Router Lab Report

Environment

环境采用了助教提供的 UTM 虚拟机，没有进行额外的配置。

Code Implementation

Routing Table

路由表采用最长前缀匹配算法。具体来说就是同时对目的地址和路由表项中的地址与掩码做按位和，如果得到的结果相同则说明匹配；每次匹配后再比较掩码和上次掩码的大小，如果更大则说明匹配到了更长的前缀。由此便可得到对应的 Interface。

Arp Cache

在 Arp Cache 的实现中，`periodicCheckArpRequestsAndCacheEntries`函数定期检查并处理ARP请求和ARP缓存条目。对于等待ARP回复的请求，如果尝试发送次数未超过上限(5次)，则再次发送ARP请求；若已达到上限则对无法解析的目标IP对应的源数据包发送ICMP主机不可达消息(ICMP type=3, code=1)，并删除该请求。同时，该函数还会清理无效的ARP缓存条目，从而保持ARP缓存和请求列表的有效性和及时性。

Router

核心内容为实现函数`handlePacket`，具体逻辑如下：

- 首先确认数据包大小是否足够包含以太网头部(ethernet_hdr)。如果数据包过小(小于以太网头部长度)，就输出错误信息并返回。
- 从数据包中提取以太网头部，并获得：
    - 源MAC地址(src_mac)
    - 目的MAC地址(dst_mac)
    - 以太网类型字段(ether_type)，用于区分ARP和IP数据包。
- 根据以太网类型字段进行分支处理
    - 处理ARP数据包(ethertype_arp)
        - 如果是ARP请求(arp_op_request)且目的IP刚好是路由器接口的IP地址，说明对方在请求本路由器的MAC地址。调用`sendArpReply`生成并发送ARP应答，让请求方获知本路由器对应IP的MAC地址。
        - 如果是ARP应答(arp_op_reply)，则从ARP回复中获取映射的MAC-IP条目。查询本地ARP缓存是否已存在该IP的请求条目。如果ARP缓存中没有记录，但有等待该IP地址的请求队列(ArpRequest)，则插入新的ARP缓存条目，并对所有在此请求上等待的包进行处理(使用请求中记录的封装方式再次递归调用`handlePacket`)。处理完后移除对应的ARP请求。
    - 处理IP数据包(ethertype_ip)
        - 如果目标IP就是本路由器，说明数据包是发给路由器本身的。根据IP协议字段判断数据包类型：
            - 如果是ICMP协议类型的IP包，调用`sendIcmp`进行处理和响应
            - 如果是TCP(0x06)或UDP(0x11)等其他协议号且并非路由器可处理的协议：发送ICMPType3目的不可达消息，指示无法处理该协议。

        - 在处理这些包前，如果ARP条目不存在，需要先对源IP发送ARP请求(通过`m_arp.queueRequest`)将数据包暂存，待ARP解析完成后再处理。
    - 如果目标IP不是本路由器，说明需要转发该数据包。首先检查IP头部中的TTL字段：
        - 如果TTL <= 1，表示该数据包已经无法继续向前转发，需要发送一个ICMP时间超时报文(Type 11，Code 0)告知源主机数据包已超时。同样，在发送ICMP超时报文前需要确保ARP缓存中有对应源IP的MAC地址，否则再次排入ARP请求队列。
        - 若TTL > 1，则正常进行路由查找(`m_routingTable.lookup(ip_dst)`)，找到下一跳接口并检查ARP缓存是否有下一跳IP对应的MAC地址。如果ARP条目缺失，则需要排入ARP请求队列等待解析；如果ARP条目存在，则调用`sendIpv4`将数据包沿对应的接口和MAC地址继续转发出去。

Review

本次实验中较为困难的一点是 Debugging 过程。由于代码的高度封装性已经代码量较大，定位错误较为困难；但另一方面，代码中本身提供了大量的可供打印信息的函数等工具，在理解本次实验的内容的基础上可以较为顺畅的利用这些工具辅助排查错误。

该代码可以在本地的 UTM 虚拟机上运行，并通过评测脚本得到 45/45 的分数。