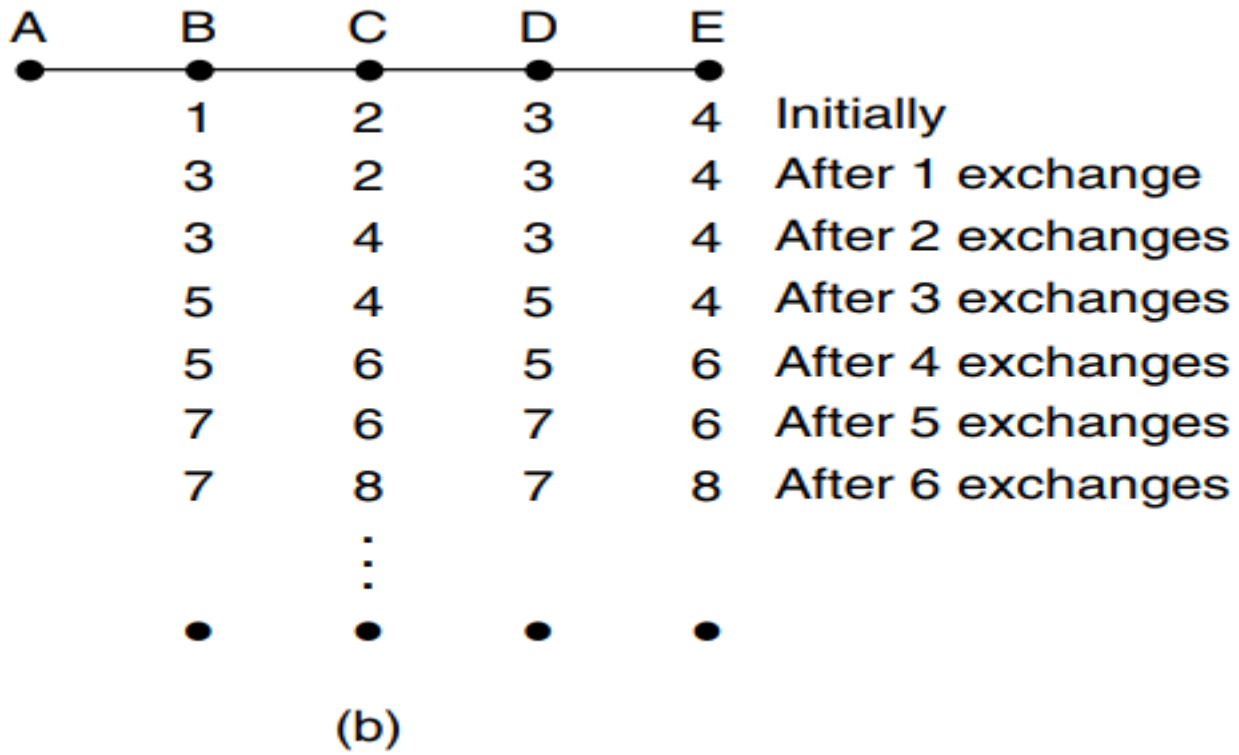


Figure 5-9. (a) A network. (b) Input from *A*, *I*, *H*, *K*, and the new routing table for *J*.

实验背景介绍

- The Count-to-Infinity Problem
  - 网络中的路由稳定到最佳状态---收敛



(b)

实验背景介绍

- The Count-to-Infinity Problem
  - RFC 1058
  - 水平分割
    - 在路由信息传送过程中，路由器从某个接口接收到的更新信息不允许再从这个接口发回去
  - 毒性逆转
    - 毒性逆转（Poisoned Reverse）实际上是一种改进的水平分割，这种方法的运作原理是：路由器从某个接口上接收到某个网段的路由信息之后，并不是不往回发送信息了，而是发送，只不过是将这个网段标志为不可达，再发送出去。收到此种的路由信息后，接收方路由器会立刻抛弃该路由，而不是等待其老化时间到（Age Out）。这样可以加速路由的收敛
  - 抑制计时器
    - 路由器如果在相同的接口上收到某个路由条目的距离比原先收到的距离大，那么启动一个抑制计时器。在抑制计时器的时间内该目的不可到达。

实验背景介绍

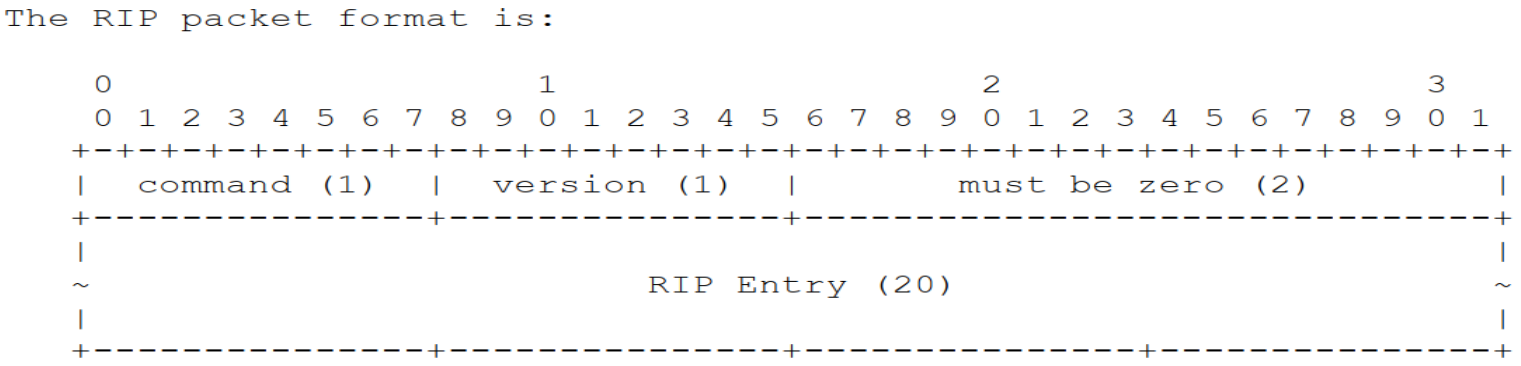
- RIP（Routing Information Protocol ）
  - 内部网关协议，动态路由协议，距离向量算法
- 实验实现RIP v2,RFC2453中的3.Basic Protocol
- 理解距离矢量算法---section 3.4
- RIP协议描述---section 3.5-3.10
  - 距离向量协议，使用hop count作为metric,最大支持15跳【1,15】，大于或等于16的跳数被定义为无穷大，即目的网络或主机不可达。由于这个限制，使得RIP不可能在大型网络中得到应用
  - 路由表表项：目的地址，metric,下一跳的地址，路由变化标志，各种与该路由相关的计时器
  - 消息格式
- 其中的4.Protocol Extensions不在要求范围内

实验背景介绍

- 基本工作原理
- 路由设备启动RIP之后，会向直连主机组播发送RIP请求报文，请求一份完整的路由表，同时对端路由也会发送RIP请求报文。
- 路由设备在接收到request请求报文后，会查询自身的RIP路由表，获取RIP路由表信息，然后修改命令为response，再组播发送回去，自身路由发送变化时，会通过更新路由组播发送给直连路由器

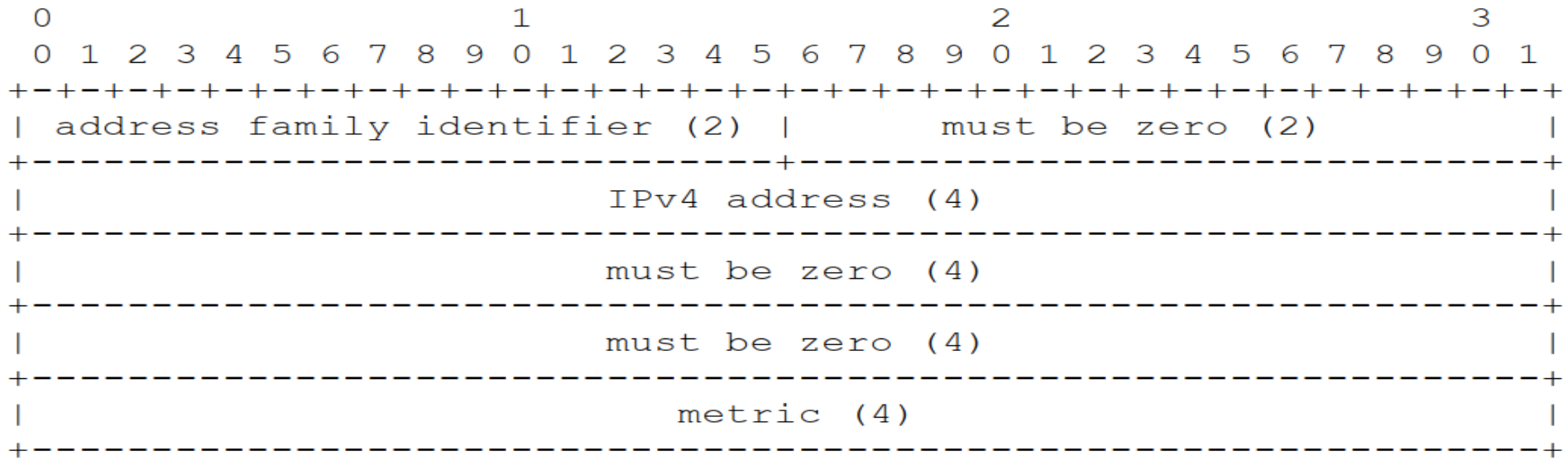
实验背景介绍

- 消息格式(RFC)
  - 基于UDP的协议
  - 运行RIP协议的路由器在UDP端口520上收发数据报文
  - 每个消息都包含一个RIP头(command and version)和多个路由表项，RIP表项的数量【1,25】
    - Command:该消息的目的
      - Request
      - Response:主动和被动
    - Version 1
      - 版本间的兼容以及处理见4.6和5.1，默认为2



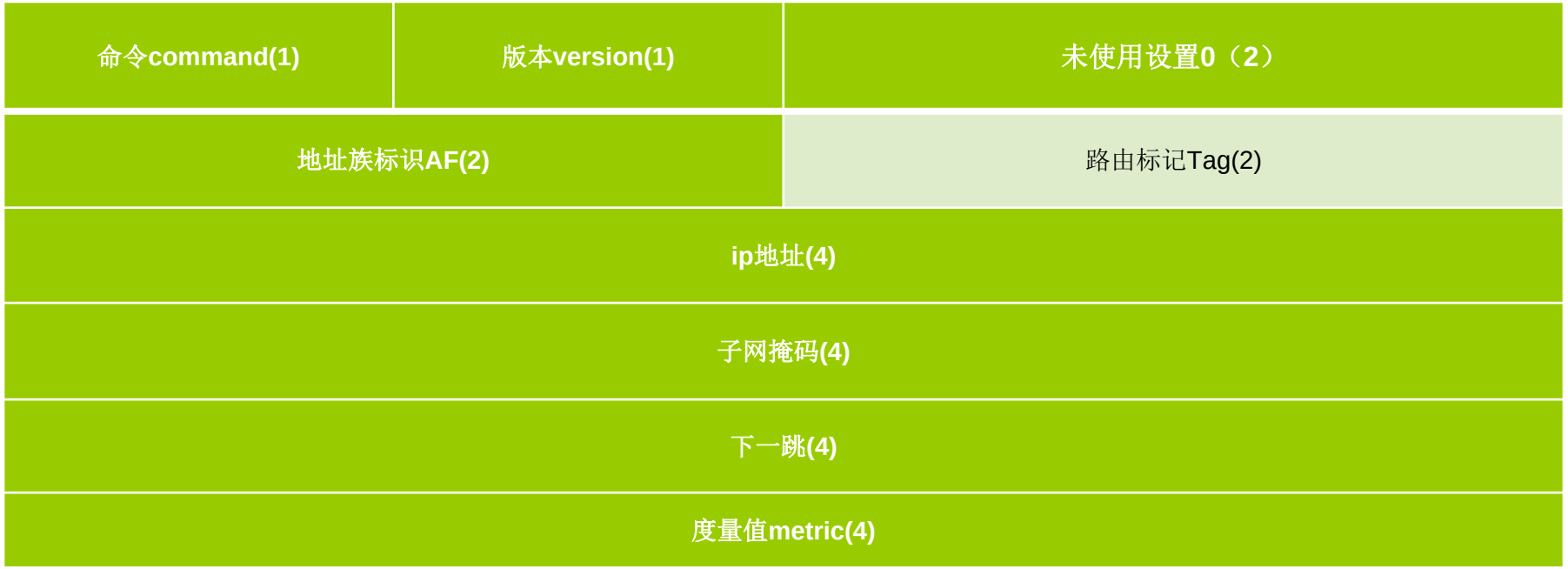
实验背景介绍

- 表项格式（RFC）
  - 地址族—AF_INET(2)，对于Request报文，此字段值为0
  - 目的IPv4地址，可以是网段或者主机
  - Metric—[1,15],16表示目的地不可达，对于Request报文，此字段值为16。



实验背景介绍

- command:定义了报文类型(response或者request)；
- version:RIP报文版本，默认情况下为2；
- AF:Address family identifier,发送请求报文时地址族默认为0；
- Tag:外部路由标记；
- IP地址:目标地址IP
- 子网掩码:用于区分子网；
- Next Hop:下一跳地址；
- metric:路由跳数，请求完整的RIP报文时该值为16；

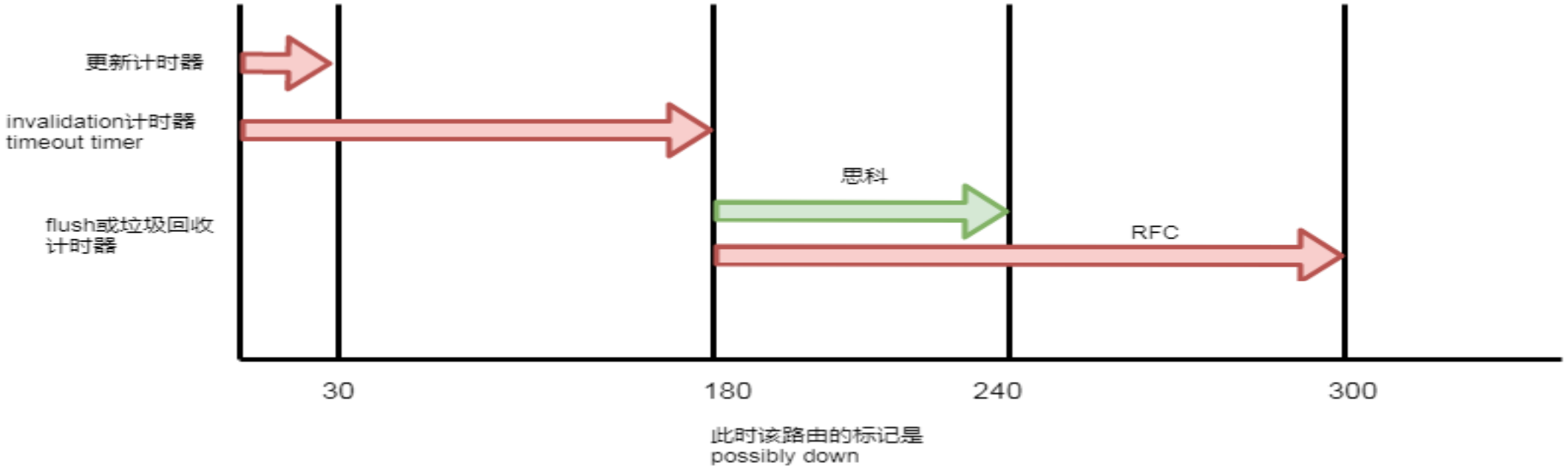


报文头部是4个字节，包括了“命令”字段1个字节、“版本”字段1个字节、未使用2个字节。路由表项是20个字节，其中地址族标识占2个字节，路由标记占2个字节，IP地址、子网掩码、下一跳、度量值各自占4个字节。一个完整的RIP报文可以最多25个路由表项

实验背景介绍

- 计时器
  - 更新计时器：每隔30秒，RIP进程被唤醒发送一个Response消息，其中包含全部路由表，给每个邻居路由器（考虑水平分割）
  - 每条路由都有2个计时器
    - Timeout（老化定时器，180s）:当过期之后，路由不再有效（该路由的度量值置为16），保留在路由表中一段时间，等待通知邻居该路由无效，启动垃圾回收定时器
    - Garbage-collection（垃圾收集定时器,120s）：如果垃圾收集定时器超时，设备仍然没有收到更新报文，完全从路由表中移除
    - 每次收到路由更新时，该路由的计时器初始化
  - 触发更新计时器
    - 限制路由触发更新的频率，保护网络

实验背景介绍


更新计时器
invalidation计时器
timeout timer
flush或垃圾回收
计时器
思科
RFC
30
180
240
300
此时该路由的标记是
possibly down

实验背景介绍

- 输入处理
  - 根据收到的报文的command字段决定处理流程
  - 请求消息
    - 如果请求消息中只有一条entry，其地址族是0，metric是16，请求发送整张路由表
    - 逐项检查请求消息中entry，查询路由器中的对应目的地址的路由信息，告知请求方，如果没有，metric域填infinity16
  - 响应消息
    - 收到响应消息的3种情况
      - 对于特定查询的响应
      - 例行更新
      - 由于路由变化导致的触发更新
    - 收到响应消息谨慎处理
      - 不是来自RIP端口直接忽略
      - 源地址是否是有效的邻居—必须是直连网络
      - 检查是否来自自己的地址，广播/组播

实验背景介绍

- 响应消息
  - 检查完来源合法之后，逐个检查表项
    - 检查目的地址是否有效，例如，单播而不是net 0 或者 127
    - Metric是否有效
  - 之后就是计算路由，metric = MIN (metric + cost, infinity)
  - 检查是否存在目的地址路由的表项
    - 若存在
      - 比较2者
        - 首先查看这两条路由信息是否来自同一个网络接口（下一跳相同）重新初始化计时器
        - 之后，比较metric，如果是来自同一个邻居且新的metric，采取新路由，将route change flag设置为1，触发输出程序触发更新；若新metric是无穷大，直接进行删除
        - 若新metric等于旧metric，最简单的就是什么都不做
        - 若来自不同的邻居，比较metric大小进行选择
    - 若不存在
      - 添加该路由，除了填充必要字段之外，将route change flag设置为1，触发输出程序触发更新

实验背景介绍

- 输出处理
  - 创建响应消息的触发
    - 收到请求
    - 例行路由更新
    - 触发更新
      - 路由发生变化之后，触发的更新，route change flag set
      - 限制更新频率—一个触发更新发送之后，随机计时器（1-5秒）开始工作，在这段时间内即使发生变化，不再进行触发更新，直到计时器过期；如果定期更新在发送触发更新时到期，则应禁止触发更新
      - 如果在输出生成的过程中，可以进行输入处理，必须进行合适的互锁，在触发更新生成的过程中，route change flag不应受到输入处理的影响而变化
  - 生成响应消息
    - 如果是对request的响应，版本要一致
    - 填充response消息的各个字段，注意大小有限制，最多25个entry
    - 即使路由的metric是infinity，响应消息中也要包含。

实验背景介绍

- 转发引擎
- 基本转发功能
  - IP头检查：检查未通过的包直接丢弃，IP协议版本号，包的头长度，检查校验和
  - 包生命周期控制：决定包的TTL，防止陷入环路，如果TTL小于等于0，丢包，生成ICMP包
  - 校验和重新计算
  - 路由查询：数据包中的目的地址用来在转发表中查找对应的output port，此搜索的结果将指示数据包的目的地是路由器还是输出端口（单播）或多个输出端口集（多播）。
  - 分片：转出链路的最大传输单元（MTU）小于传输的数据包的大小

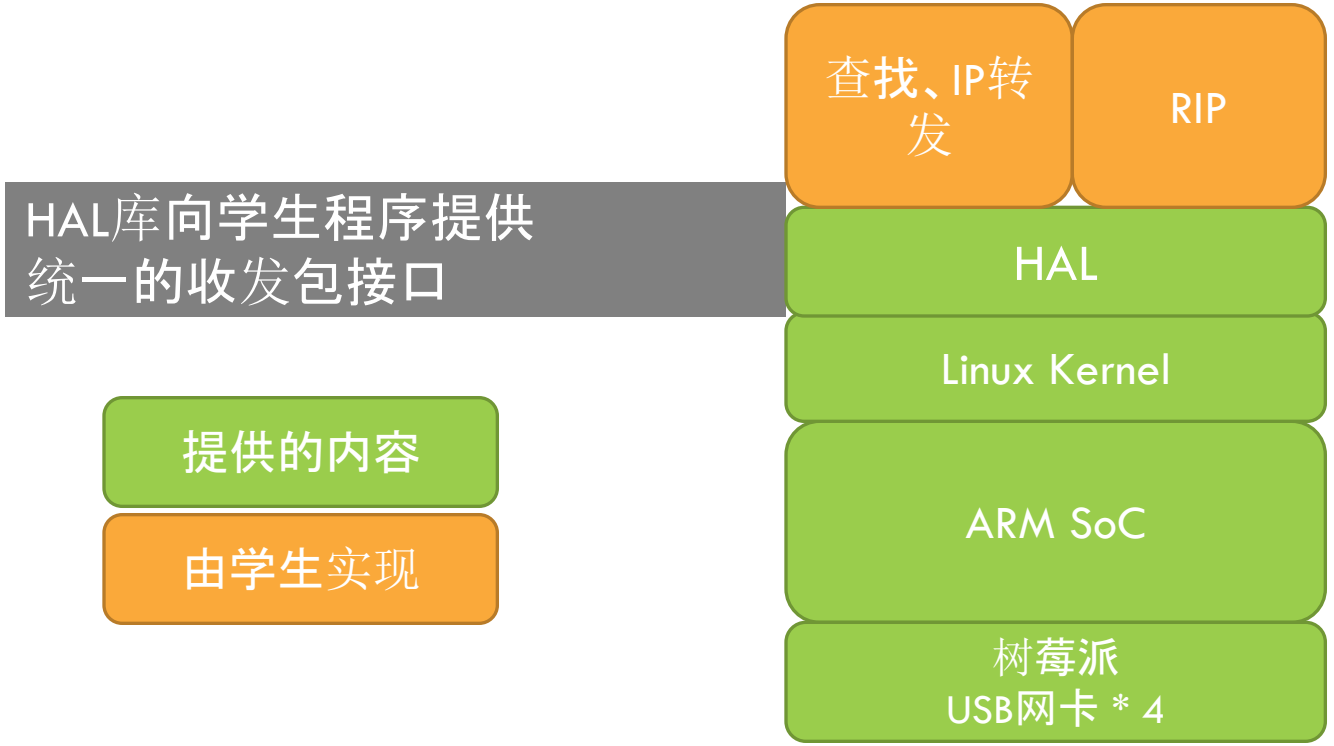
实验目的

- 设计实现运行RIP协议的路由器
  - 路由部分：RIP协议
  - 转发部分：实现转发引擎
- 实验预期效果
  - 正确实现RIP路由协议的基本功能，通过实验理解掌握路由协议
  - 转发引擎正确设计和实现
  - 实现简单组网，运行RIP协议的路由器可以进行组网，完成路由的正常功能
  - 性能测试：查表性能测试

实验方案

- 树莓派（4个USB网卡）+ Linux + HAL
  - 独立完成（单人组）
  - 提供内容：树莓派，HAL库文档及源码
  - 实验要求：软件实现IP转发、路由查找、RIP路由协议，在硬件上测试通过，实现3台以上路由器组网

实验系统架构


查找、IP转发
RIP
HAL库向学生程序提供
统一的收发包接口
HAL
Linux Kernel
提供的内容
ARM SoC
由学生实现
树莓派
USB网卡 * 4

HAL

- 开源项目
- https://freedesktop.org/wiki/Software/hal/ 别人的HAL
- 动机
  - 硬件种类和复杂性增多，由Unix内核提供的抽象不足以满足
  - 现代总线支持热插拔和复杂拓扑，设备的活动难以被系统调用追踪
  - 为桌面环境下用户提供用户友好的硬件接口，允许桌面应用通过一套简单的API发现和使用宿主系统的硬件，而无需关注其底层的硬件的类型。
- 简介
  - HAL: hardware abstraction layer,unix-like 操作系统中用于提供硬件抽象的软件子系统
  - 传统上，操作系统内核负责提供硬件的抽象，应用使用system call接口或者进行文件I/O与硬件进行通信
  - HAL用于发现，枚举和作为中介对主机上硬件的访问

我们为大家提供的HAL库

- 旨在为学生程序提供平台无关的通用编程接口

  主要功能：IP分组收发接口、ARP状态维护、ARP表查询、网口状态查询

- 单线程工作，轮询模式，无需CPU中断
- 通过编译选项支持多种运行模式

  仿真模式：调用libpcap读写pcap格式，从输入文件中接收网络数据，并把发送的数据写入输出文件

  Linux模式：调用libpcap通过BPF接收数据包，绕过系统原有协议栈，解决学生反映的raw socket行为不可控的问题

  Bare-metal模式：访问Xilinx EMAC IP，收发以太网帧

HAL库

- ARP：address resolution protocol,在以太网中，同一局域网中的主机要和另外一台主机进行通信，必须知道目的主机的MAC地址；IP中只关注目的主机的IP地址，在以太网中使用IP协议时，数据链路层的以太网协议接到上层IP协议提供的数据中，只包含目的主机的IP地址。于是需要一种方法，根据目的主机的IP地址，获得其MAC地址。这就是ARP协议要做的事情。地址解析（address resolution）就是主机在发送帧前将目标IP地址转换成目标MAC地址的过程
- Libpcap—抓取数据包的库，wireshark的基础
- BPF：Berkeley Packet Filter，是Unix-like操作系统上数据链路层的原始接口，提供原始链路层封包的收发
- Bare-metal:裸机

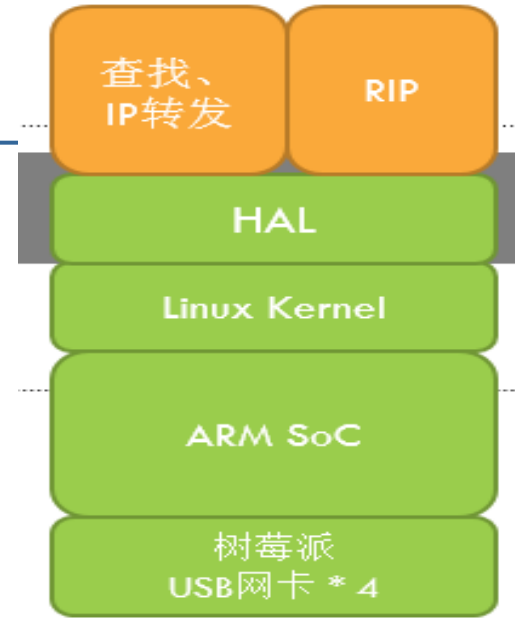
HAL库

- HAL_Init: 使用 HAL 库的第一步，必须调用且仅调用一次，需要提供每个网口上绑定的 IP 地址
- HAL_GetTicks：获取从启动到当前时刻的毫秒数
- HAL_ArpGetMacAddress：从 ARP 表中查询 IPv4 地址对应的 MAC 地址，在找不到的时候会发出 ARP 请求
- HAL_GetInterfaceMacAddress：获取指定网口上绑定的 MAC 地址
- HAL_ReceiveIPPacket：从指定的若干个网口中读取一个 IPv4 报文，并得到源 MAC 地址和目的 MAC 地址等信息
- HAL_SendIPPacket：向指定的网口发送一个 IPv4 报文

本地自测

- 本地自测：
  - git clone https://github.com/z4yx/Router-Lab.git
  - 所需依赖：make, g++, python3, libpcap
  - cd Homework/checksum
  - Make
  - 如果编译成功，按照要求编辑 Homework/checksum/checksum.cpp 文件，运行 python3 grade.py 进行测试。如果没有问题，可以把 checksum.cpp 作为本题的答案提交上来。

树莓派——环境配置



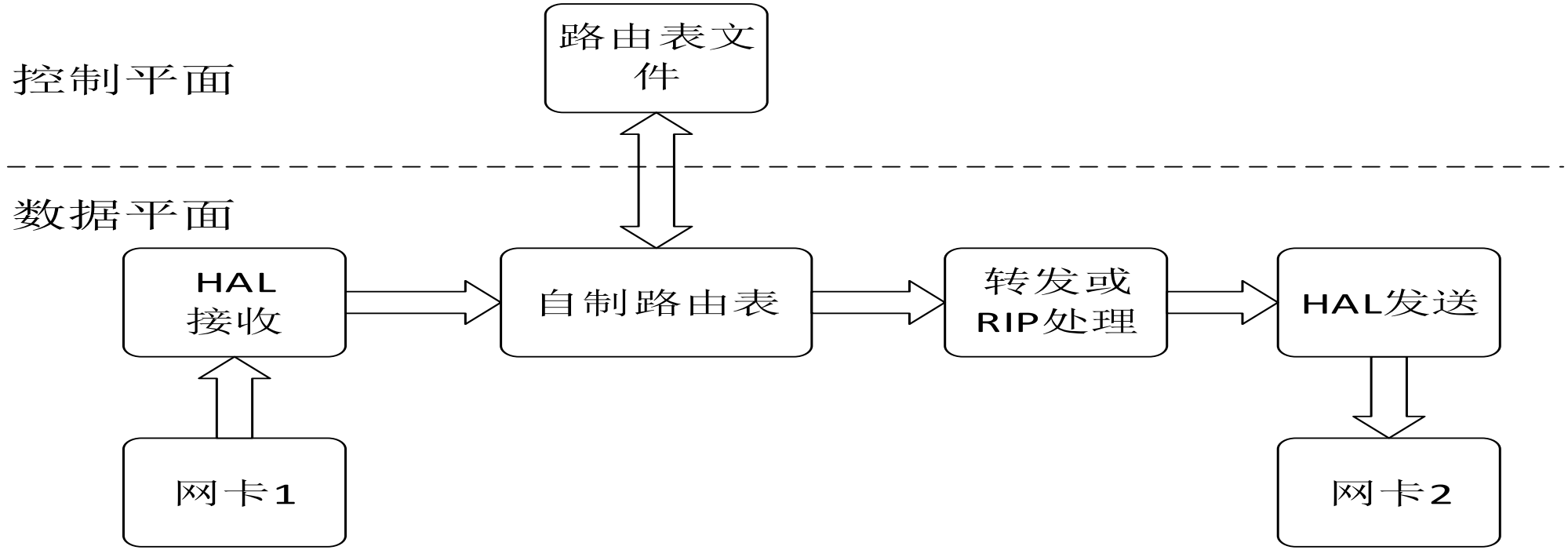
- 树莓派：基于linux的单片机电脑
- 环境配置烧录 Raspbian 系统
  - https://www.raspberrypi.org/downloads/raspbian下载 Raspbian 系统
  - SD 卡插入读卡器连接电脑，用 SD Card Formatter 软件格式化 SD 卡
  - 用 Win32 Disk Imager(windows) 或 ApplePi-Baker(macOS) 将下载的 img 文件烧录入 SD 卡中
- PC 登录树莓派
  - 进入 PC 的 System Preference->Sharing，设置网络共享到 USB LAN
  - 使用网线连接 PC 与树莓派，使用 arp -a 指令查询树莓派 ip 地址
  - 使查询用获得的树莓派的 ip 地址，通过 ssh 或 VNC Viewer 登录树莓派 (用户名为 pi，初始密码为 raspberry)

树莓派——环境配置

- 树莓派初次运行配置配置
  - 参考博客https://www.jianshu.com/p/67b9e6ebf8a0更换国内软件源，注意更换后运行 sudo apt-get update 更新
  - su root 切换到 root 用户，raspi-config 进入树莓派的高级配置面板
    - 扩展 SD 卡可用空间：选项 7.Advanced Options -> A1.Expand Filesystem 确认后 TAB键切换 <Finish> 保存并退出
    - 更换软件源：8.Update 耐心等待
  - 固定 ip 地址：编辑/etc/dhcpcd.conf，参考注释模板，添加配置网卡的固定 ip，设置了静态 IP 后，PC 便可通过固定 IP，用 ssh 或 VNC Viewer 等其他工具登录树莓派
  - 针对 VNC Viewer 登录树莓派桌面的桌面窗口分辨率过小问题：编辑/boot/config.txt，解除对 framebuffer_width framebuffer_height 的注释如下所示即可

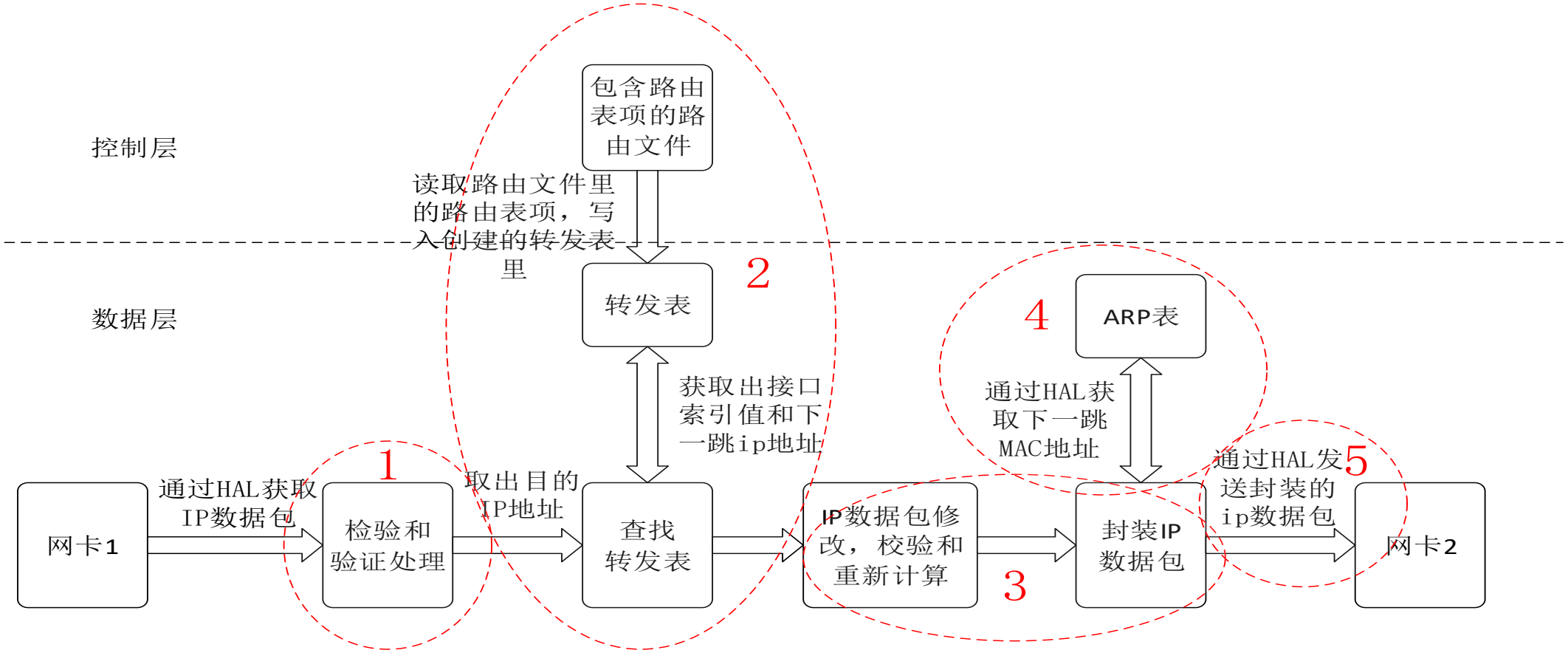
```
# uncomment to force a console siz e .
# By defaultitwill be display 's size minus overscan .
framebuffer_width=1280
fram ebuff er_h eight=720
```

总体说明



转发引擎

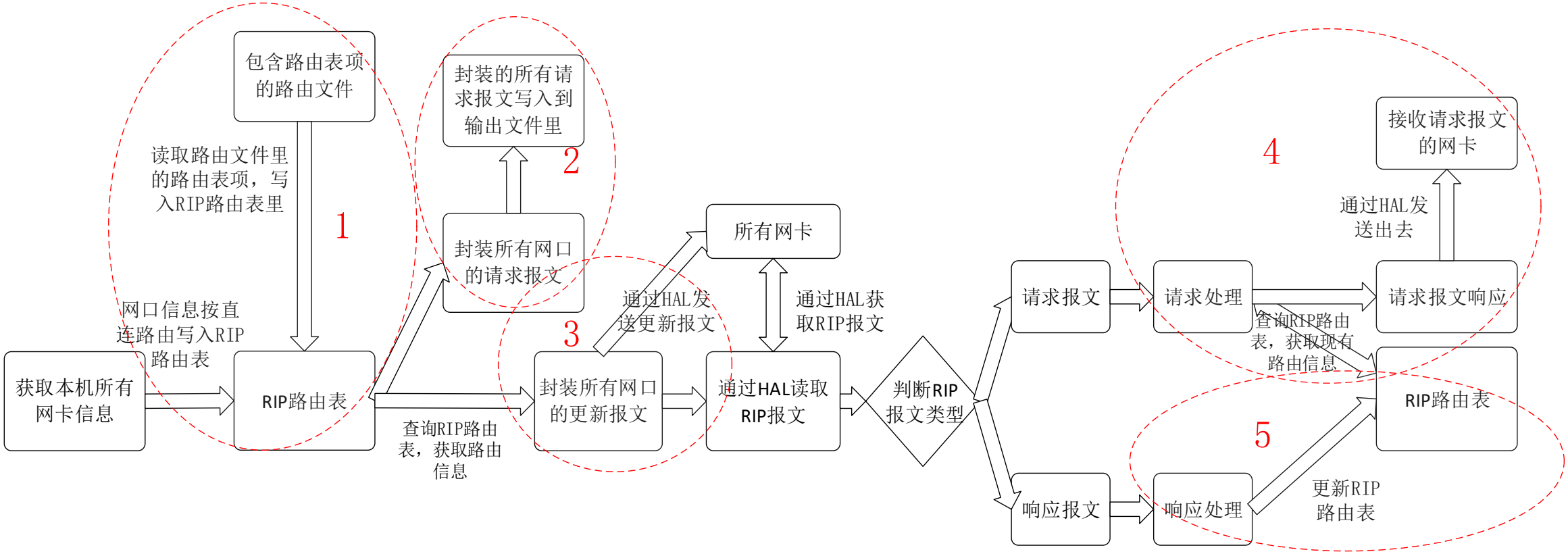
基于HAL真实网络形式的转发流程，红圈标注部分需要依次完成



转发引擎

1. 填写validateIPChecksum函数，对IP数据包的IP首部依次进行校验和验证，正确的返回ture,错误的返回false，主函数根据validateIPChecksum函数返回作出相应处理，返回true则继续,返回false则丢弃数据包
2. 第2步
    1. 获取IP数据包的目的ip地址；
    2. 创建转发表，读取路由表项文件，将路由表项写入自己创建的转发表，转发表的数据结构选用tree树、链表、哈希等其中之一；
    3. 根据目的ip地址查找转发表，获取正确的下一跳信息
3. 第3步
    1. 将IP数据包的IP首部TTL值减一，重新计算校验和
    2. 重新封装IP数据包；
4. 根据查找到的下一跳信息调用HAL_ArpGetMacAddress函数获取下一跳的MAC地址；
5. 调用HAL_SendIPPacket函数将重新封装的IP数据包发送出去

基于HAL真实网络形式的RIP



基于HAL真实网络形式的RIP

- 创建RIP路由表
  - 读取路由表项文件，将路由表项写入自己创建的RIP路由表；
  - 将本机网卡信息按直连路由写入到RIP路由表。
- 请求报文封装及发送
  - 封装IP头：源ip为本机网口ip，目的ip为组播ip地址224.0.0.9；
  - 封装UDP头：源、目的端口都为520；
  - 封装RIP请求，command字段为1，version字段等于2，其他字段等于0，度量值字段等于16；
  - 将封装的所有网口请求报文调用HAL_SendIPPacket函数从各自网口发送出去。
- 更新报文封装及发送（更新报文与接受RIP报文放在同一循环里）
  - 封装IP头：源ip为现有本机网口ip，目的ip为组播ip地址224.0.0.9；
  - 封装UDP头：源、目的端口都为520；
  - 封装更新RIP路由条目，遍历rip路由表，将除了与源ip在同一网段和下一跳为源ip地址的路由都封装到路由条目里；
  - 调用HAL_SendIPPacket函数从所有网口转发出去。

基于HAL真实网络形式的RIP

- 请求报文处理：
  - 检验是否为请求报文，判断度量值是否为16，地址族标识是否为0；
  - 获取请求报文的源ip地址；
  - 遍历RIP路由表，封装所有和源ip地址不在同一网段的路由表项到RIP报文里，填充Rip报文，command为字段2；
  - 封装成UDP的IP数据包，源ip为本机接收请求报文网口的ip地址，目的ip为请求报文的源ip，调用HAL_SendIPPacket函数从接收请求报文的网口发送出去。
- 响应报文处理：
  - 按照RIP表项的数目对RIP报文进行依次处理；
  - 如果度量值字段加1后大于16，表示删除这条路由，遍历rip路由表，删除和RIP报文对应的路由，同时将失效（删除的）路由封装成失效报文，调用HAL_SendIPPacket函数从所有网口发送出去（接收失效报文的网口不发送）；
  - 对比rip路由表，没有相同的路由则直接把这条RIP报文里的路由插入rip路由表；
  - 对比rip路由表，有相同的路由，如果RIP报文里路由的度量值加1后小于或等于rip路由表里的度量值，更新rip路由的原来度量值为RIP报文里路由的度量值加1；若大于的则忽略。

开发工具

编译环境

通用平台实验：Linux GCC，树莓派系统内置

测试环境

提供实验板，多机互联测试

HAL框架使用说明

- 基于HAL读写文件形式：
  - 进入Homework目录下，是需要测各项功能的文件，eg：checksum，进入文件里，make编译，执行./grade.py即可自测
- 基于HAL真实网络形式：
  - 进入测试功能文件里后，需要修改Makefile里hal.o生成的依赖，将stdio改成linux，然后make编译，执行./grade.py即可自测。

自测指南

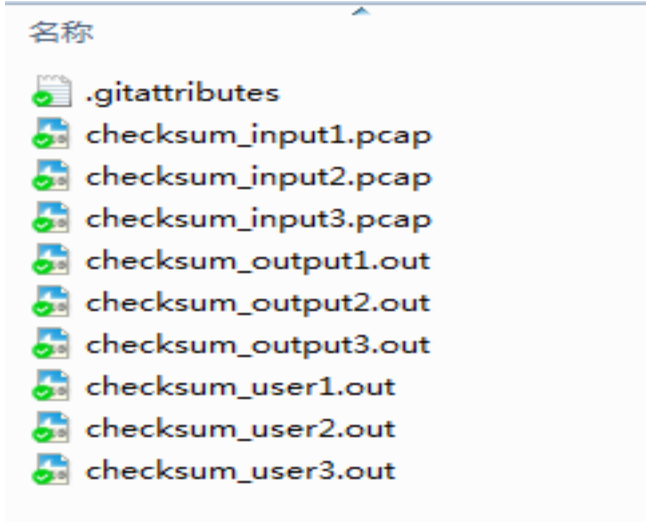
基于HAL读写文件形式

data文件里的input1为输入文件，output1为标准文件，程序输出后会生成输出文件user1；

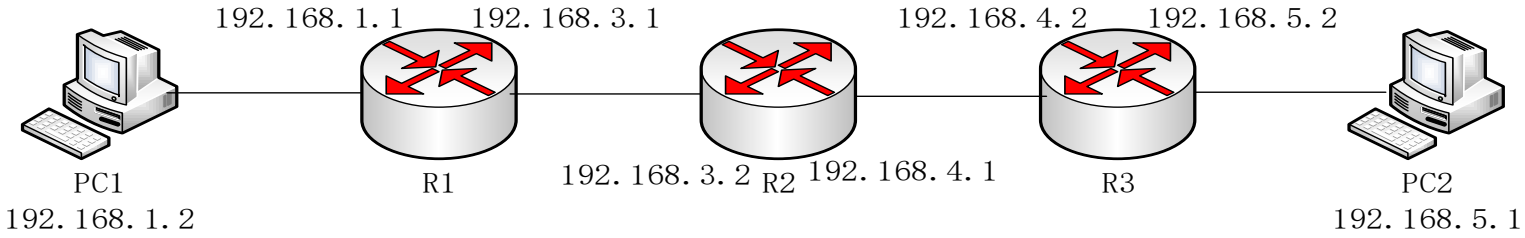
各个验证出错的话，程序框架会有错误提示，会告知文件第几个数据包处理错误，错误在哪，正确是什么，根据提示修改错误。

Grade后面标注是否通过，正确是 Grade: 1/1，错误是Grade: 0/1。

```
[root@localhost 1checksum]# ./grade.py
Removing all output files
Running './checksum < data/checksum_input1.pcap > data/checksum_user1.out'
Grade: 1/1
[root@localhost 1checksum]#
```



组网



<1>.RIP建联

按照网络拓扑图搭建环境，使3台路由之间能够正确学到RIP路由，通过抓取RIP报文来分析路由是否正确。

<2>.RIP、转发引擎的集成

按照网络拓扑图搭建环境，使3台路由能正常交互，将学到的RIP路由正常同步添加、删除、更新到转发表。

<3>.组网成功现象

3台路由各自启动，等待相互学习过程；PC1 ping PC2，可以ping通，反过来PC2也可以ping通PC1：

关于附加题加分——树莓派路由实验

- 感兴趣并且时间充裕的同学，可选择课后完成附加题
  - 如果能满足以下要求，实验课能拿到30分

实验课成绩（20分）+ 课堂表现分（10分）

1. 课上每个**验证性**实验项目的完成分都达到80%以上；
2. 本次附加题提交完整项目代码；
3. 本次附加题均提交完整的项目分析报告，内部包含对附加题目的内容分析、设计方法、算法分析等，格式不限；
4. 由老师审核项目代码和分析报告，并**通过验收答辩**，确认附加题完成情况达到要求。
5. 验收答辩时间：第17周（待定）

Q&A