西安交通大学

XI'AN JIAOTONG UNIVERSITY

计算机网络
软件大实验指导书

电子与信息工程学部

计算机科学与技术学院

2023 年 11 月

合抱之木，生于毫末；

九层之台，起于累土；

千里之行，始于足下。

------《老子》

前　言

本指导手册是为配套《计算机网络》课程的软件大实验（大作业）指导用书，该软件大实验基于计算机网络基础理论，进行网络软件编程，培养学生网络技术编程能力。

软件大实验包括两部分内容，分别是 socket 编程实现与路由协议实现。根据实验内容，分为 1-2 人一组进行相关编程，尽量做到每人角色和职责均不相同。

张未展、王烨

二〇二三年十一月

注意事项

1. 包含 2 个网络编程实验。
2. 可以讨论代码思路，严禁抄袭代码（包含往届学长的代码），希望保持高度的学术诚信，并且为自己的工作感到自豪，涉嫌作弊或者伪造的作业将根据课堂指定的相关规则予以处理。
3. 现场软件验收、和《软件实验报告》是反映本实验效果的 2 个主要环节，也是成绩评定的主要依据。

目 录

1 实验一 Socket 网络编程实验

1.1 实验目的

1) 掌握 Sockets 的相关基础知识，学习 Sockets 编程的基本函数和数据类型
2) 掌握 UDP、TCP Client/Server 模式的通信原理。
3) 掌握 socket 编程命令

1.2 实验内容

1) 实现一个简单的客户机/服务器程序,基于 TCP 和 UDP 协议分别实现。
2) 应用场景为一个验证用户登录的程序。

1.3 实验原理

1.3.1 熟悉 TCP，UDP 协议原理

略。

1.3.2 Socket 网络编程原理

下面简单介绍互联网的 Client/Server 模式的工作原理，以 TCP 服务器为例说明，UDP 服务器略有不同。客户端也是如此。

1）服务器

服务器先创建一个套接字（Socket），并将该套接字和特定端口绑定，然后服务器开始在此套接字上监听，直到收到一个客户端的连接请求，然后服务器与客户端建立连接，连接成功后和该客户端进行通信（相互接收和发送数据），进行用户信息验证，并返回验证信息。最后，服务器和客户端断开连接，继续在端口上监听。

2）客户端

客户端创建一个套接字，里面包含了服务器的地址和端口号，客户端的端口号由系统自动分配，不需要指明。和服务器建立连接，如果连接成功则 socket 创建成功。然后客户端发送用户名和密码，等待验证。通信结束后主动断开连接，释放资源。

1.4 实验要求

1）比较 TCP DUP 两种协议的不同，在实验报告中写出自己的理解；

2）可用多种语言实现，建议 C/C++，JAVA 或 Python。

1.5 实验环境和分组

1）每 2 位同学一组，共编写程序（一人客户端，服务）。

2）编程时请自备电脑编程调试。验收时电脑 1 台。

1.6 实验步骤

步骤 1：编写 server 端程序

步骤 2：编写 client 端程序

步骤 3：client 端和 server 端实现互联通信，验证用户登录验证用户登录信息。例如，客户端发送用户名和密码，如若信息正确服务器端返回：送用户名和密码，如若信息正确服务器端返回：送用户名和密码，如若信息正确服务器端返回：“*信息正确*”；否则，服务器端返回：“*用户名或密码错误请再次输入*”。（提示信息不唯一，可自由改变）

1.7 结果分析

1) 如何服务器能实现循环监听？

2) 比较两种协议在代码层面的区别。

1.8 互动讨论主题

1) TCP 协议和 HTTP 协议的区别和联系；

1.9 进阶自设计

1) 在服务器端实现多线程的好处是什么？如何在服务端实现多线程？

2 实验二 RIP 路由协议软件实验

2.1 实验目的

通过软件实现 RIP 协议，详细分析距离矢量路由算法，掌握网络协议的构建过程。

2.1 实验内容

在软件层面上构建网络拓扑，编写路由器等代码，然后基于网络拓扑实现 RIP 路由协议，以及相应分析实验结果，如何实现路由表动态更新等。（本实验的基础代码会提供给大家，供大家参考和借鉴）。

2.2 实验原理

2.2.1 RIP 路由协议

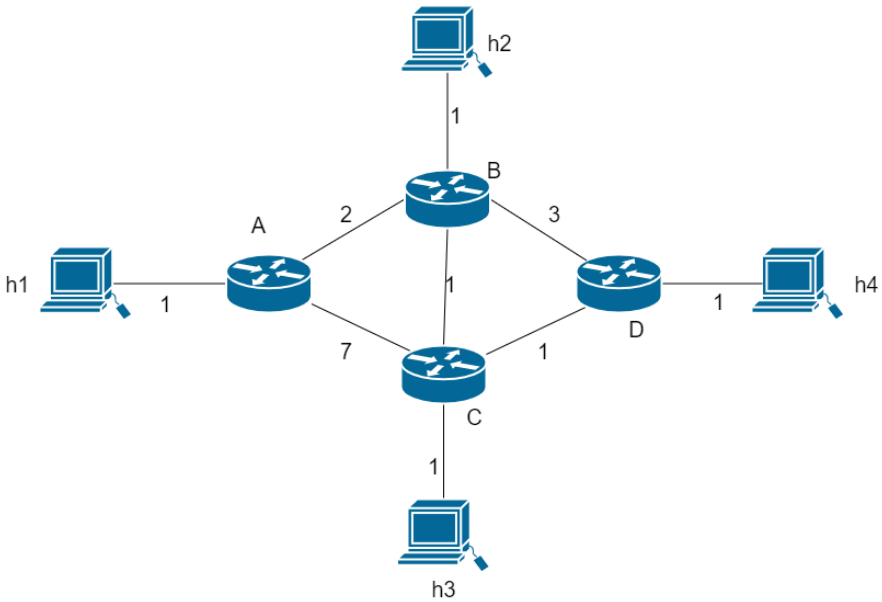
RIP 路由协议的基本内容已在第 9 节详细描述。此处略。

2.3 实验环境与分组

1） 每 2 名同学一组，共同完成。

2.4 实验网络结构

该图是本实验的组网图，权重如图所示。



2.5 实验步骤

最后实验验收分为必选项和可选项（可选项对成绩不会造成太大影响，主要目标是让大家体验下伯克利大学（该项目源自 cs168）的作业形式）

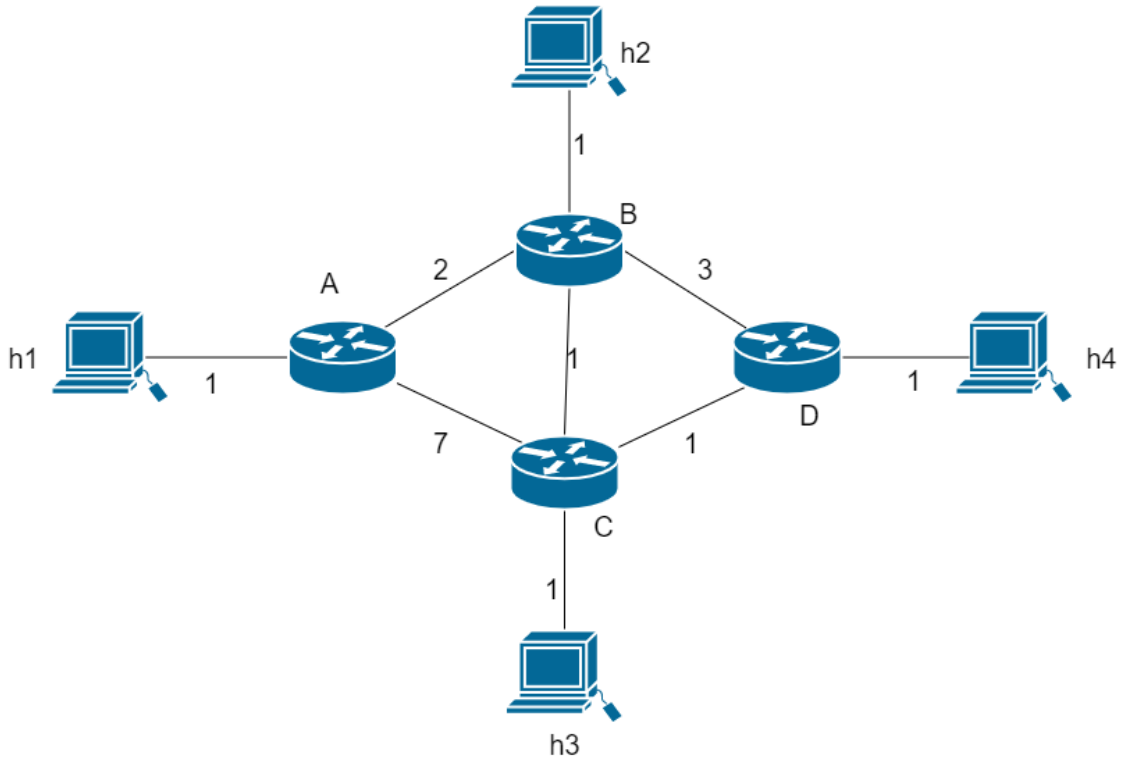
2.5.1 必选项

1) 熟悉所给资料，平台代码，仿真器（可以对编写的部分进行测试）。

```
一些重要文件(需要阅读)如下：
simulator.py Starts up the simulator
dv_router.py Starting point for your distance vector router
dv_unit_tests.py Stage-by-stage unit tests for your distance vector router.
dv_comprehensive_test.py Comprehensive test for your distance vector router.
sim/api.py Parts of the simulator that you'll need to use (such as the Entity class
). See
help(api).
sim/basics.py Basic simulator pieces built with the API. See help(basics).
sim/core.py Inner workings of the simulator.
topos/ Test topologies and topology generators that you can use and modify for your

own testing.
examples/ Examples for Entities, interacting with NetVis, automating your testing,
and
so forth.
cs168/ Additional topologies
```

2) 编写网络拓扑代码，可以参考 topos 文件夹中的其他拓扑结构，然后在该文件夹下编写自己的拓扑文件（注意添加权重，也就是 latency，可以参考别的 topo 文件来写）。要实现的拓扑结构如下图所示：



3) 编写路由器代码（实际上为补全 dv_router.py），实现 **project1_writeup.pdf** 文件中的 **state1~5**（在每一步 state 处均有详细的介绍与测试方法，这里不再赘述），此时应该已经实现了 RIP 路由器的基本功能，使用编写的网络拓扑代码进行下面的测试：

```
1. python simulator.py --start --default-switch-type=dv_router topos.myTopo
```

此时使用 topos.myTopo 是因为 topos 文件夹下有自己编写的拓扑文件 myTopo.py（也就是上一步我们编写的拓扑文件），使用时替换为自己定义的名字即可。

```
1. h1.ping(h4)
```

最后网络中没有出现泛洪现象，同时包走的是最短路径（A->B->C->D），说明我们的 RIP 算法设计成功。

ATTENTION:
对于 state1~5，每一个 state 均有相应的测试，只有测试结果正确才能说明我们的路由器代码编写正确（最终验收时可以展示每一个 state 的测试也可以不测试，只要最终的测试——即上面的测试通过即可）

2.5.2 可选项

完成 state6~10，通过每一个 state 的测试即可（具体要求请参照 project1_writeup.pdf 中相关要求）

2.6 实验工具说明（Quick Start）

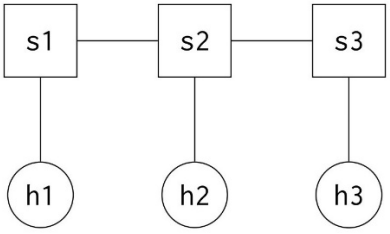
4) 首先确保你的运行环境为 python3，请打开命令行输入 python -V 来确保你的版本为 python3 版本。（对操作系统并无要求，可以使用 windows 和 linux）

5) 解压文件，进入 simulator 文件夹，在该文件夹下打开命令行。

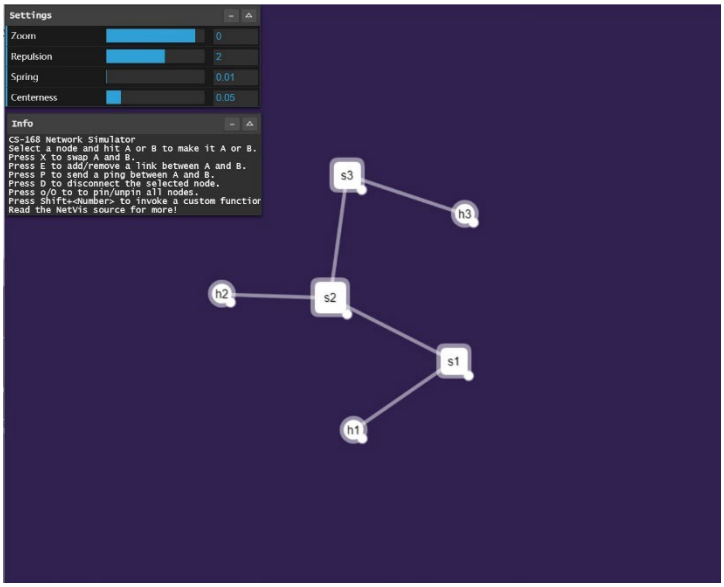
6) 我们的目标是编写一个具有 RIP 的路由设备，那么我们作为演示我们提供一个 hub 的实现，hub 是一种网络设备，它可以将它收到的任何数据包泛洪到它所有的端口(除了数据包来的端口)。该集线器我们已经实现，可以直接使用。

下面我们尝试使用一个有着三个 hosts 的线性拓扑图(如下图所示)，我们使用之前提供的默认的 hub 来充当网络中的路由器，在命令行中输入如下指令：

```
1. python simulator.py --start --default-switch-type=examples.hub topos.linear --n=3
```



之后就可以使用可视化工具，请使用浏览器访问 http://127.0.0.1:4444，此时可以观察到整个路由结构并且可以拖动各个部件，如下图所示：



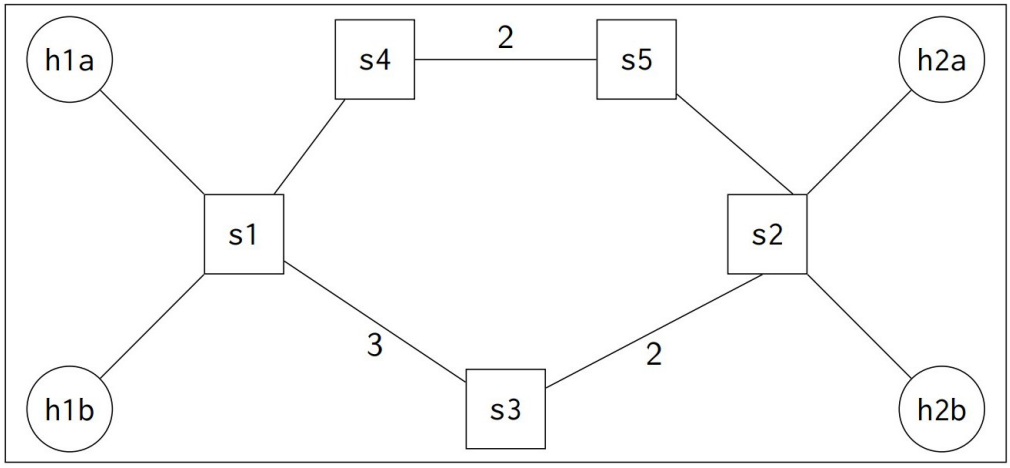
此时在原来的命令行中输入如下参数：

```
1.  >>> h1.ping(h3)
```

此时应该可以看到 ping 和 pong 数据包在网络中出现，但是可以注意到 h2 端此时也接收到了来自 h1 的 ping 包和 h2 的 pong 包(WARNING 信息如下图所示)，这种行为是可以预料到的，因为 hub 只是接收数据包后向除过接收数据包的端口之外的所有端口泛洪数据包。

```
1.  WARNING:user:h2:NOT FOR ME: <Ping h1->h3 ttl:17> s1,s2,h2
2.  DEBUG:user:h3:rx: <Ping h1->h3 ttl:16> s1,s2,s3,h3
3.  WARNING:user:h2:NOT FOR ME: <Pong <Ping h1->h3 ttl:16>> s3,s2,h2
4.  DEBUG:user:h1:rx: <Pong <Ping h1->h3 ttl:16>> s3,s2,s1,h1
```

如果网络中出现环路，那么可能使用 hub 这种方式连接网络会出现比较大的问题，此时我们使用另一个网络，该网络为环形网络，拓扑结构如下图所示：



此时退出原命令（输入 exit()或者直接 ctrl+z），再次在命令行中输入：

```
1. python simulator.py --start --default-switch-type=examples.hub topos.candy
```

此时从主机 h1a 向主机 h2b 发送 ping 命令。此时可以看到已经出现泛洪问题，我们观察到有过多的 冗余数据包在网络中流动，这也就是我们为什么要编写新的路由器，编写新的路由协议的原因。

ATTENTION：

在这里命令行中输入什么样的规则请参照文件夹中的 simulator_guide.pdf 文件，里面有如何进行调试的详细讲解（不需要过多了解，因为后面的测试都会给出具体的命令行参数）