中山大學

计算机网络实验报告

智能路由器开发

学院名称：计算机学院

专业（班级）：20 级计科人工智能与大数据

学生姓名：崔璨明

学号：20337025

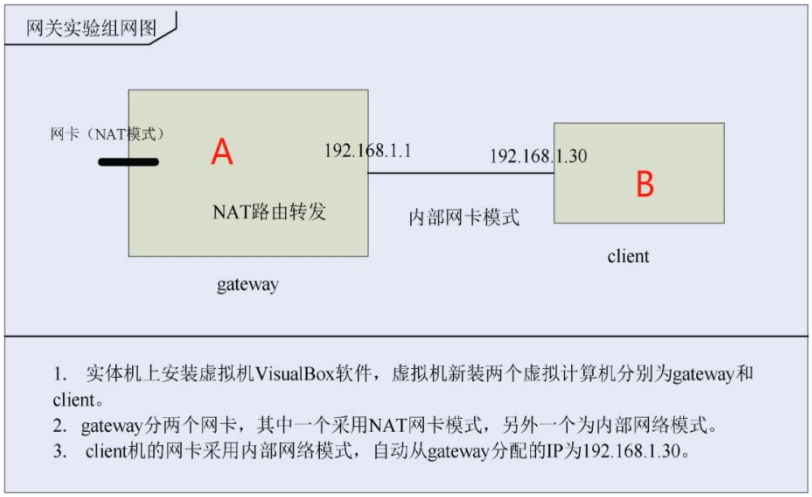
【实验题目】智能路由器开发

【实验目的】

1. 掌握开发智能路由器的方法。
2. 掌握 openwrt 智能路由器的配置。
3. 实现路由器的基本功能。
4. 巩固本学期计算机网络的所学知识。

【实验内容】

1. 组建路由器实验环境



如图所示，将虚拟机 A 当作智能路由器，安装 OpenWrt 软件，并创建两个网卡，NAT 网卡用于连接互联网，内部网络用于连接家庭网个人计算机。虚拟机 B 当作家庭 PC 使用，自动从 OpenWrt 网关处分配 IP 地址。这样我们就可以模拟常见的路由器场景，例如上网、防火墙和 DNS 代理等功能，任何家庭网的数据流量均通过路由器来转发到外部网络。

2. Helloworld

H 公司想了解已售出路由器的使用情况，如路由器的启动次数。期望在路由器每次启动时将访问指定服务器（实体机）通过访问记录来保存路由器启动次数，可以通过修改路由器的配置文件实现。

为了防止某些小区在断电恢复后自动启动所有路由器同时立即访问服务器，对服务器产生瞬间流量冲击，因此需要对路由器启动后提供一个随机延迟时间，然后再访问服务器，以减少服务器压力。

实现路由器启动时向服务器发送消息（你的学号），并在服务器端保存该访问记录。

3. 路由器基本功能（每种基本功能中选择 2 个子功能实现即可）

网络互连：路由器支持各种局域网和广域网接口，主要用于互连局域网和广域网，实现不同网络互相通信；

数据处理：提供包括分组过滤、分组转发、优先级、复用、加密、压缩和防火墙等功能；

网络管理：路由器提供包括路由器配置管理、性能管理、容错管理和流量控制等功能。

【实验环境】

- Openwrt
- Virtualbox

【实验记录】(如有实验拓扑请自行画出)

1. 组建路由器实验环境

首先下载 Virtualbox 并搭建虚拟机，为了避免不必要的差错，我选择的 ubuntu 版本为 16.04，和教程《智能路由器开发指南》中的一致。

我创建了三台虚拟机，分别命名：openwrt、Client、Gateway。其中，虚拟机 openwrt 安装系统 ubuntu 16.04，将网卡设置为网络地址转换(NAT)，用于下载和编译 OpenWrt 软件，以生成 OpenWrt.vdi 供给 Gateway 使用。

虚拟机 Gateway 安装系统为虚拟机 openwrt 编译生成的 openwrt.vdi，用于运行 openwrt，充当路由器。该虚拟机创建两个网卡，NAT 网卡用于连接互联网，内部网络用于连接家庭网个人计算机。

虚拟机 Client 充当家庭网络中的个人计算机，自动从 OpenWrt 网关处分配 IP 地址，可以进行各种功能的测试。

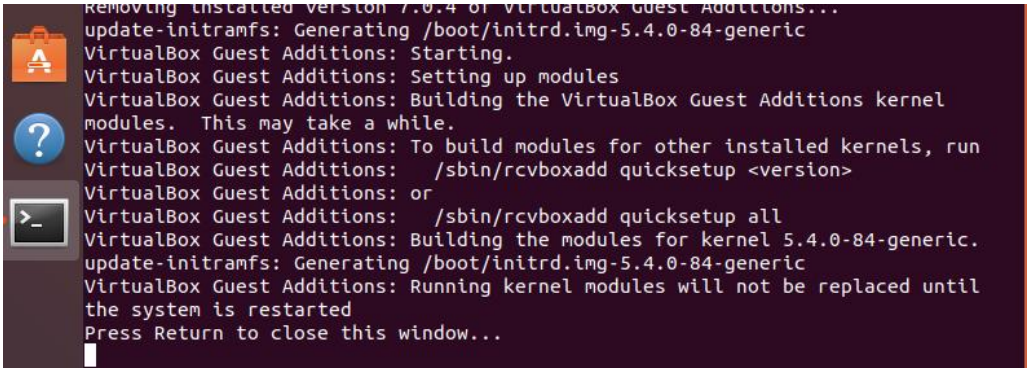
1.1 编译生成 openwrt

虚拟机 openwrt 配置如下：

启动虚拟机并安装 ubuntu16.04，由于后续实验需要从中获取 openwrt.vdi 文件，因此需要创建共享文件夹，首先安装增强功能以方便后续实验：

首先安装所需要的包，命令为“sudo apt install gcc make perl -y”接着点击“设备”->“安装增强功能”：

安装成功显示信息：

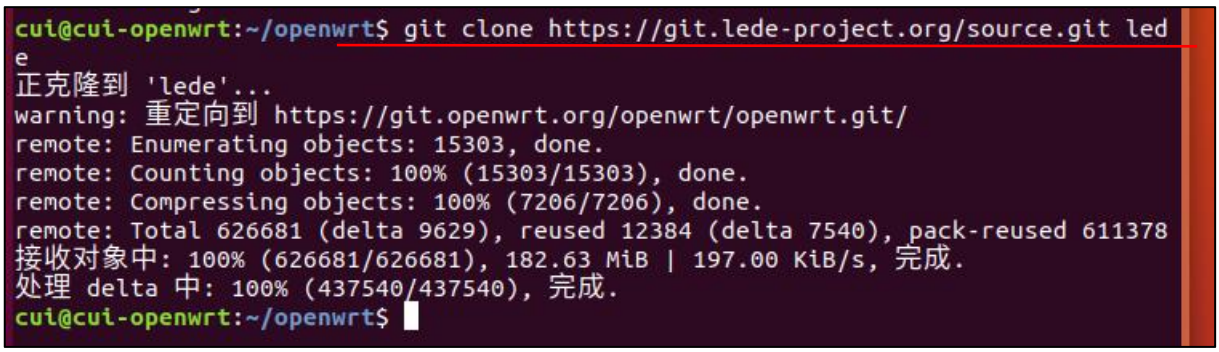


后续实验中的下载速度较慢，导致很多次下载都因为网速较慢而出现各种各样的错误，为了避免这些情况，首先可以进行下载地址换源：打开“软件和更新”->在“下载至”一栏中搜索当前最佳服务器，并将服务器改为最佳服务器：

接下来进行 ubuntu 的系统配置，在终端输入“sudo apt-get update”进行系统更新，更新完成后，按照教程安装必须的库和程序，ubuntu16.04 的安装命令为：

“sudo apt-get install gcc g++ binutils patch bzip2 flex bison make autoconf gettext texinfo unzip sharutils libncurses5-dev ncurses-term zliblg-dev gawk asciidoc libz-dev git core uuid-dev libacl1-dev liblzo2-dev pkg-config libc6-dev curl libxml-parser-perl ocaml-nox”

然后下载 openwrt 的源码，新建一个文件夹 openwrt，并在该文件夹中打开终端，输入命令“git clone https://git.lede-project.org/source.git lede”：



进入到获取的源码中的 lede 文件夹，进入 scripts 脚本目录，下载编译所需的 feeds 工具：

所用到的命令：

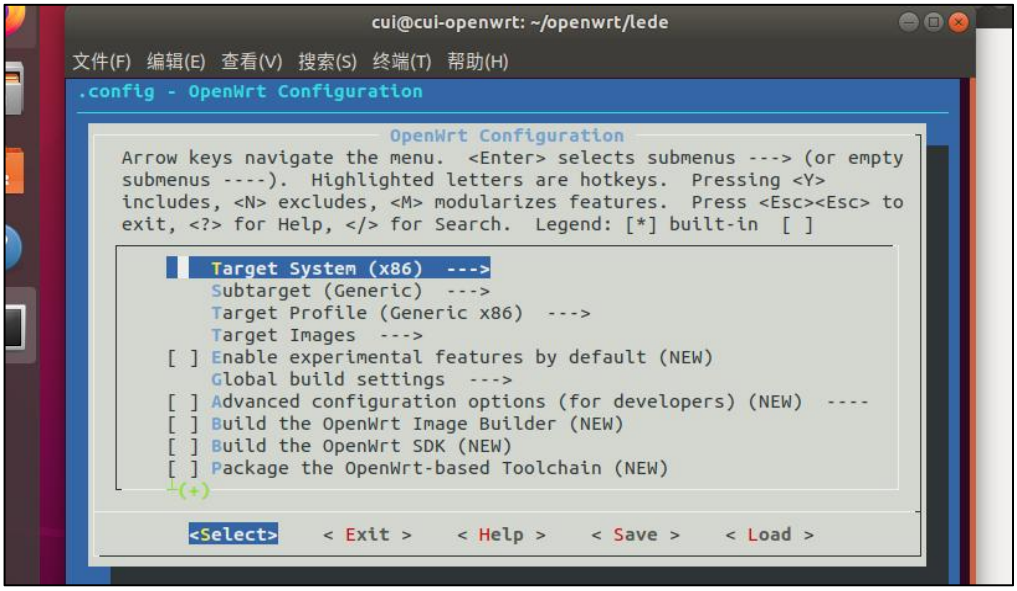
./feeds update -a

./feeds install -a

结果如下：

```
Installing package 'freetdm' from telephony
Installing package 'kamailio' from telephony
Installing package 'libosip2' from telephony
Installing package 'miax' from telephony
Installing package 'pcapsipdump' from telephony
Installing package 'rtpengine' from telephony
Installing package 'rtpproxy' from telephony
Installing package 'sipgrep' from telephony
Installing package 'sipp' from telephony
Installing package 'siproxd' from telephony
Installing package 'sngrep' from telephony
Installing package 'yate' from telephony
cui@cui-openwrt:~/openwrt/lede/scripts$
```

然后切换到 lede 文件夹下，在终端输入命令“make menuconfig”进入到配置页面，注意这里要确保终端的窗口大小足够大，不然打不开该配置窗口：



如图所示，在配置窗口选择“Target System （x86）”，若选择了其他系统则也可以编译出来镜像文件，但不能在 x86 的虚拟机上使用。

配置完成后保存并退出，在终端输入编译指令“make V=99”，编译时间较长，在编译过程中我还遇到了磁盘空间不足、网速过慢自动断开、交换区空间不足系统自动杀死进程等问题。解决方法：将磁盘空间扩大到 40G 或以上，通过“mkswap”等命令将交换区空间扩大到 3G 以上，若网速过慢，则可以将当前正在下载的文件的网址记录，用实体机进行下载后，通过共享文件夹传进去，然后继续编译即可。

编译结果如下：

```
Signing package index...
make[2]: Leaving directory '/home/cui/openwrt/lede'
export MAKEFLAGS= ;make -w -r json_overview_image_info
make[2]: Entering directory '/home/cui/openwrt/lede'
WORK_DIR=/home/cui/openwrt/lede/build_dir/target-i386_pentium4_musl/json_info_fi
les /home/cui/openwrt/lede/scripts/json_overview_image_info.py /home/cui/openwrt
/lede/bin/targets/x86/generic/profiles.json
make[2]: Leaving directory '/home/cui/openwrt/lede'
export MAKEFLAGS= ;make -w -r checksum
make[2]: Entering directory '/home/cui/openwrt/lede'
make[2]: Leaving directory '/home/cui/openwrt/lede'
make[1]: Leaving directory '/home/cui/openwrt/lede'
cui@cui-VirtualBox:~/openwrt/lede$
cui@cui-VirtualBox:~/openwrt/lede$
```

接下来需要将其转换为可供虚拟机使用的 vdi 镜像，进入到 openwrt/lede/bin/x86 文件夹中，可以看到编译生成的“openwrt-x86-generic-generic-ext4-combined.img.gz”文件，将其解压缩得到 openwrt-x86-generic-generic-ext4-combined.img。

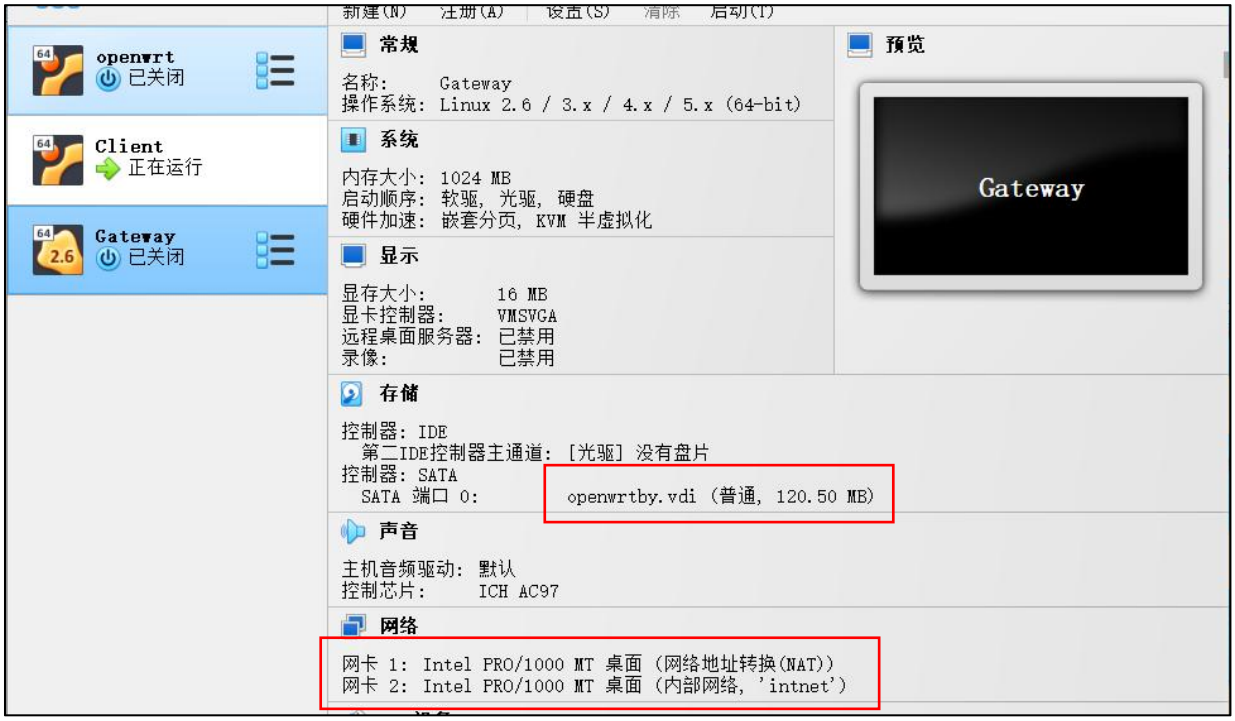
但还需要将其转换为 VirtualBox 支持的 vdi 文件，利用共享文件夹将其复制到实体机中，然后打开 powershell，在 vitualbox 文件夹下输入以下命令进行转换：

```
PS F:\VirtualBox> .\VBoxManage.exe convertfromraw -format VDI F:\VBPC\shared\openwrt-x86-generic-generic-ext4-combined.i
mg F:\VBPC\router\openwrtby.vdi
Converting from raw image file="F:\VBPC\shared\openwrt-x86-generic-generic-ext4-combined.img" to file="F:\VBPC\router\op
enwrtby.vdi"...
Creating dynamic image with size 126353408 bytes (121MB)...
PS F:\VirtualBox>
```

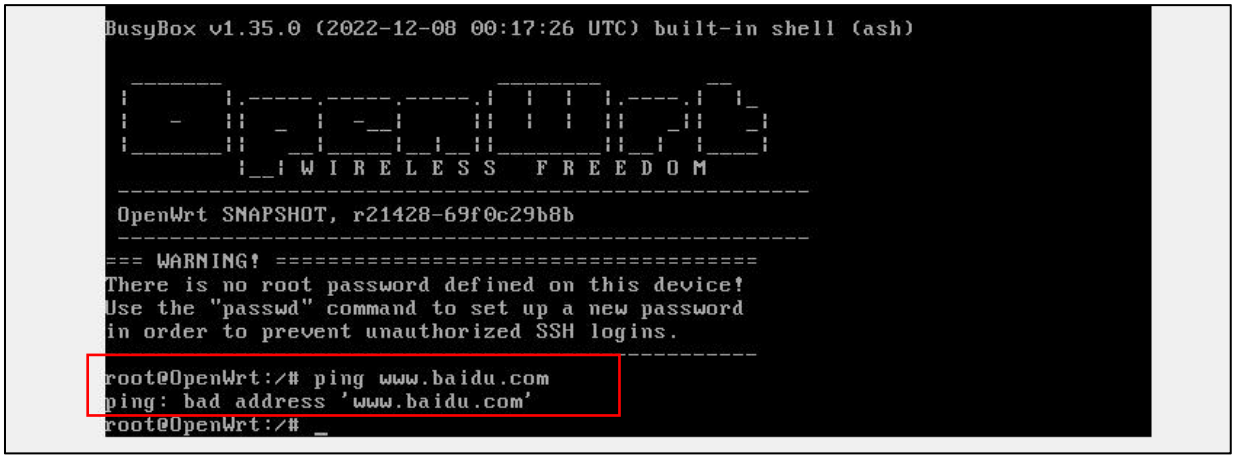
转换成功后便可以用其建立智能路由器 Gateway。

1.2 路由器 Gateway 配置

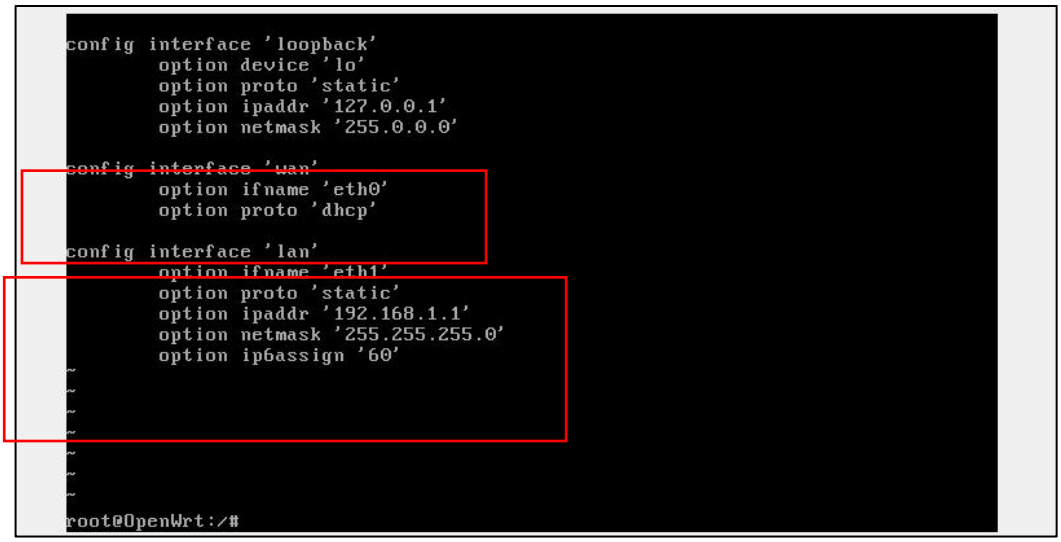
Gateway 虚拟机使用上一步得到的 openwrtby.vdi，设置两个网卡，网卡 1 设置为网络地址转换 NAT，网卡 2 设置为内部网络 intnet：



打开该虚拟机，首先进行测试，执行 ping www.baidu.com 命令，可见此时是 ping 不通的，即上不了网：

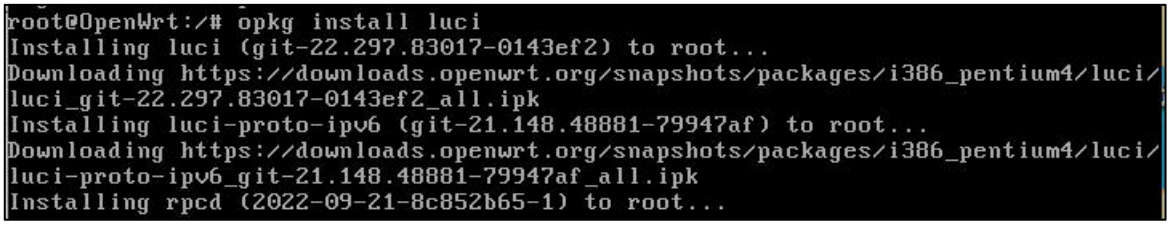


进入到 openwrt 的配置文件中进行配置，命令为“vim /etc/config/network”，将其修改如下，主要修改无线网‘wan’和局域网‘lan’中的内容：



配置完后要重启网络服务才能生效，重启命令“/etc/init.d/network restart”，此时再次进行测试，ping 百度成功，说明此时可以上网，智能路由器配置成功：

为了模拟路由器的使用场景，需要使用 opkg 命令安装 OpenWrt 的 Web 管理界面，在终端依次输入以下命令：“opkg update”->“opkg install luci” ->“/etc/init.d/uhttpd enable” -> “/etc/init.d/uhttpd start” -> “/etc/init.d/firewall stop”：



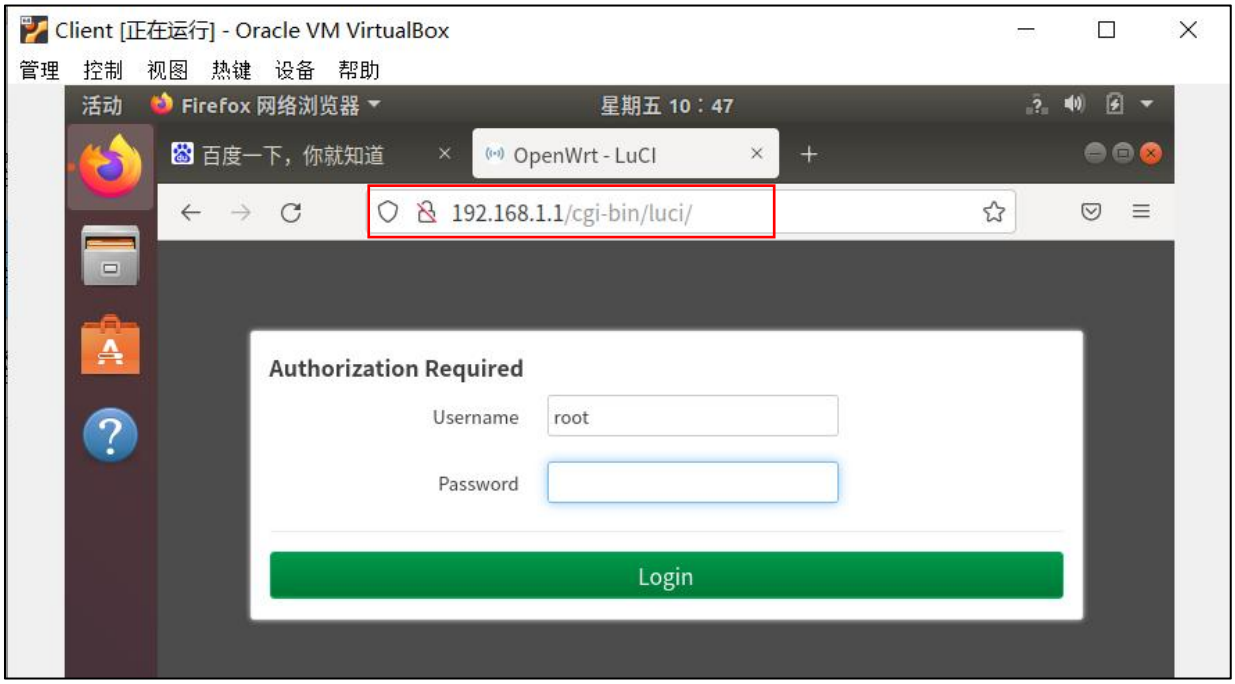
执行完后重启虚拟机即可，接下来进行客户端的虚拟机的配置和验证智能路由器。

1.3 客户端 Client 配置

客户端虚拟机安装的也是 ubuntu16.04 系统，网卡设置为内部网络 intnet：

该客户端当作家庭 PC 使用，自动从 OpenWrt 网关处分配 IP 地址，打开即可上网：

输入 192.168.1.1 可以进入 OpenWrt 的 Web 管理界面，在这里可以直接对路由器进行配置：

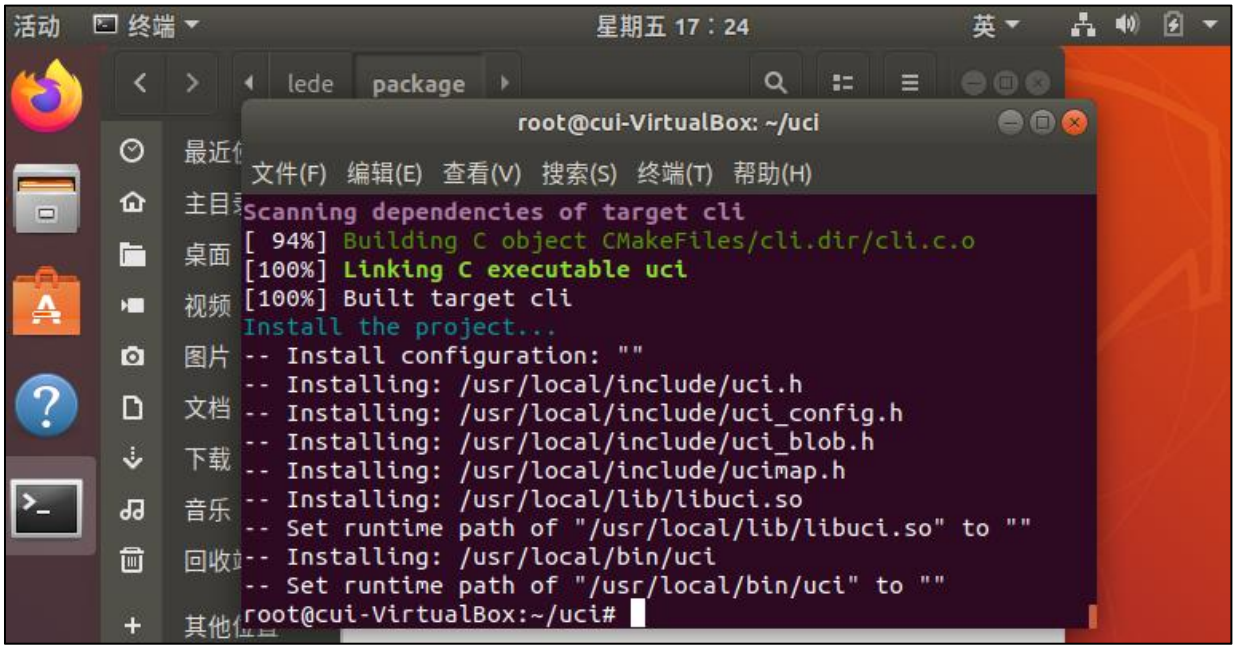


2. 进行 Helloworld 实验

根据教程编写 helloworld 的源代码和 makefile 文件，讲发送的信息设置为我的学号 20337025，发送的网址设为 www.baidu.com，开机后随机发送的时间设置为 10 秒之内，具体的代码及文件见附件的 hello 文件夹。

编写完 hello 文件后，将 hello 文件拖入 package 文件夹中，准备编译，在编译之前还需要完成以下工作

1. 安装 UCI 在 root 模式下依次执行以下命令：cd /root->git clone http://git.nbd.name/luci2/libubox.git libubox -> cd libubox ->cmake -DBUILD_LUA=off ->make install ->cd /root ->git clone https://git.openwrt.org/project/uci.git uci ->cd uci ->cmake -DBUILD_LUA=off ->make install



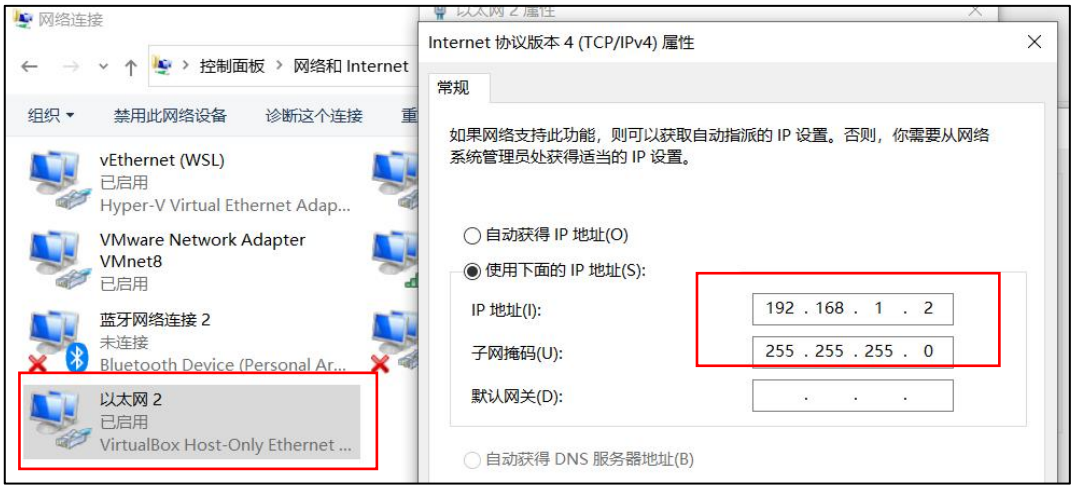
2. 添加依赖库路径，命令：vim /etc/ld.so.conf
3. 在最后一行加上/usr/local/lib，
4. 在 root 模式下输入 ldconfig 命令，使依赖路径生效
5. 在 lede 文件夹下执行命令“make menuconfig”中进行配置，将 hello 添加到 Network 中,并设置为 modularizes features

然后进行编译,编译命令和结果如下：

```
cui@cui-VirtualBox:~/openwrt/lede$ make package/hello/compile v=s
make[2]: Entering directory '/home/cui/openwrt/lede/scripts/config'
make[2]: 'conf' is up to date.
make[2]: Leaving directory '/home/cui/openwrt/lede/scripts/config'
 make[1] package/hello/compile
 make[2] -C package/libs/toolchain compile
 make[2] -C package/libs/libjson-c compile
 make[2] -C package/utils/lua compile
 make[2] -C package/libs/libubox compile
 make[2] -C package/system/uci compile
 make[2] -C package/hello compile
cui@cui-VirtualBox:~/openwrt/lede$
```

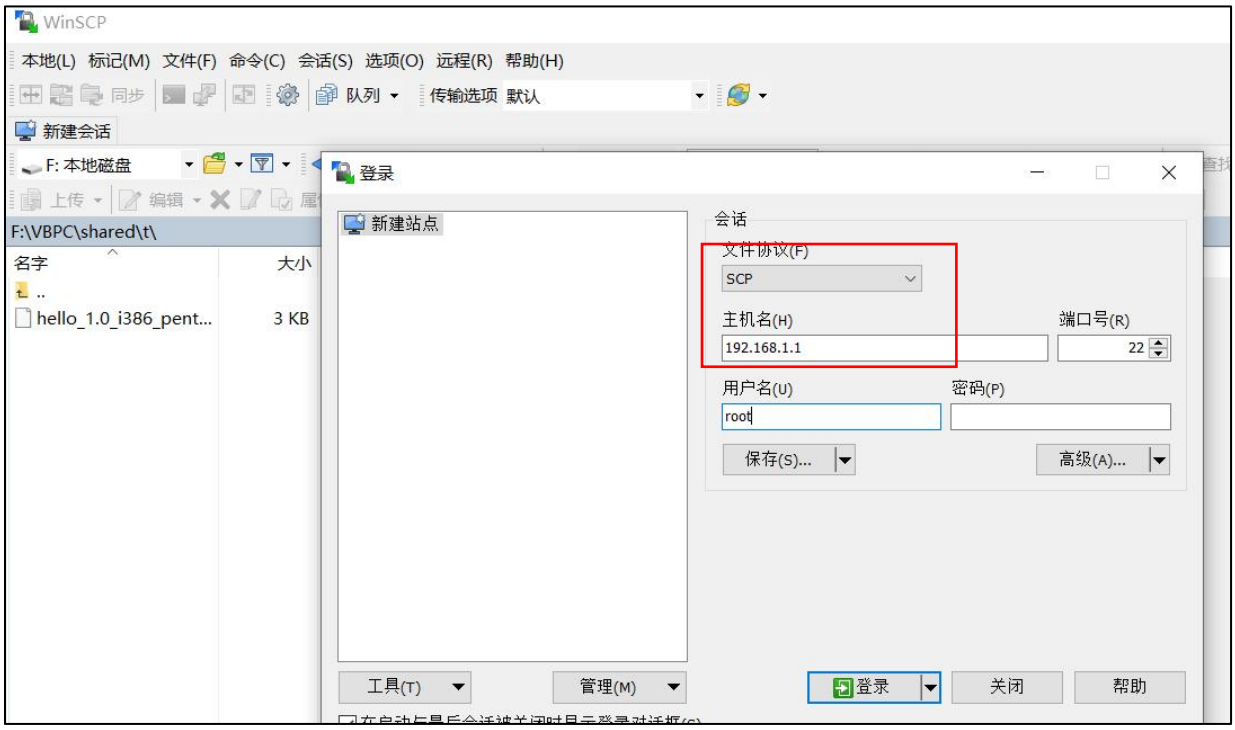
编译完后生成 hello_1.0_i386_pentium4.ipk 文件，利用共享文件夹将其拷贝到实体机上面来。

接着将虚拟机 Gateway 的第二张网卡换成 Host-Only，然后打开实体机的网络连接，找到 Host-Only 对应的网络，将其设置如下，更改其 IP 地址：

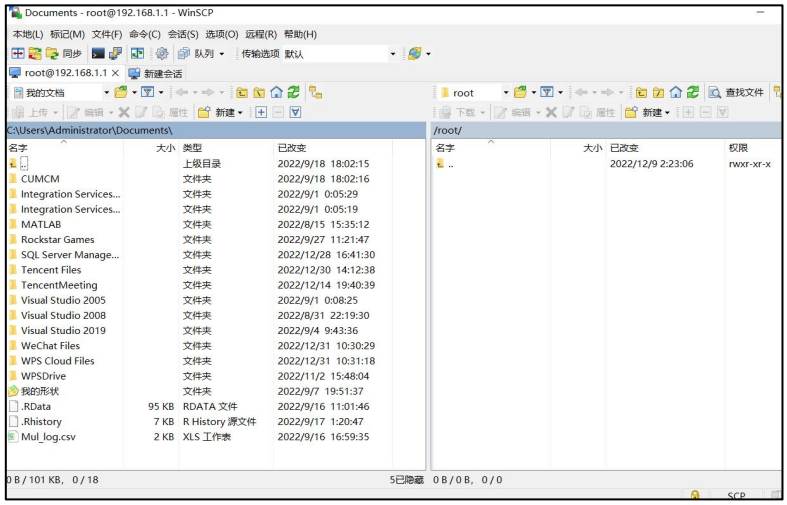


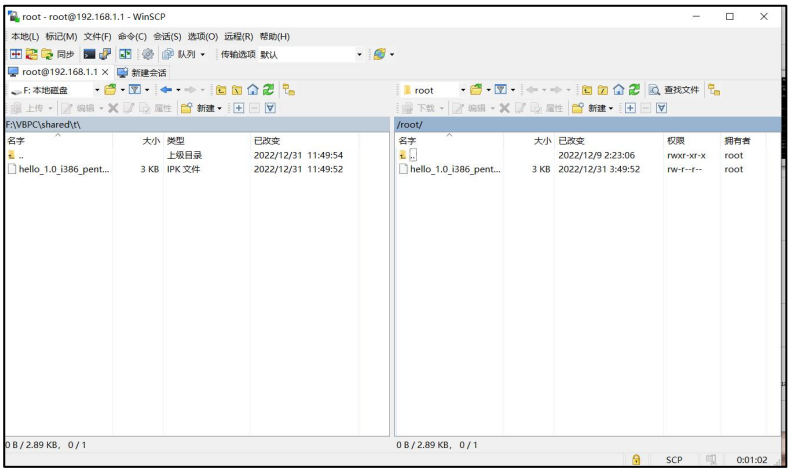
更改完后可进行测试，确保实体机能 ping 通 Gateway 虚拟机。

下载 WinSCP 软件，新建站点，主机名设置如下：

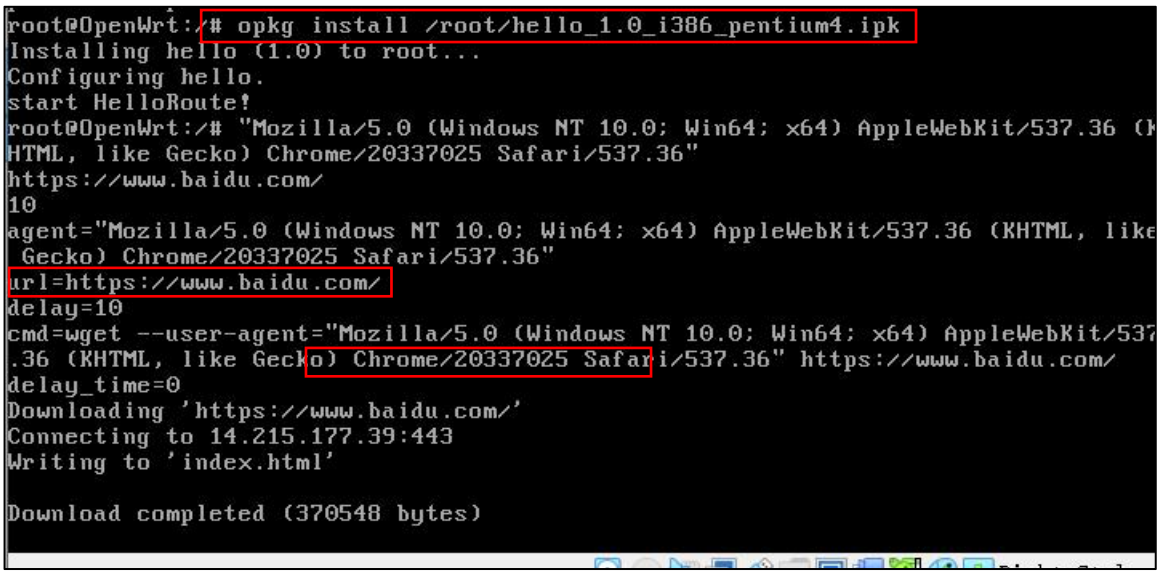


将 ipk 文件拖到 Gateway 虚拟机中：



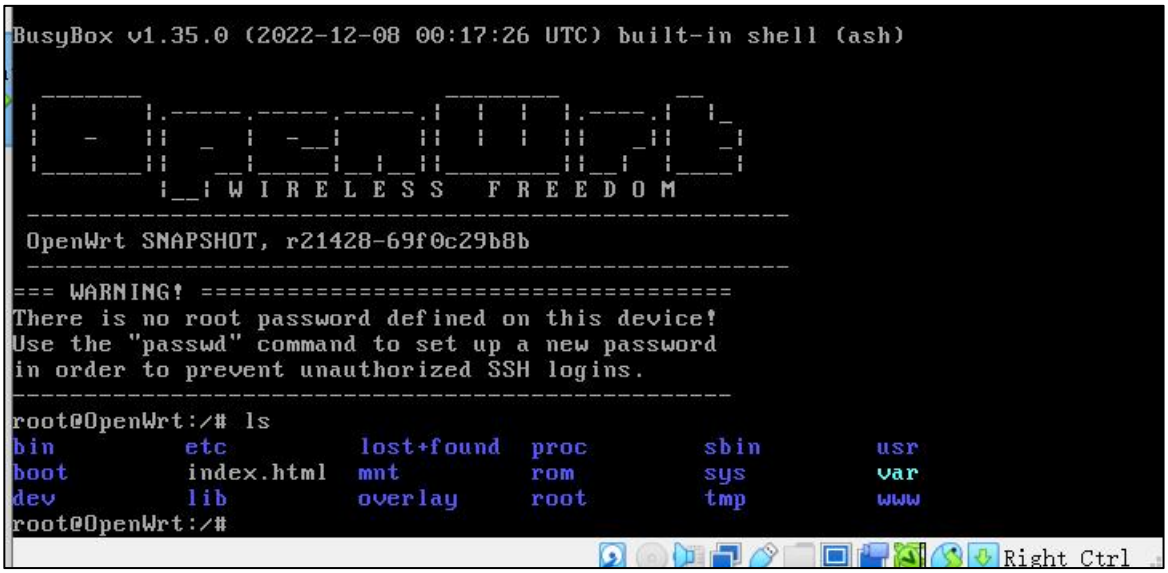


打开 Gateway，输入 opkg install /root/hello_1.0_i386_pentium4.ipk 安装文件，可以在输出信息中看到，设置的网址为 www.baidu.com，发送的内容为我的学号 20337025：



安装完成后，在目录下使用 ls 命令可以看到有个 index.html 文件，为来验证它能开机自启动，先将其删掉，然后输入 reboot 命令重启：

可以看到目录下又出现了该文件，说明开机自动下载了文件，实现成功：



如上述步骤所示，我完成了 helloworld 实验，路由器每次启动时将访问指定服务器 www.baidu.com，在该网站的后台可以查看到路由器的启动次数。除此之外，路由器启动后会有一个随机延迟时间，在 hello.c 使用 rand()函数实现，然后再访问服务器。路由器启动时向服务器发送消息（学号 20337025），在服务器端可查看到该记录。

3. 路由器基本功能

3.1 网络互连

在网络互联方面，路由器支持各种局域网和广域网接口，实现了局域网和广域网的互

连，可以使不同网络互相通信。如下所示，路由器可以连通其他网站，说明连通了广域网：

```
root@OpenWrt:/# ping www.bilibili.com
PING www.bilibili.com (14.17.92.72): 56 data bytes
64 bytes from 14.17.92.72: seq=0 ttl=52 time=14.564 ms
64 bytes from 14.17.92.72: seq=1 ttl=52 time=14.215 ms
64 bytes from 14.17.92.72: seq=2 ttl=52 time=17.019 ms
64 bytes from 14.17.92.72: seq=3 ttl=52 time=14.751 ms
64 bytes from 14.17.92.72: seq=4 ttl=52 time=14.495 ms
64 bytes from 14.17.92.72: seq=5 ttl=52 time=19.255 ms
64 bytes from 14.17.92.72: seq=6 ttl=52 time=14.807 ms
```

而客户机 Client 可以 ping 通广域网（www.baidu.com），也可以 pin 通路由器(192.168.1.1)，这说明路由器连通了局域网，以互连局域网和广域网，使得不同网络互相通信：

```
cuiclient@cuiclient-VirtualBox:~$ ping www.baidu.com
PING www.baidu.com (14.215.177.39) 56(84) bytes of data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=1 ttl=53 ti
me=21.9 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=2 ttl=53 ti
me=19.0 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=3 ttl=53 ti
me=20.8 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=4 ttl=53 ti
me=19.5 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=5 ttl=53 ti
me=21.4 ms
```

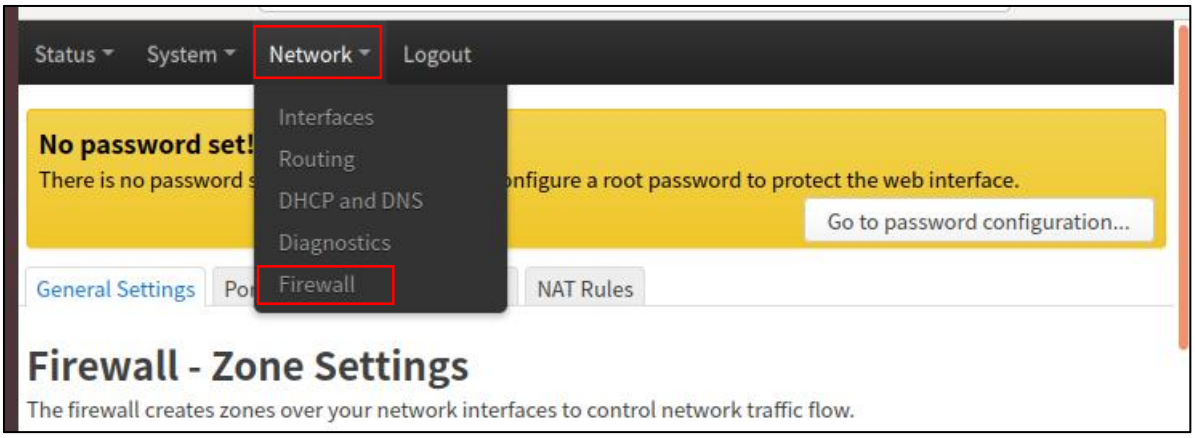
```
cuiclient@cuiclient-VirtualBox:~$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.804 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.982 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.720 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=0.717 ms
```

3.2 数据处理

在数据处理方面，实现了两个功能：防火墙和分组转发。

防火墙的实现可以直接用客户机 Client 登录路由器的 Web 管理界面，通过修改上面的配置来进行设置，设置步骤如下：

1. 打开 Network 中的 firewall 选项：



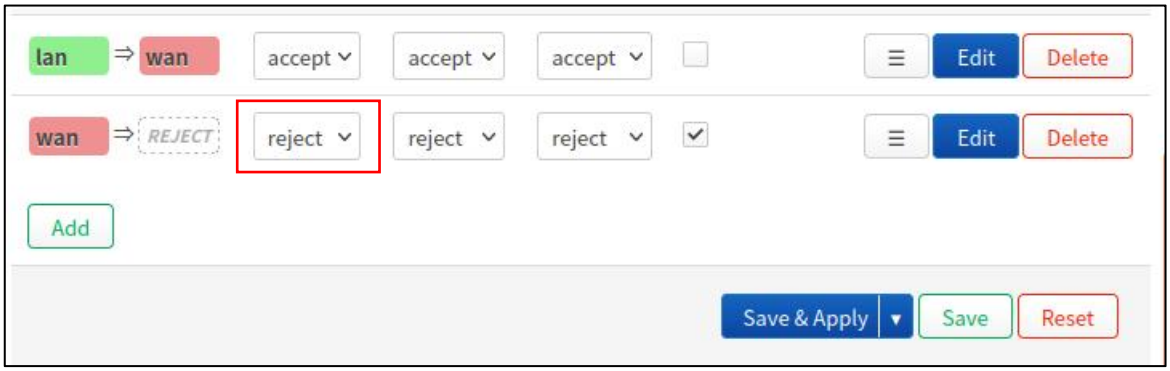
General Settings 是总的设置，可以在一个范围内设置防火墙。如接下来演示在局域网中设置防火墙，将局域网内的主机和广域网相隔开：

2. 可以看到此时的未设置防火墙，wan 端口是 accept 的状态：

此时客户机可以连通广域网：

```
cuiclient@cuiclient-VirtualBox:~$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.884 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.501 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.674 ms
64 bytes from 10.0.2.15: icmp_seq=4 ttl=64 time=0.728 ms
64 bytes from 10.0.2.15: icmp_seq=5 ttl=64 time=0.890 ms
64 bytes from 10.0.2.15: icmp_seq=6 ttl=64 time=0.679 ms
^Z
[3]+  已停止               ping 10.0.2.15
cuiclient@cuiclient-VirtualBox:~$ ping www.baidu.com
PING www.baidu.com (14.215.177.38) 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=53 ti
me=32.2 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=53 ti
me=17.2 ms
```

3. 将其设为 rejected，即在广域网和客户机所处的局域网之间设置防火墙：



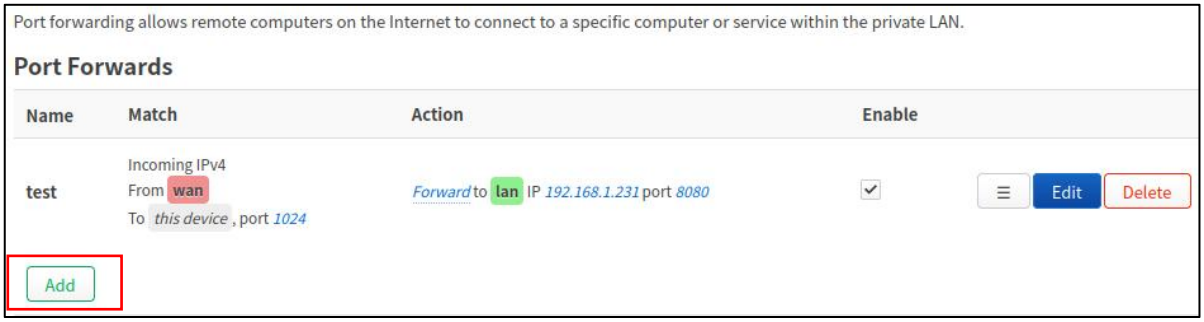
4. 设置防火墙后，客户机不能和广域网连通，只能在局域网之间进行通信：

```
cuiclient@cuiclient-VirtualBox:~$ ping www.baidu.com
ping: www.baidu.com: 未知的名称或服务
cuiclient@cuiclient-VirtualBox:~$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=1.02 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=1.01 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.818 ms
64 bytes from 10.0.2.15: icmp_seq=4 ttl=64 time=0.813 ms
```

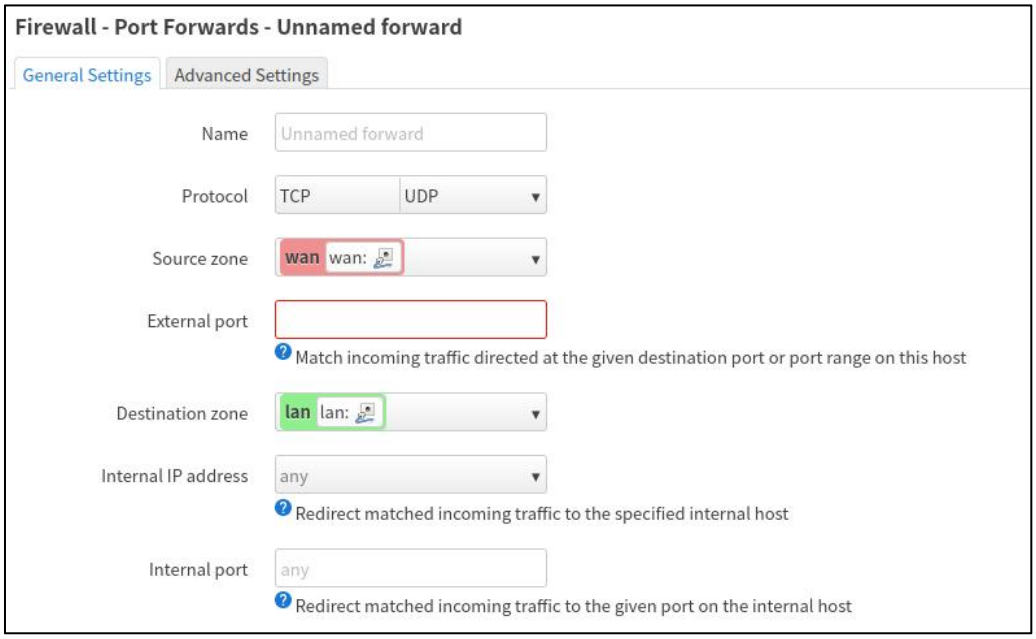
设置分组转发的步骤如下：

1. 在防火墙设置的页面下点击 Port Foward 选项，则可设置分组转发，接下来我以设置来自广域网的到客户机 Client 的所有分组都经过端口 8080 到达为例：

2. 点击 add 按钮添加配置：



3. 配置界面如下所示，name 是设置的规则的名称，Protocol 为要转发的分组的协议，Source zone 是接收的包的源地址，External port 是外部端口，这里选择广域网，因为目的是将广域网的包通过指定端口转发到 Client，匹配定向到主机上给定的目的端口或端口范围的传入流量，Destination 是目的区域，这里选择局域网，最后在 Internal IP address 中选择客户机 Client 和在 Internal port 中指定端口即可：

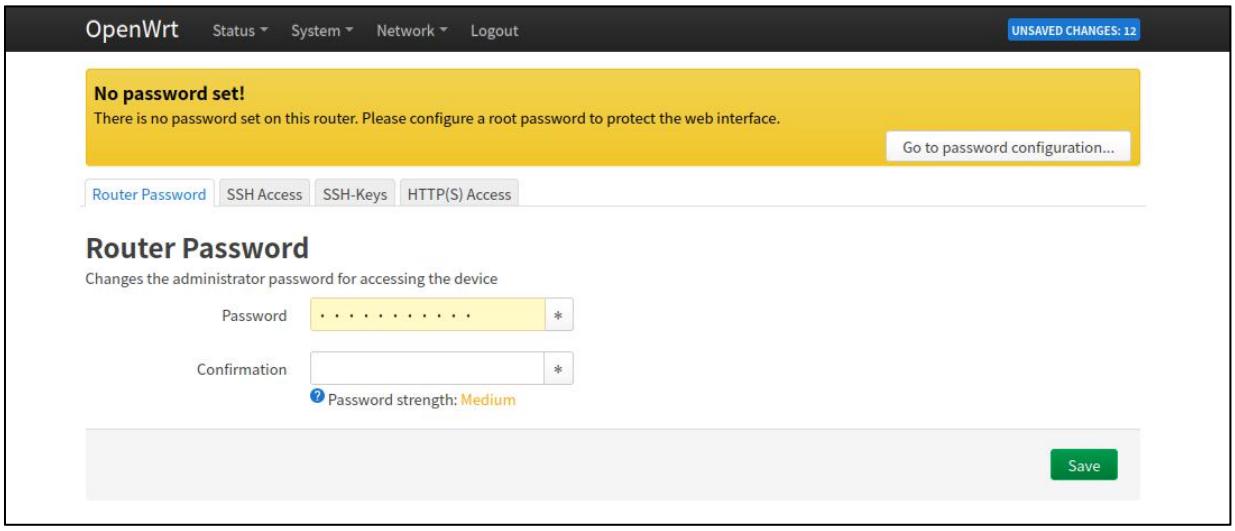


4. 设置成功如下，此时所有来自广域网的 IPv4 的分组都将通过端口 8080 转发到客户机 Client，实现了分组转发的功能：

3.3 网络管理

在网络管理方面实现的是路由器配置管理。

在安装了 luci 插件后，客户机便可以通过 Web 界面来进行路由器配置管理，只需要输入相应的管理员的用户名和密码即可：



【总结&实验心得】

本次期中实验智能路由器是一次全新的体验，和以往我们在计算机网络实验课上进行的实验不同。以往的实验学习只需要跟着实验指导书一步步完成即可，且指导书上的教程和命令都十分详细，所以我们实现起来的时候就比较容易。但这次期中实验却不一样，首先是这次期中实验的指导书是一整本，需要我们去阅读相关的内容，结合自己开发过程中的实际环境，这毫无疑问是一次极大的挑战。

除此之外，指导书上的内容并不是非常详细，许多细节需要自己去探索和研究。在开发智能路由器的期间我也遇到了许多问题，在搜索资料、思考问题的过程中我也积累了许多宝贵的经验。也因此我将此次报告许多配置和安装的细节都记录的十分详细，这也让我收获颇丰。

值得一提的是，使用 VirtualBox 虚拟机来搭建实验环境的体验非常良好，使用起来也非常方便，它不像 VMWare 那样存在着许多问题。除此之外，VirtualBox 安装增强功能的步骤也比 VM 要方便许多。因为我是通过 openwrt 的源码来直接编译生成 vdi 的，因此也花费了大量的时间在编译上面，期间也遇到了许多问题。首先是网速的问题，在编译 openwrt 时，需要下载许多依赖包，许多时候会出现带 pull、fetch 等字眼的报错信息，这些大多数是因为网速原因，因此我建议选取合适的服务器或这记住出错的网址和正在下载的文件，在实体机上下载后再通过共享文件夹传入到虚拟机的对应位置中。openwrt 编译过程中生成的文件较多，因此需要留有 30G 以上的磁盘空间，不然很容易因为磁盘空间不足而导致编译无法进行。

在进行 openwrt 的配置时，网上有许多可以参考的资料，因此这部分较为顺利，只需细致地阅读相关的教程即可。

通过这次实验，我巩固了本学期学到的计算网络的知识，且锻炼了自己的能力，能将所学知识应用到实践中。虽然进行本次实验花费了大量的时间和遇到了许多困难，但在最后看到客户端的虚拟机能在自己一步步搭建的智能路由器的作用下上网，感到由衷的自豪，成就感很足。

这学期的计算机网络课程是一段非常有意义的学习历程。在这门课程中，我深入学习了计算机网络的基本原理、网络协议、网络拓扑结构、网络安全等知识。一方面，这门课程让我更加了解了计算机网络的运作方式，对网络的工作原理有了更加深刻的理解。我学习了网络分层模型，了解了每一层的职责和作用，并且掌握了如何在不同的网络层之间进行数据的传输和交互。另一方面，这门课程还让我更加关注网络安全问题。我学习了如何防范网络攻击，如何保护网络数据的安全，以及如何应对网络安全事件。这些知识都是非常实用的，在日常工作和学习中都能够得到应用。总的来说，学习计算机网络这门课程是一段非常有意义的学习历程，让我对计算机网络有了更加深刻的理解，并且增强了我的网络安全意识。