Sub-Internet

Table of Contents

1. Introduction 1
2. Regional Area Network 1
3. Gateway PC 2
   3.1. Hardware 2
   3.2. Software 2
4. Caching Proxy Software 3
5. Gateway Operation 3
   5.1. Web Access 3
   5.2. File Transfer 5
   5.3. eMail 5
   5.4. etc 5
Appendix A Gateway PC 6
   A.1. Set Up Raspberry Hardware 6
   A.2. Load Berryboot 7
Appendix B Caching Proxy Software 17
   B.1. Set up Squid Proxy Software 17
   B.2. Peer Mode Test / Verify 23
      B.2.1. Windows Peer Proxy 23
      B.2.2. Linux Peer Proxy 28
   B.3. Gateway Mode Test / Verify 29
      B.3.1. Setup 29
      B.3.2. Basic Test / Verify 29
Appendix C Supplemental Information 34

1. Introduction

This document presents an outline for setting up a Squid based Caching Proxy to serve as the Gateway of a Regional Area Network established by the EzIP technology. This configuration supports Internet-like services to a geographical area with up to 256M publicly manageable IoTs from only one IPv4 address.

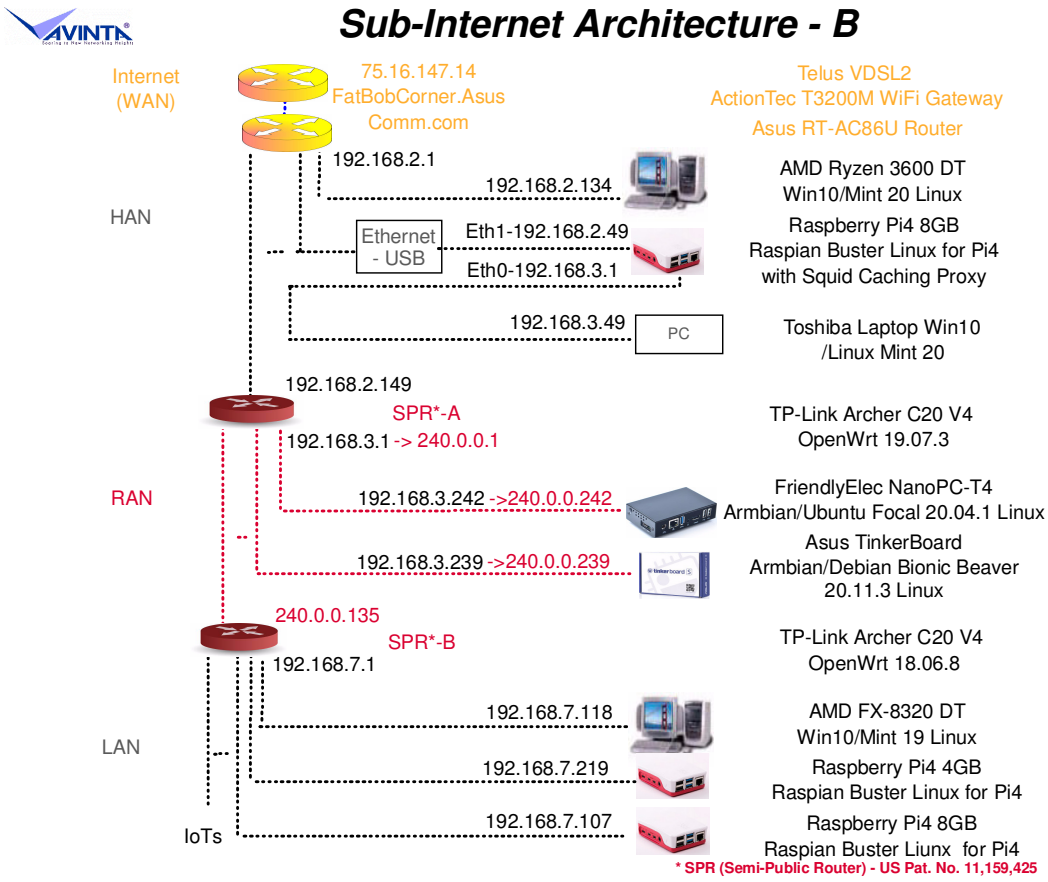
2. Regional Area Network

A Regional Area Network (RAN) is made up with Semi-Public Routers (SPR) operating with 240/4 netblock addresses. Basically, the SPRs are derived from existing IPv4 routers by disabling the portion of their program code that has been disabling the use of the 240/4 netblock. Since the 240/4 address is not recognized by existing Internet routers, packets with such addresses will be rejected. On the other hand, this enables SPRs to utilize the existing router hand shake protocols for building up their on routing tables to form the RAN. Below is a feasibility configuration of a RAN.

https://www.avinta.com/phoenix-1/home/RegionalAreaNetworkArchitecture.pdf

3. Gateway PC

Note: The discussion below refers to a configuration document named, SubInternetArchitectureB. Its main diagram is attached here for now.



The full document will be posted online to be accessed via its URL for more detials.

3.1. Hardware

The main hardware is a Raspberry Pi 4 8GB single board computer:

https://www.raspberrypi.org/blog/8gb-raspberry-pi-4-on-sale-now-at-75/

3.2. Software

Its Operating System (OS) software is the Berryboot images at:

https://berryboot.alexgoldcheidt.com/images/

For the first time reader, please follow through Appendix A. for details of where and how to procure the hardware then set them up, followed by the procedure for downloading and installing the OS and eventually check out the basic PC operations.

4. Caching Proxy Software

We will be using the Squid Cache Proxy software for the Gateway functions. It may be found at:

https://en.wikipedia.org/wiki/Squid

For the first time reader, please follow through Appendix B. for specific steps of installing the Squid software and checking out its basic operations.

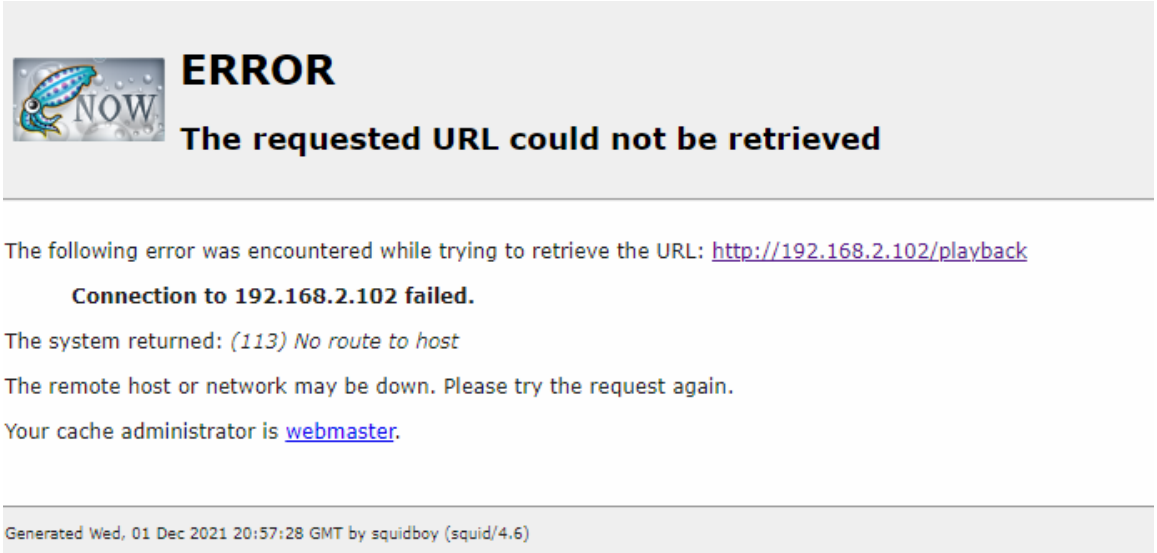
5. Gateway Operation

The below description is based on a Raspberry Pi 4 8GB single board PC (192.168.2.49 and 192.168.3.1) operating with the Squid Cache Proxy software in the Gateway mode. 192.168.3.1 is the IP of the Ethernet port (eth0) on the Raspberry Pi 4 8GB and this acts as a gateway/router for all computers and networks connected to this IP. Originally it was indented that the Ethernet-USB adapter (eth1) perform this function but experimentation showed that the transparent Squid functionality worked through eth0 and not through the Ethernet-USB adapter (eth1) so the Ethernet-USB adapter was used as 192.168.2.49. All computers and networks connected to 192.168.3.1 will pass through the Raspberry PI 4 8Gb with Squid transparent proxy server. The Raspberry Pi 4 8Gb was set up to act as a router/gateway and provide DHCP addresses to computers (currently a single PC) connected to the 192.168.3.1 Ethernet port. The PC used herein may be replaced by the 240/4 netblock RAN as described in Section 2 to present the overall sub-Internet characteristics,

5.1. Web Access

If IP 192.168.3.1 is up any Web browser will pass through the Squid transparent proxy server for HTTP protocol ONLY on port 80 which will be redirected to the Squid transparent proxy server on port 3129. Valid Internet addresses will return a Web page. Invalid Internet addresses will return an error page from Squid.

Protocol	Incoming Port	Redirected	To
HTTP/192.168.3.1	**80**	**Yes**	**SQUID:3129**



If IP 192.168.3.1 is up any Web browser will pass through the Raspberry PI 4 8Gb but NOT pass through the Squid transparent proxy server for HTTP protocol if the port is NOT 80 which will go directly to the Internet. Invalid Internet addresses will return an error page from the Internet.

Protocol	Incoming Port	Redirected	To
HTTP/192.168.3.1	**Not 80**	**No**	**Direct to Web page**

If IP 192.168.3.1 is up any Web browser will pass through the Squid transparent proxy server for HTTPS protocol ONLY on port 443 which will be redirected to the Squid transparent proxy server on port 3129. Valid Internet addresses will return a Web page. Invalid Internet addresses will return an error page from Squid.

Protocol	Incoming Port	Redirected	To
HTTPS/192.168.3.1	**443**	**Yes**	**SQUID:3129**

If IP 192.168.3.1 is up any Web browser will pass through the Raspberry PI 4 8Gb but NOT pass through the Squid transparent proxy server for HTTPS protocol if the port is NOT 443 which will go directly to the Internet. Invalid Internet addresses will return an error page from the Internet.

Protocol	Incoming Port	Redirected	To
HTTPS/192.168.3.1	**Not 443**	**No**	**Direct to Web page**

If IP 192.168.3.1 is down then the Web browser will not be able to connect to the Internet.

5.2. File Transfer

If IP 192.168.3.1 is up FTP will pass through the Raspberry PI 4 8Gb but NOT pass through the Squid transparent proxy server. Valid FTP addresses will connect to a valid FTP site. Invalid FTP addresses will not connect to a FTP site.

Protocol	Incoming Port	Redirected	To
FTP/192.168.3.1	**FTP Ports**	**No**	**Direct to Web page**

If IP 192.168.3.1 is down then FTP will not be able to connect to an FTP site.

5.3. eMail

If IP 192.168.3.1 is up then email will pass through the Raspberry PI 4 8Gb but NOT pass through the Squid transparent proxy server and be functional.

Protocol	Incoming Port	Redirected	To
email/192.168.3.1	**email Ports**	**No**	**Direct to Web page**

If IP 192.168.3.1 is down then email will not be functional.

5.4. etc.

If IP 192.168.3.1 is up then etc. will pass through the Raspberry PI 4 8Gb but NOT pass through the Squid transparent proxy server and be functional.

Protocol	Incoming Port	Redirected	To
Etc./192.168.3.1	**Etc. Ports**	**No**	**Direct to Web page**

If IP 192.168.3.1 is down then etc. will not be functional.

Appendix A Gateway PC

A.1. Set Up Raspberry Hardware

The hardware components used to build the Squid Caching proxy are:

a. The Raspberry PI I used was Raspberry Pi4 Model B Single Board Computer w/ 8GB DDR4 RAM
https://www.memoryexpress.com/Products/MX00113949

b. Micro Connectors Raspberry Pi 4 Acrylic Case Kit w/ 3 Heat Sinks, 40mm Cooling Fan, 15W USB Power Supply
https://www.memoryexpress.com/Products/MX79419

c. Elite HDMI 1.4 to Micro HDMI Cable, 6ft
https://www.memoryexpress.com/Products/MX33073

d. Vantec NexStar TX External 2.5in SATA HDD Enclosure, USB 3.0, Black
https://www.memoryexpress.com/Products/MX51952

e. Western Digital Green Series SATA III 2.5in Solid State Drive, 120GB
https://www.memoryexpress.com/Products/MX75426

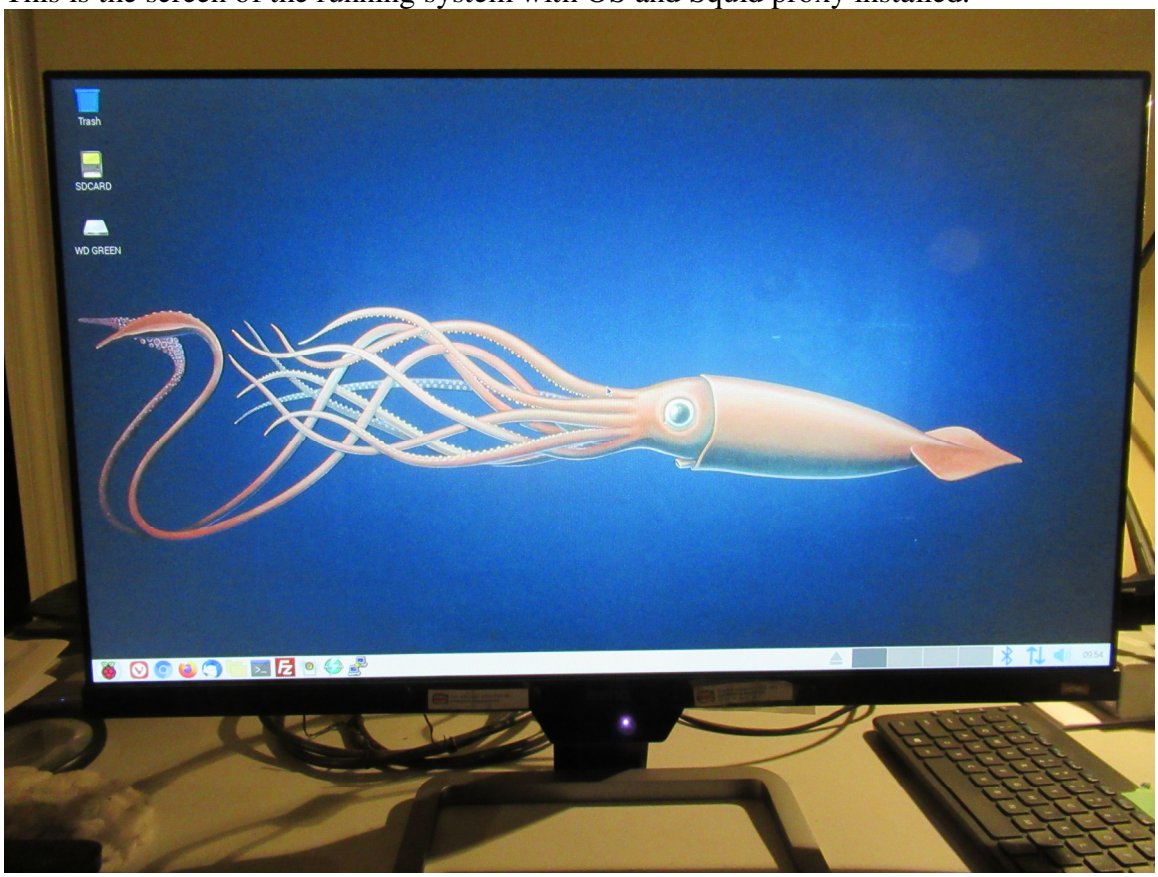
f. SanDisk Ultra microSDHC UHS-I Card w/ SD Card Adapter, 16GB
https://www.memoryexpress.com/Products/MX72326

g. Logitech K400 Plus Wireless Multimedia Keyboard w/ TouchPad
https://www.memoryexpress.com/Products/MX60065

h. IOGear USB 3.0 to Gigabit Ethernet Adapter
https://www.memoryexpress.com/Products/MX54704

This is what the assembled system looks like

This is the screen of the running system with OS and Squid proxy installed.



When the Squid proxy is running to our satisfaction it can run “headless” meaning no screen and keyboard are needed. Communication will be through Putty (ssh) and all the Squid proxy will require is an ethernet connection. This however is an option and it is much easier to have a screen and keyboard if you want to do any serious maintenance or upgrades.

Pre-SquidCaching Proxy Software: Isn’t the below material about Raspberry’s OS?

Yes it is, but you need this before you can install Squid.

A.2. Load Berryboot

We will use Berryboot as a bootloader, Berryboot will let you load multiple Raspberry PI 4 images and let you move save/delete them onto/from the SSD. This way you can try out multiple Linux OS images.

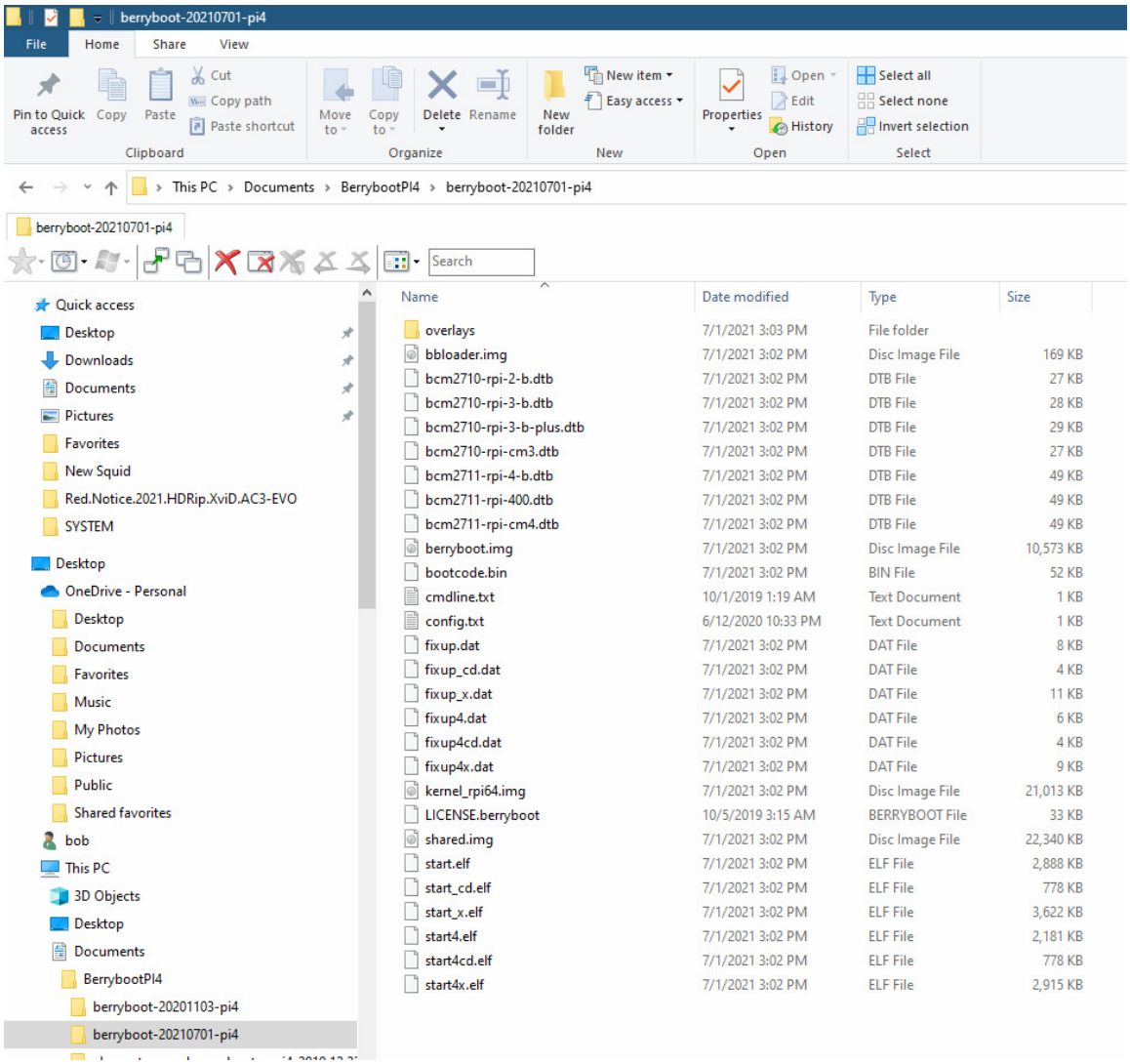
Another advantage of Berryboot is it let’s you choose which medium you want to store your OS images on, you can use the SDcard that Berryboot runs on, a USB memory stick, a hard disk or a solid state disk, SSD. I prefer an SSD for speed, disk space and reliability.

On your Windows PC download Berryboot from here:
https://berryterminal.com/doku.php/berryboot

The Berryboot file is called berryboot-20210701-pi4.zip

On Windows use WINRAR to unzip the file.

This will produce a folder called berryboot-20210701-pi4 and will contain the following files:



These files have to be transferred to the microSDHC card.

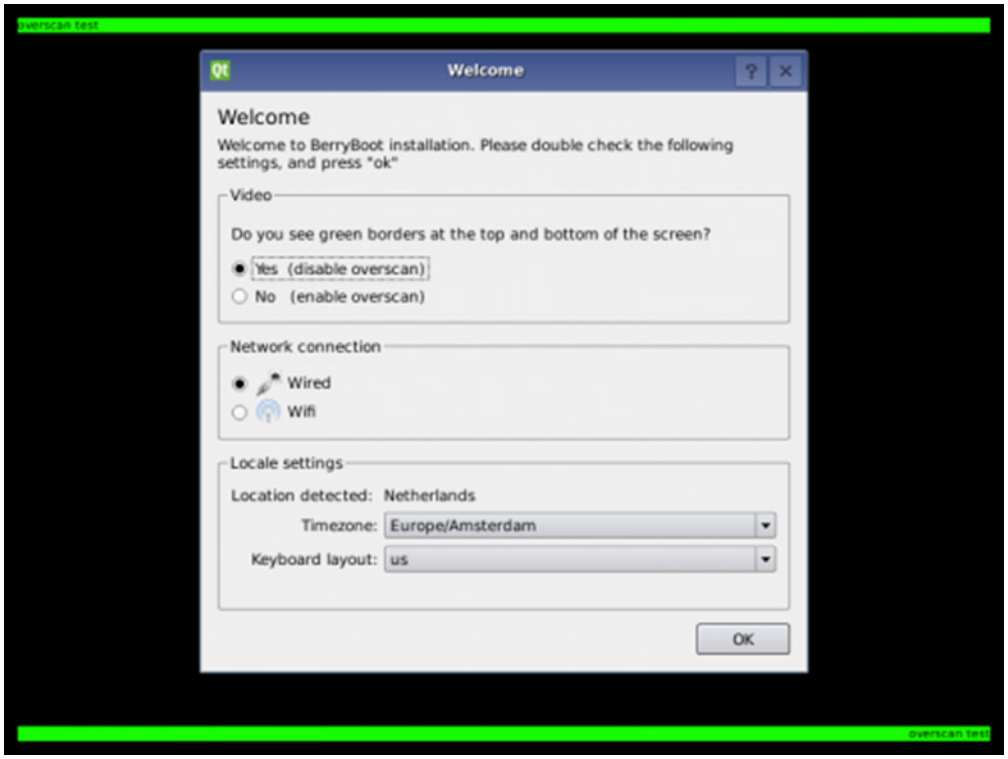
If you don't have a memory card reader/writer you will need to buy one. I prefer one that lets you write a variety of memory card types. Here is an example of a memory card reader/writer. The memory card reader/writer plugs into a USB port on your Windows

computer. The size of the microSHCD card can be as small as 8Gb and there is no need to go larger than 32Gb. So 8-16-32 Gb microSHCD is all you need.

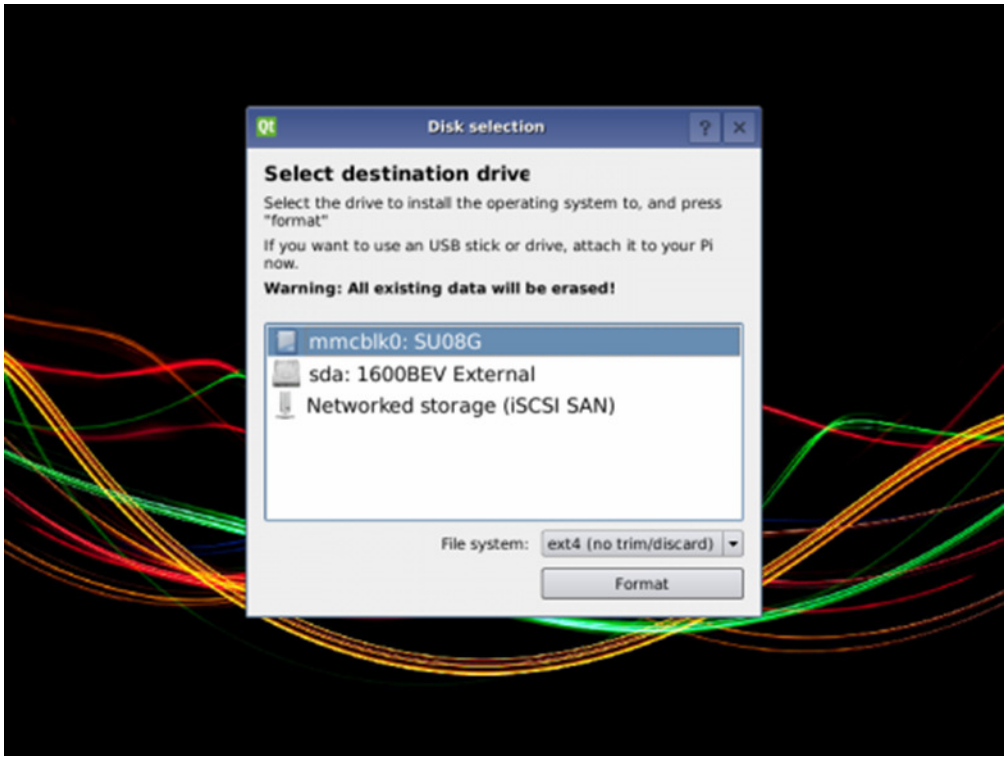
Most microSHCD cards come with an adapter that lets you plug your microSDHC card into the SHCD slot on your memory card reader/writer. I prefer this as the microSDHC card is fragile and the adapter offers more protection. The microSHCD card will go into a microSHCD slot on the Raspberry Pi 4 so you have to careful when you insert it into the Raspberry. It can only go in one way into the Raspberry so if it is not going in all the way you have to flip the microSHCD card over. If you force it in the wrong way you stand a chance of breaking the microSHCD card.

Once you have the microSDHC card inserted into your memory card reader/writer it will show up in your Windows files system as empty. Go to the folder berryboot-20210701-pi4, click the first file, then Ctrl-a from your keyboard, right click on your mouse and click Copy. Move to the berryboot-20210701-pi4 folder and Paste in the files. Now unmount and eject the microSDHC from your Windows file system. Remove the microSDHC card from the memory card reader/writer.

With your Raspberry Pi 4 hardware assembled but in a powered down state insert the microSDHC card into the microSDHC slot of the Raspberry carefully, don't force it, lol. If it does not go in easily it is upside down. Make sure your ethernet is connected, USB-Sata enclosure containing your SSD and is plugged into a USB 3 port, your keyboard dongle is plugged into a USB 2 port for your keyboard, your keyboard switch is powered on and your HDMI cable is connected to your moniter/TV HDMI port and your moniter/TV is on. Now plug your Raspberry Pi power supply in and hopefully your system will come to life. The first screen you will see will look like the following, you can enable/disable overscan, your network connection should be wired if you plugged in an ethernet wire, if not you can hook up to wifi (never tried this, always used the wired connection) and set your locale settings, timezone and keyboard layout.



Once you are happy with your settings hit “OK” and move onto the second screen. The second screen let’s you select your destination drive. Here is where you will select your SSD. On the screen below mmcblk0 is your microSHCD card, sda is an external disk and the third entry is networked storage. Your SSD will be the second entry on the screen so choose that. Once you select the SSD and if it has never been used before Berryboot will change the SSD format from exFat to ext4. If it has previously been formatted to ext4 and it contains files you can either delete the files and start with a clean SSD or keep the files and use them.



Now is the time to add a Berryboot image to your SSD. You can choose directly from the Berryboot screens or you can down load from the Berryboot server on the Internet. The Berryboot screen will look like the screen below but keep in mind that this will change over time with new Berryboot images being added and old ones being removed.

I prefer to load Berryboot images from the Berryboot server at:
https://berryboot.alexgoldcheidt.com/images/

Please note that Berryboot images are not just Linux OS images, they also include Linux images (Raspbian, Debian, Ubuntu, Arch, Centos etc.), Learning, Gaming, Media Center, Networking, Utilities etc. You must be sure to get images that will run on a Raspberry Pi 4. There are also images for Raspberry Pi 1, 2 and 3 plus Odroid C images (another single board computer).

I downloaded the
raspberry_pi_os_buster_desktop_full_rpi2_rpi3_rpi4_2021.03.04_berryboot.img
from Berryboot server to my Windows computer.

These are the official Debian Buster optimized images for the Raspberry Pi 2/3/4 and are also called Raspian.

The full Desktop is called
raspberry_pi_os_buster_desktop_full_rpi2_rpi3_rpi4_2021.03.04_berryboot.img
I install this desktop to my Raspberry Pi 4 SSD.

On your Windows computer use WINRAR to un-compress the file and you will see a folder called
raspberry_pi_os_buster_desktop_basic_rpi2_rpi3_rpi4_2021.03.04_berryboot.img.

Inside these folders are the file image:
raspberry_pi_os_buster_desktop_basic_rpi2_rpi3_rpi4_2021.03.04_berryboot.img
This image has to be written to a USB memory stick.

Insert your USB memory stick into a USB port on your Windows computer can copy the two image files to the USB memory stick. Unmount and eject the USB memory stick on Windows. Insert the USB memory stick into a USB port on the Raspberry Pi 4 and press the CANCEL button on the Add OS screen. If you can see an Edit Menu button press that and you should be able to see the Add OS button, this will allow you to install an OS from the USB memory stick. If you can see an Add OS button right after the CANCEL this will allow you to install an OS from the USB memory stick. If you cannot get to the Add OS button at this time just reboot the Raspberry Pi 4 from Berryboot or as a last resort power the system off and back on. When the system comes back on press the Edit Menu button and it will take you to the Add OS button.

Assuming you now are “somehow” at the Add OS button, hold your left mouse button downfor a while and click on “Copy OS from USB stick”. This will allow you to copy the OS(s) from your USB stick one at a time. Once you have copied the OS(s) set the one you want to start up as default by pressing the Set Default button. Then press the Exit button and the Raspberry Pi 4 will reboot and hopefully your system will start up for the first time.

If you are having trouble installing from the USB memory stick then just install an OS from the Berryboot loader. The first one in the Add OS menu is OK … usually a Debian

Linux. After it is installed and running you can try the USB memory stick again. You can have multiple OS images and delete the ones you are no longer using.

If you are “absolutely” stuck and nothing at all seems to work with your Raspberry Pi system power it down, take the microSDHC card and the SSD out of the Raspberry and plug them into a USB port on your Windows PC. Reformat the microSDHC card to FAT if less than 64Gb and exFAT if more than 64Gb (microSDXC card). If the format worked, it is now blank, copy the Berryboot files back onto it. Reformat the SSD to exFAT because it will be bigger than 64Gb. If the format worked, the SDD is now blank.

If you cannot get the reformatting to work on the microSDHC download sdcard formatter.
https://www.sdcard.org/downloads/formatter/

If you cannot get the reformatting to work on the SSD on Windows and you have access to a Linux system install ‘gparted’ on that and reformat your SSD there. Be very careful that you are formatting/partitioning the SSD and not the wrong device.

A Windows ISO version of “gparted” can be downloaded and written to a DVD/CD or USB memory stick and run as a standalone image by rebooting from the DVD/CD or USB memory stick on Windows.
https://gparted.org

When your system comes up for the first time some systems will start a task running that you interact with, raspberry_pi_os_buster has this. You will be asked for your country, language, time zone, to change your password (hay every Raspberry Pi system has a

default password of ‘raspberry’, lol) and then your software will update and you will be asked to reboot your system.

After the reboot bring up a terminal (command line). Type the following.
$ sudo -i
# <your favorite editor> update
Once in the update file add the lines:
apt update
apt upgrade
apt autoremove
Save the update file
Change the mode of the update file
chmod 777 update
Now run the file and your software will update. This should be minor since we updated the software before the previous reboot.
./update

You can run the update file every time you reboot to update your software.
Now install Thunderbird (email), FTP server and FTP client. These are for email and sending and receiving files to/from other computers and external disks on routers in our system.
apt install thunderbird proftpd-basic filezilla
Install Putty an SSH client and Synaptic a graphics program to install other programs.
apt install putty synaptic
Now run “raspi-config”
raspi-config

Under Interface Options enable SSH, under Localisation Options set Timezone and WLAN Country.
Exit the terminal.
Start up Chromium and turn on sync with your Google account, you will want your bookmarks and extensions.

Start up Thunderbird and setup up your email. You can use Filezilla to get your address book from somewhere else on your system.

Use Chromium and go to https://angryip.org for Angry IP Scanner. Angry IP Scanner will let you scan for computers/routers etc. on your network.

Download the DEB Package for ARM … 3rd one in the table.

Download version 3.7.6 below or browse more releases or even older releases.

- DEB Package for Ubuntu/Debian/Mint, 64-bit
- RPM Package for Fedora/RedHat/Mageia/openSUSE, 64-bit
- DEB Package for Ubuntu/Debian/Mint, any architecture (e.g. 32-bit or ARM)
- Executable Jar - you need to provide your own swt.jar to classpath

The file will be downloaded to /home/pi/Downloads. Open this directory in your File Manager and see the file ipscan???.deb. Right click on the file and then choose Package Install. The installer will ask for your password and then install the software. Previous versions of Debian-variants-Linux used gdebi to install the software and required you to install Java-JDK and other libraries. This installer does everything for you, nice. Each version of Linux can be different as you can see above, for example Redhat uses RPMs to install.

Appendix B Caching Proxy Software

B.1. Set up Squid Proxy Software

Set up our Squid Proxy software on the Raspberry Pi 4. Most of changes to the Squid Proxy conf files and software added where taken from an Internet article at:

https://medium.com/@bindassbasanta/squid-proxy-cache-using-raspberry-pi-diy-45720395ae21

called "Squid Proxy Cache — Using Raspberry Pi (DIY)"
Change the hostname from "raspberrypi" to "squidboy". Or whatever you like.
Bring up a terminal (command line). Type the following.
$ sudo -i
# cd /etc
# <your favorite editor> hosts
Change "raspberrypi" to "squidboy"
<your favorite editor> hostname
Change "raspberrypi" to "squidboy"
Now give your Raspberry Pi – Squid proxy server a static IP address.
Make a backup of the dhcpcd.conf file called dhcpcd.conf.org
cp dhcpcd.conf dhcpcd.conf.org
<your favorite editor>dhcpcd.conf

Where the dhcpcd.conf for says
Example static IP configuration I changed it to:
interface eth1
static ip_address=192.168.2.49/24
static routers=192.168.2.1
static domain_name_servers=192.168.2.1

```
# Example static IP configuration:

interface eth1
static ip_address=192.168.2.49/24
static routers=192.168.2.1
static domain_name_servers=192.168.2.1
static domain_search=
```

eth1 is the name of the USB-Ethernet adapter and this can have different name based on your USB-Ethernet adapter

You can change to the addresses you want for your system.
Reboot the Raspberry Pi 4.

When the system comes up open a terminal (command line).
$ sudo -i
# ifconfig eth1
This will verify both the hostname and static IP address change.

```
root@squidboy:/etc# ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.49  netmask 255.255.255.0  broadcast 192.168.2.255
        inet6 fe80::17b9:24d9:46d9:1506  prefixlen 64  scopeid 0x20<link>
        ether 80:3f:5d:e1:33:ba  txqueuelen 1000  (Ethernet)
        RX packets 34611  bytes 41783156 (39.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7354  bytes 759026 (741.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

OK we will install Squid now.
apt install squid

We will not make any changes to the squid.conf yet, we will reboot and see if we can use the Squid proxy server.
reboot

On your Windows computer Click on Settings and type “proxy” in the search box. Fill in the proxy for Windows, looks like this, IP is 192.168.2.49 and the port is 3128 for the Squid proxy server. Hit the Save button to save.

Settings

Home

Find a setting

Network & Internet

Status

Ethernet

Dial-up

VPN

Airplane mode

Mobile hotspot

Proxy

Proxy

Automatic proxy setup

Use a proxy server for Ethernet or Wi-Fi connections. These settings don't apply to VPN connections.

Automatically detect settings

On

Use setup script

Off

Script address

Save

Manual proxy setup

Use a proxy server for Ethernet or Wi-Fi connections. These settings don't apply to VPN connections.

Use a proxy server

On

Address: 192.168.2.49

Port: 3128

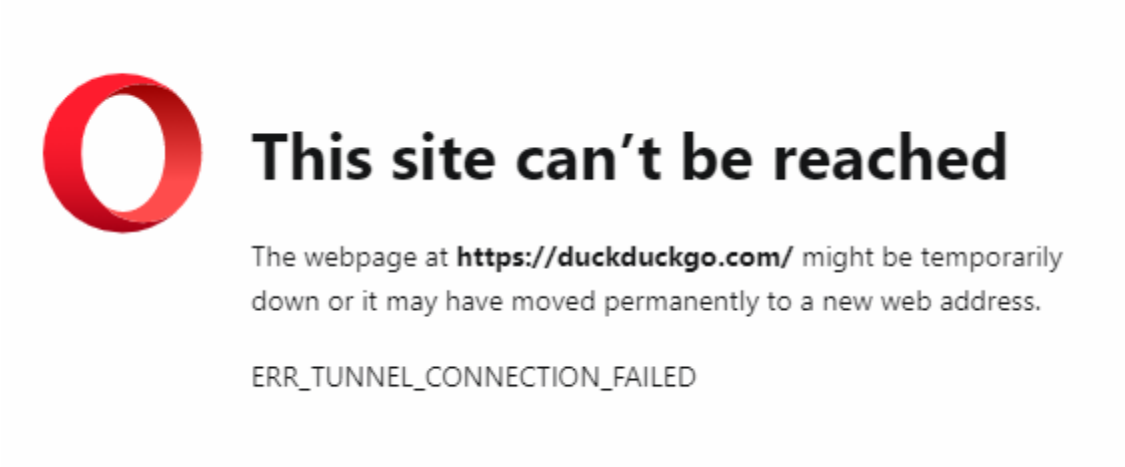
Use the proxy server except for addresses that start with the following entries. Use semicolons (;) to separate entries.

*.local

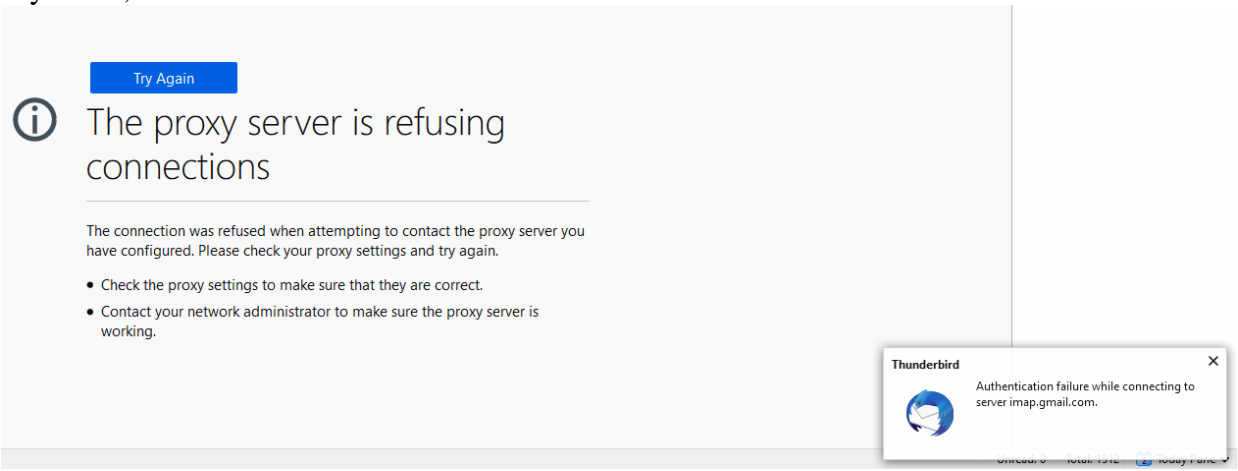
Don't use the proxy server for local (intranet) addresses

Save

After this my Opera browser could only reach local websites like my own website, all non-local websites were blocked.



My email, Thunderbird was also blocked.



Go back to the Raspberry Pi 4 and we will edit the /etc/squid.conf file. These are the changes from "Squid proxy Cache — Using Raspberry Pi (DIY)" webpage. Bring up a terminal (command line). Type the following.
$ sudo -i
# cd /etc/squid

Copy the original file to squid.conf.org
cp squid.conf squid.conf.org
<your favorite editor>squid.conf
Remove the comment # from the line #http_access allow localnet
It should now read: http_access allow localnet
Write the squid.conf and reboot.
reboot

After this change everything worked through the Squid proxy server from the HAN(192.168.2.0/24), RAN(240.0.0.0/4) and LAN (192.168.7.0/24).
I will describe how to set up Squid proxy clients later in this document.

Go back to the Raspberry Pi 4 and we will edit the /etc/squid.conf file again. These are the changes from “Squid proxy Cache — Using Raspberry Pi (DIY)” webpage. Bring up a terminal (command line). Type the following.

```
$ sudo -i
# cd /etc/squid
# <your favorite editor>squid.conf
```

Find: acl localnet section and I added the following:

```
acl localnet src192.168.2.0/24          # HAN
acl localnet src240.0.0.0/4             # RAN
acl localnet src192.168.7.0/24          # LAN
```

Find: # dns_v4_first off remove the # symbol and change “off” to “on”

```
dns_v4_first on
```

Find: # cache_mem 256 MB and remove the # symbol

```
cache_mem 256 MB
```

Find: # maximum object size 4 MB and remove the # symbol and change size to 4096 MB

```
maximum object size 4096 MB
```

Find: # maximum object size in memory 512 KB and remove the # symbol and change size to 8192 KB

```
maximum object size in memory 8192 KB
```

Find: # cache_dir ufs /var/spool/squid 100 16 256 and remove the # symbol and change 100 to 8192

```
cache_dir ufs /var/spool/squid 8192 16 256
```

Write the squid.conf, save a back up somewhere on your networks as you don’t want to lost the file after all that work and reboot.

```
# reboot
```

This is a diff comparison of the original and edited version of squid.conf so you can see the changes made.

```
1196,1199d1195
<acllocalnetsrc 192.168.2.0/24          # HAN
<acllocalnetsrc 240.0.0.0/4             # RAN
<acllocalnetsrc 192.168.7.0/24          # LAN
<
1411c1407
<http_access allow localnet
---
> #http_access allow localnet
3367c3363
<cache_mem 256 MB
---
> # cache_mem 256 MB
3375c3371
<maximum_object_size_in_memory 8192 KB
---
> # maximum_object_size_in_memory 512 KB
3481c3477
```

```
<maximum_object_size 4096 MB
---
> # maximum_object_size 4 MB
3639c3635
<cache_dirufs /var/spool/squid 8192 16 256
---
> #cache_dir ufs /var/spool/squid 100 16 256
8157c8153
< dns_v4_first on
---
> # dns_v4_first off
```

After this change everything still worked through the Squid proxy server from the HAN(192.168.2.0/24), RAN(240.0.0.0/4) and LAN (192.168.7.0/24).

These are the changes from "Squid proxy Cache — Using Raspberry Pi (DIY)" webpage to setup Webmin. Bring up a terminal (command line). Type the following.

```
$ sudo -i
# apt -f install
# apt install mariadb-client mariadb-server
# apt -y install apache2 apache2-suexec-custom libnet-ssleay-perl libauthen-pam-perl libio-pty-perl apt-show-versions samba bind9 webalizer locate mariadb-server
# apt install squid-cgi
# cd
# pwd
# mkdir installed-packages
# cd installed-packages
```

Download Webmin

```
# wget http://www.webmin.com/download/deb/webmin-current.deb
```

Install Webmin

```
dpkg -i webmin-current.deb
```

Once Webmin is installed, you can use raspberry pi IP address as following to, note this is different from the address in the Web article which did not work for me.

https://localhost:10000/

The default username and password is:

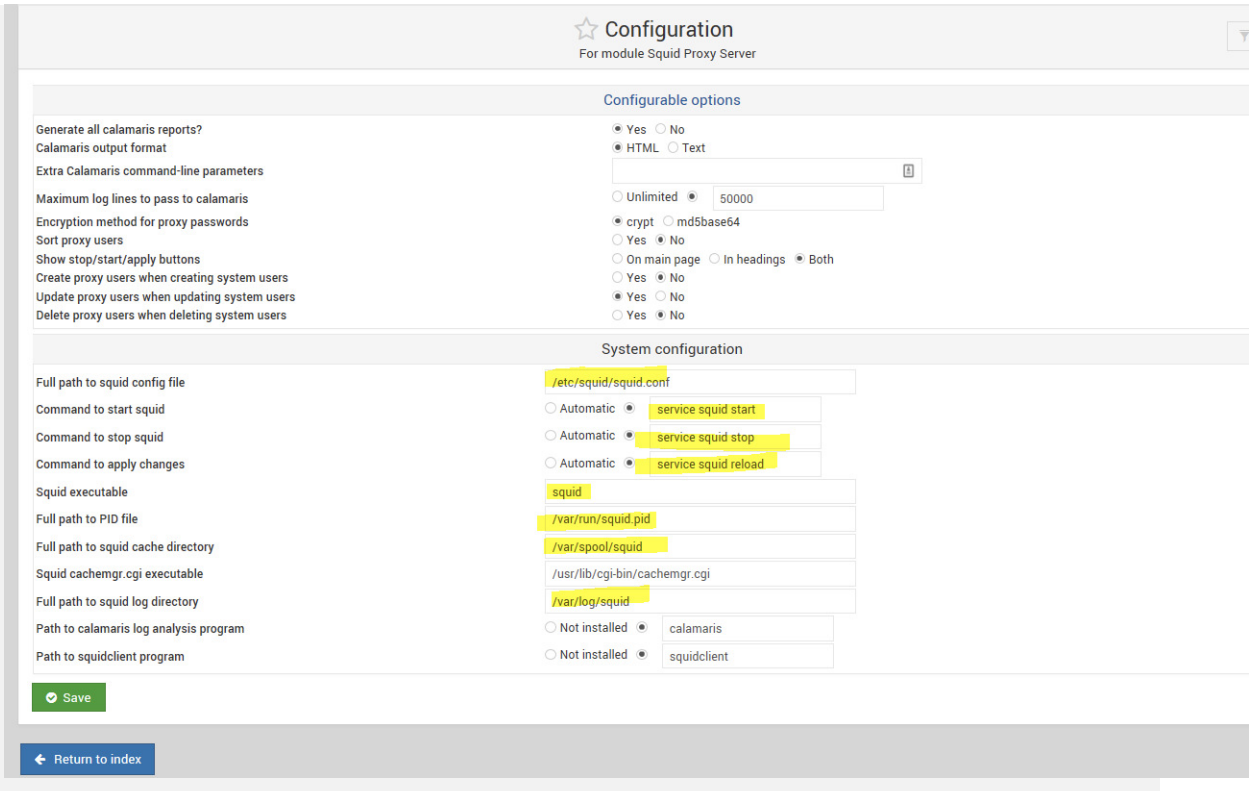
Username: pi

password: "what you changed it too" "raspberry" if you did not change it.

In Webmin, you need to adjust SQUID to use it.

Click on edit configuration for SQUID.

Make changes everywhere you see SQUID3 to SQUID as shown below



and press save.

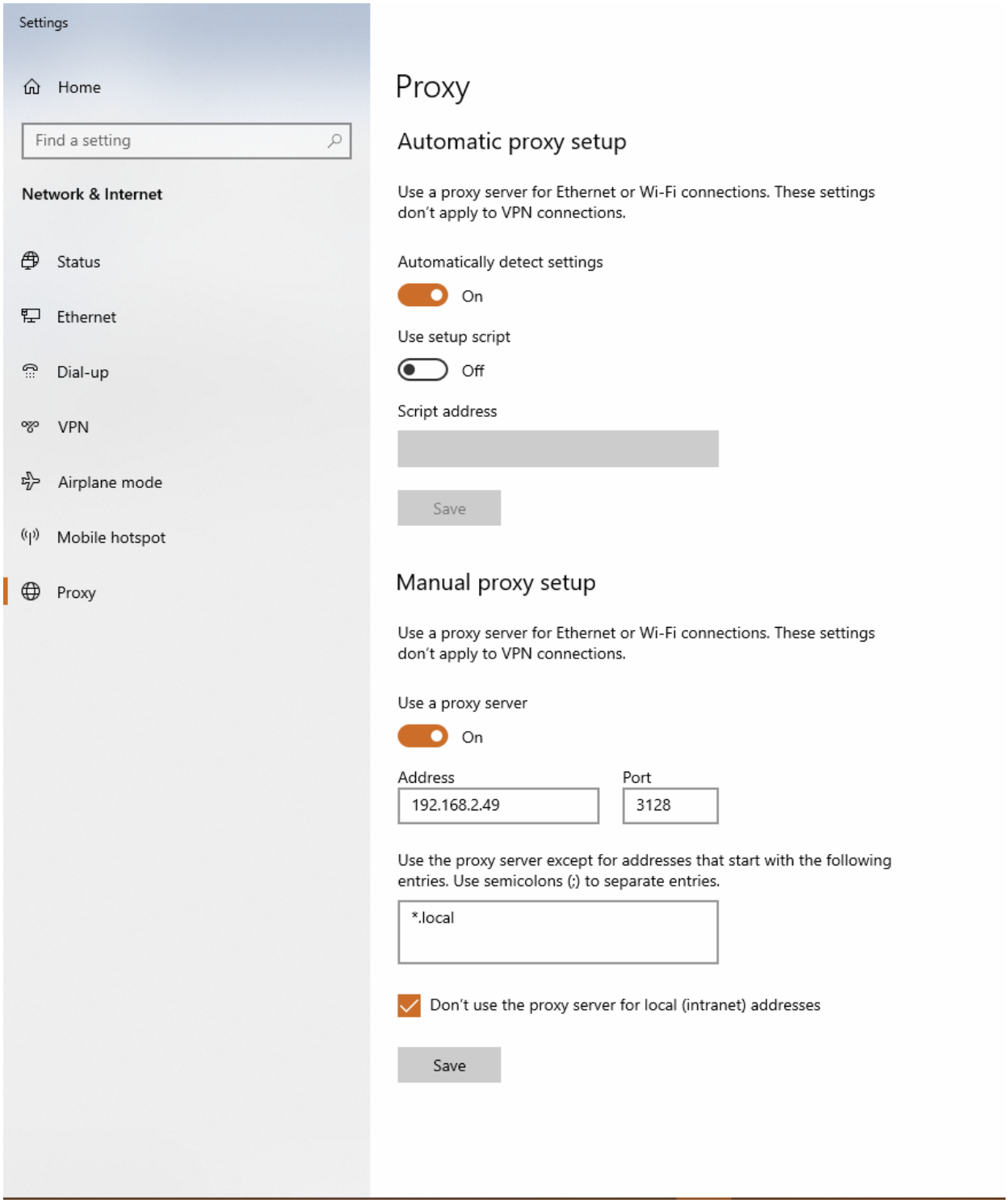
reboot

After this change everything worked through the Squid proxy server from the HAN(192.168.2.0/24), RAN(240.0.0.0/4) and LAN (192.168.7.0/24). Use Webmin to verify squid proxy as an active server.

B.2. Peer Mode Test / Verify

B.2.1. Windows Peer Proxy

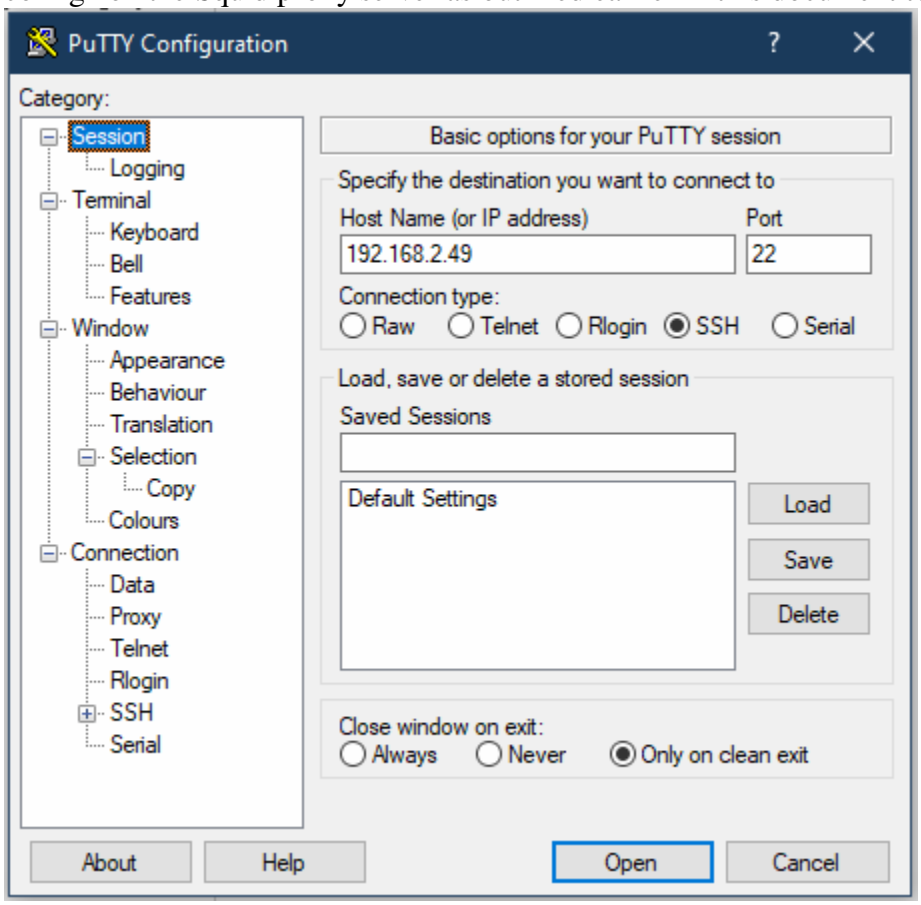
Once you have the Squid proxy server up we can begin testing. We will begin with Windows 10 on 192.168.2.134 Click the Start button then Settings when then type “proxy” in the search box. From there you can get to the screen to set up your proxy. Fill in the Squid proxy IP address and port (3128) unless you modified it with Webmin. Should look like this.



After setting up the proxy verify you can still use your Internet browsers, email, FTP etc.

Assuming everything is working now is the time to install Putty, it is available from many places on the Internet for free. Start Putty and set the font (the default is too small, under Window->Appearance) then enter the IP of your proxy server, port 22 for SSH communication, highlight Default Settings and Save. You only need to do this the first

time if you Save. Now click Open and the first a box will appear asking you to accept keys from the Squid proxy server, which you say Yes to. If you did not enable SSH on the Squid proxy server you will not be able to connect and you will have to run “raspi-config” on the Squid proxy server as outlined earlier in this document to enable SSH.



Once you are connected to the Squid proxy server log in as ‘pi’ then your ‘password’ and the type ‘sudo -i’ to enter supervisor mode. Next type ‘netstat -a | grep 3128’ to look for connections on port 3128 which the Squid proxy server is using. Should look something like the following.

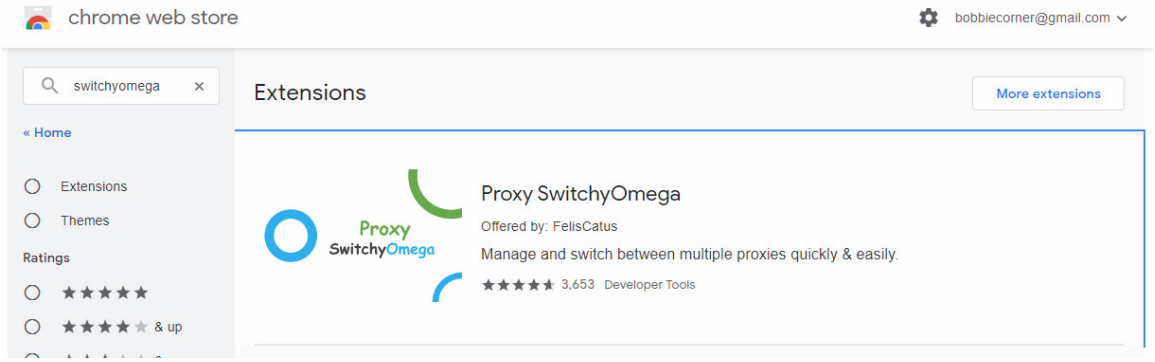
```
pi@squidboy: ~
login as: pi
pi@192.168.2.49's password:
Linux squidboy 5.4.73v64 #2 SMP PREEMPT Tue Nov 3 16:11:05 CET 2020 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Mar  6 17:34:44 2021
pi@squidboy:~ $ sudo -i
root@squidboy:~# netstat -a | grep 3128
tcp6       0      0 [::]:3128               [::]:*                  LISTEN
tcp6       0      0 192.168.2.49:3128       toshiba:50197           ESTABLISHED
tcp6       0      0 192.168.2.49:3128       ASPIRE:55649            ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:2527        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:2515        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:2529        FIN_WAIT2
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:2524        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       ASPIRE:55695            ESTABLISHED
tcp6       0      0 192.168.2.49:3128       ASPIRE:55689            ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:2526        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:2479        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:2512        ESTABLISHED
root@squidboy:~#
```

Note I can see 3 of my computers using the Squid proxy server. If you start up/stop Internet browsers you will see connections being added and remove by re-typing the netstat command.

If you are running another browser from the RAN(240.0.0.0/4) or LAN(192.168.7.0/24) use will not see the computer name but OpenWrt240 as it is the router communicating to the Squid proxy server.

```
pi@squidboy: ~
netstat -a | grep 3128
tcp6       0      0 [::]:3128               [::]:*                  LISTEN
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36926        TIME_WAIT
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36960        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3459        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3478        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36884        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36968        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3460        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36892        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       ASPIRE:57196            ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36924        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36896        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36948        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36908        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3479        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3475        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36880        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36966        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36922        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36916        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36964        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36914        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3007        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36878        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36912        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3473        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36940        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36888        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36950        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3474        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36898        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36882        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3461        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36946        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36954        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36958        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36962        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36894        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3480        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36906        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3472        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36910        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36930        TIME_WAIT
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36956        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       toshiba:50238           ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36936        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36938        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3481        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36900        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       COOLBASTARD:3471        ESTABLISHED
tcp6       0      0 192.168.2.49:3128       OpenWrt240:36886        ESTABLISHED
root@squidboy:~#
```

One more thing before moving onto setting up Squid proxy clients on Linux is the Proxy SwitchyOmega extension that I founding while setting up Squid proxy clients on Linux. You can find this extension in the Chrome Web Store by typing “switchyomega” in the search box and you should see the following.



You can install this extension in most any Internet browser, Chrome, Opera, Firefox, Vivaldi, Microsoft Edge etc. and it will let you switch between your system proxy, another proxy or bypass your proxy and connect directly. Our system proxy and another proxy are currently the same but there is nothing stopping you from assigning a different proxy to the “another proxy”. Once this extension is set up in any Internet browser and you are logged in with the same account on multiple computers it will add the extension to other computers (most of the time, lol). Note however this applies to Internet browsers only, so if you set up a system proxy and you try to read email through something like Thunderbird you have to go through the Squid proxy server, you cannot go directly. All this is easy to prove, just shut off the Squid proxy server and you can connect directly with an Internet browser using this extension but your Thunderbird email will time out.

B.2.2. Linux Peer Proxy

Now we will setup a Linux proxy server client on 192.168.2.134, start up your Linux system.

After the system reboots start up your Internet browser and verify that “Proxy SwitchyOmega” has been added, if not manually add it yourself. “Proxy SwitchyOmega” will have the same settings as Windows above.

Startup Putty and log into the Squid proxy server and use the “netstat -a | grep 3128” command as outlined above to verify you are using the Squid proxy server.
You need to verify connections with the “netstat -a | grep 3128” command. You can also power off the Squid proxy server and see that you are no longer connecting to the Internet … if you are then you are not using the Squid proxy server.

B.3. Gateway Mode Test / Verify

B.3.1. Setup

In this section, we will set up a configuration whereby the Raspberry Pi 4 8GB with Squid software (192.168.2.49 and 192.168.3.1) will be used as an ***inline Gateway*** for the IoTs on its downstream subnet operating ***over the 240/4 netblock***.

B.3.2. Basic Test / Verify

To begin testing the Squid transparent proxy we have to first connect eth0 as 192.168.3.1 and setup iptables for transparent Squid proxy server.

Bring up a terminal as a normal user and run crontab -e
Add the two comment lines and the crontab line to run the file squid.sh when the system reboots.

```
pi@squidboy:~ $
pi@squidboy:~ $ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
# setup up eth0 192.168.3.1 and eth1 192.168.2.49 on reboot
# setup up iptables for transparent squid
@reboot sudo -u root /root/squid.sh
pi@squidboy:~ $
```

Now Type the following.
$ sudo -i
You will be in the /root directory.
<your favorite editor> squid.sh

Edit squid.sh so it looks like the following.

```
root@squidboy:~#
root@squidboy:~# more squid.sh
#!/bin/sh
#
# /etc/sysctl.conf Configuration
# Controls IP packet forwarding
# net.ipv4.ip_forward = 1

# use eth0 192.168.3.1 for transparent squid proxy
# use eth1 192.168.2.49 to connect to our main router 192.168.2.1
# eth1 is static ip set up in /etc/dhcpcd.conf
#
ifconfig eth0 down
ifconfig eth0 192.168.3.1 netmask 255.255.255.0 up
iptables -A INPUT -j ACCEPT -p tcp --dport 3129 -m comment --comment "squid http proxy"
iptables -t nat -A PREROUTING -s 192.168.3.0/24 -p tcp --dport 80 -m comment --comment "transparent http proxy" -j DNAT --t
.1:3129
iptables -t nat -A PREROUTING -s 192.168.3.0/24 -p tcp --dport 443 -m comment --comment "transparent https proxy" -j DNAT -
.3.1:3129
root@squidboy:~#
```

#chmod 777 squid.sh

Note that you need to set net.ipv4.ip_forward = 1 in file /etc/sysctl.conf

In the /root directory
<your favorite editor> checksquid

Edit checksquid so it looks like the following.

```
root@squidboy:~# more checksquid
# check ip_forwarding is enabled in /etc/sysctl.conf
echo 'Make sure ip_forwarding is enabled'
sysctl net.ipv4.ip_forward
echo ""
# check the interfaces
echo 'Check the interfaces, eth0 should be 192.168.3.1'
ifconfig eth0
echo ""
# check our routing
echo 'Check iptables to see routing is setup for transparent squid'
iptables -t nat -L -n
echo ""
iptables -L -n


root@squidboy:~#
```

#chmod 777 checksquid
#./checksquid

```
pi@squidboy: ~
File  Edit  Tabs  Help
root@squidboy:~# ./checksquid
Make sure ip_forwarding is enabled
net.ipv4.ip_forward = 1

Check the interfaces, eth0 should be 192.168.3.1
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.3.1  netmask 255.255.255.0  broadcast 192.168.3.255
        inet6 fe80::d6ae:67d8:c5fa:a176  prefixlen 64  scopeid 0x20<link>
        ether dc:a6:32:c3:ee:9a  txqueuelen 1000  (Ethernet)
        RX packets 9050  bytes 1292126 (1.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 15844  bytes 16502044 (15.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


Check iptables to see routing is setup for transparent squid
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       tcp  --  192.168.3.0/24       0.0.0.0/0            tcp dpt:80 /* transparent http proxy */ to:192.168.3.1:3129
DNAT       tcp  --  192.168.3.0/24       0.0.0.0/0            tcp dpt:443 /* transparent https proxy */ to:192.168.3.1:3129

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

You can see ip_forwarding on, eth0 is 192.168.3.1 and the redirection for HTTP(80) and

HTTPS(443) to transpaprent Squid port 3129.

Now we have to set up the gateway/routing on the Raspberry PI 4 8Gb to computer on 192.168.3.0/24.
Run an Ethernet cable from eth0, 192.168.3.1 to the ethernet port on the your computer if only 1 computer or add a switch if more than one computer.

Install dnsmasq
apt install dnsmasq

Eth0 is already setup as 192.168.3.1 so no need to assign eth0 a static address as it already has one.

Change to /etc
#cp dnsmasq.conf dnsmasq.conf.old
#<your favorite editor> dnsmasq.conf
At the bottom of the file add
interface=eth0
dhcp-range=192.168.3.10,192.168.3.100,255.255.255.0,24h
Write the file

Start dnsmasq
#systemctl start dnsmasq
Don't worry if it does not start here

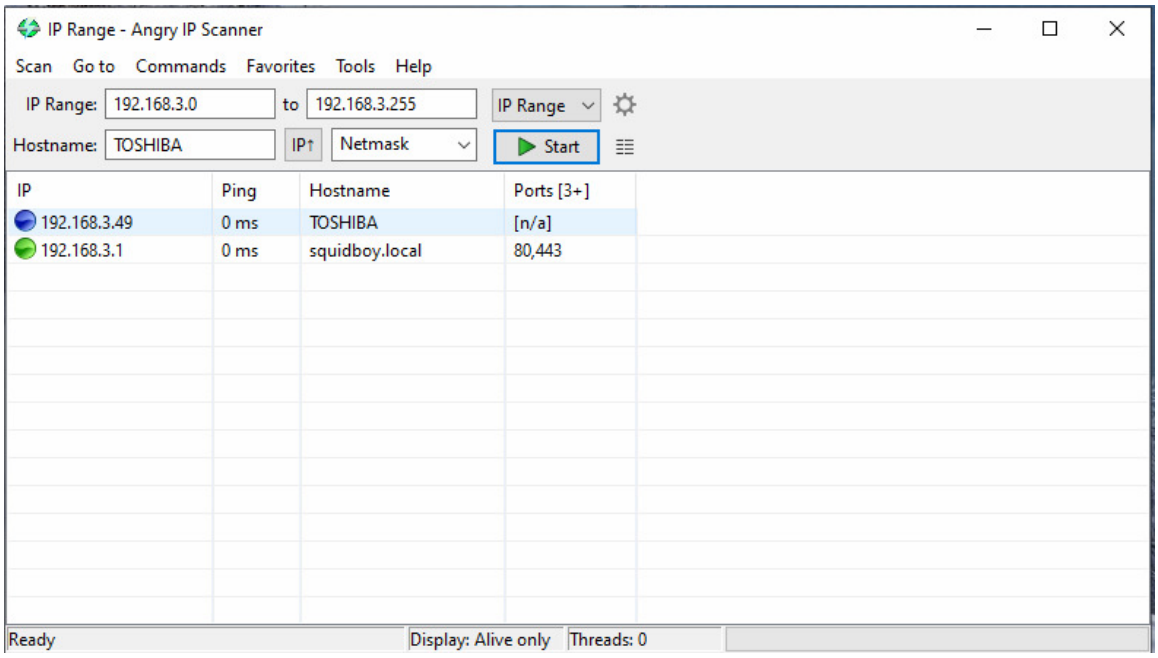
#<your favorite editor> sysctl.conf
Remove the comment from net.ipv4.ip_forward=1
Write the file

#sysctl -w net.ipv4.ip_forward=1

Add masquerade for outbound traffic on eth1
#iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
Save iptables
sh -c "iptables-save > /etc/iptables.ipv4.nat"
#iptables-restore < /etc/iptables.ipv4.nat

#reboot

When the Raspberry PI 4 8Gb reboots go to a computer on the 192.168.3.0/24 network and run Angry IP scanner. You should see the computers of the 192.168.3.0/24 network.



When I started doing the transparent Squid proxy I started over again with a new copy of the operating system. I did not install mariadb or Webmin. My squid.conf was very simple, very basic. I used port 3128 for peer mode Squid and 3129 for transparent mode Squid.

```
root@squidboy:/etc/squid#
root@squidboy:/etc/squid# more squid.conf
acl clients src 192.168.2.0/24
acl clients src 192.168.3.0/24

http_access allow localhost
http_access allow clients
http_access deny all

http_port 3128
http_port 3129 intercept

shutdown_lifetime 10 seconds
root@squidboy:/etc/squid#
```

B.3.3. Not quite there yet

Although you can use http_port 3129 for transparent Squid proxy redirection for both HTTP and HTTPS the proper way to do this is to have a separate https_port as shown in the video on transparent HTTP+HTTPS Proxy with Squid and iptables. This looks fairly involved and could even require re-compiling and re-building the Squid software.

https://youtu.be/Bogdplu_lsE

Appendix C Supplemental Information