Building an Ad-hoc network using Raspberry Pi with Raspbian and OpenWrt.

Prasoon Dhaneshwar
MT2015515
Ayush Nair
MT2015504

IIIT-Bangalore

Guided by
Sankaran Chandramouleeswaran
Professor
IIIT Bangalore

Contents

1.Introduction to OpenWRT

- OpenWrt is an embedded operating system based on the Linux kernel, primarily used on embedded devices to route network traffic. The main components are the Linux kernel, util-linux, uClibc or musl,[5] and BusyBox. All components have been optimized for size, to be small enough for fitting into the limited storage and memory available in home routers.
- OpenWrt is configured using a command-line interface (ash shell), or a web interface (LuCI). There are about 3500 optional software packages available for installation via the opkg package management system.
- OpenWrt can run on various types of devices, including CPE routers, residential gateways, smartphones, pocket computers (e.g. Ben NanoNote), and laptops. It is also possible to run OpenWrt on ordinary computers, which are most commonly based on the x86 architecture.

Uses for OpenWrt

- **Use the SSH Server for SSH Tunneling**: OpenWrt includes an SSH server so that its terminal can be accessed. For exposing the SSH server to the Internet we need to be sure to secure it with key-based authentication instead of a weak password. It can also be accessed remotely SSH tunnelling can be used to forward network's traffic over the encrypted connection. This allows secure access of websites from public Wi-Fi and access websites that can only be accessed in your home country while travelling abroad.
- **Set Up a VPN**: SSH tunneling works similarly to a VPN in many ways, but a proper VPN on your OpenWrt router can be set very easily.
- **Install a BitTorrent Client**: With some sort of network-attached storage or a router with an integrated USB port and an attached USB storage device, router itself can be used as a BitTorrent client.
- **Run Server Software**: OpenWrt's software repositories contain packages allowing it to function as a web server, IRC server, BitTorrent tracker, and more. Routers use much less power than computers, so this is a smart move if you need a lightweight server.
- **Perform Traffic-Shaping and QoS**: OpenWrt allows us to perform traffic-shaping and quality of service on the packets travelling through the router, prioritizing certain types of traffic. It can even prioritize traffic going to specific computers, de-prioritizing traffic going to other computers.
- **Create a Guest Network**: OpenWrt's wiki contains instructions for setting up a special wireless network for guests, one that's separate from your main network for security purposes. Even guest network's speed can be throttled.
- **Capture and Analyse Network Traffic:** tcpdump can be used to log all the packets travelling through the router to a network share and open the file with a tool like Wireshark to analyze the network's traffic.

So, in brief why use OpenWrt:
•Open source.
•Faster security updates.
•It makes your router better.
•No backdoors.
•Run your own VPN software.
•Additional functionality such as IPV6, WDS, RADIUS, SSH server, advanced QoS, radio output power control and so on.

Other alternatives for OpenWrt are:
Tomato and DD-WRT. While these are *open source*, manufacturers create their own custom firmware with limited hardware capabilities.

2. Building OpenWrt in Virtual Machine(VMware) and Raspberry Pi

Now we have OpenWrt firmware, but where to install it. Well, it can be either on a hardware platform or in a virtual machine. In this discussion, we'll use Raspberry Pi as our default hardware and VMware as Virtual machine.
So, generally developers build the firmware for desired router in virtual machine and then after successful build, they deploy it on the router.

In this project we have used Raspberry Pi as our platform and tried different things with OpenWrt.
So, typically installing process requires some linux tools like gunzip and qemu to convert an image file to vmdk file for VMware to read. A typical example is shown below:

gunzip openwrt-x86-generic-combined-ext4.img.gz
qemu-img convert -f raw -O vmdk openwrt-x86-generic-combined-ext4.img openwrt-x86-generic-combined-ext4.vmdk

Compilation

To compile OpenWrt following commands are used:

1. *sudo apt-get install git*
2. *git clone git://git.openwrt.org/14.07/openwrt.git*
3. *./scripts/feeds update -a*
4. *./scripts/feeds install -a*
5. *./scripts/feeds update -i*
6. *make clean*
7. *make dirclean*
8. *make distclean*
9. *make menuconfig*
10. *make -j 3*

We've picked up a tutorial which clearly explains the compilation process. Same above steps are followed.

Making a build for particular hardware

After step 9 a configuration page opens up, shown in figure 1.

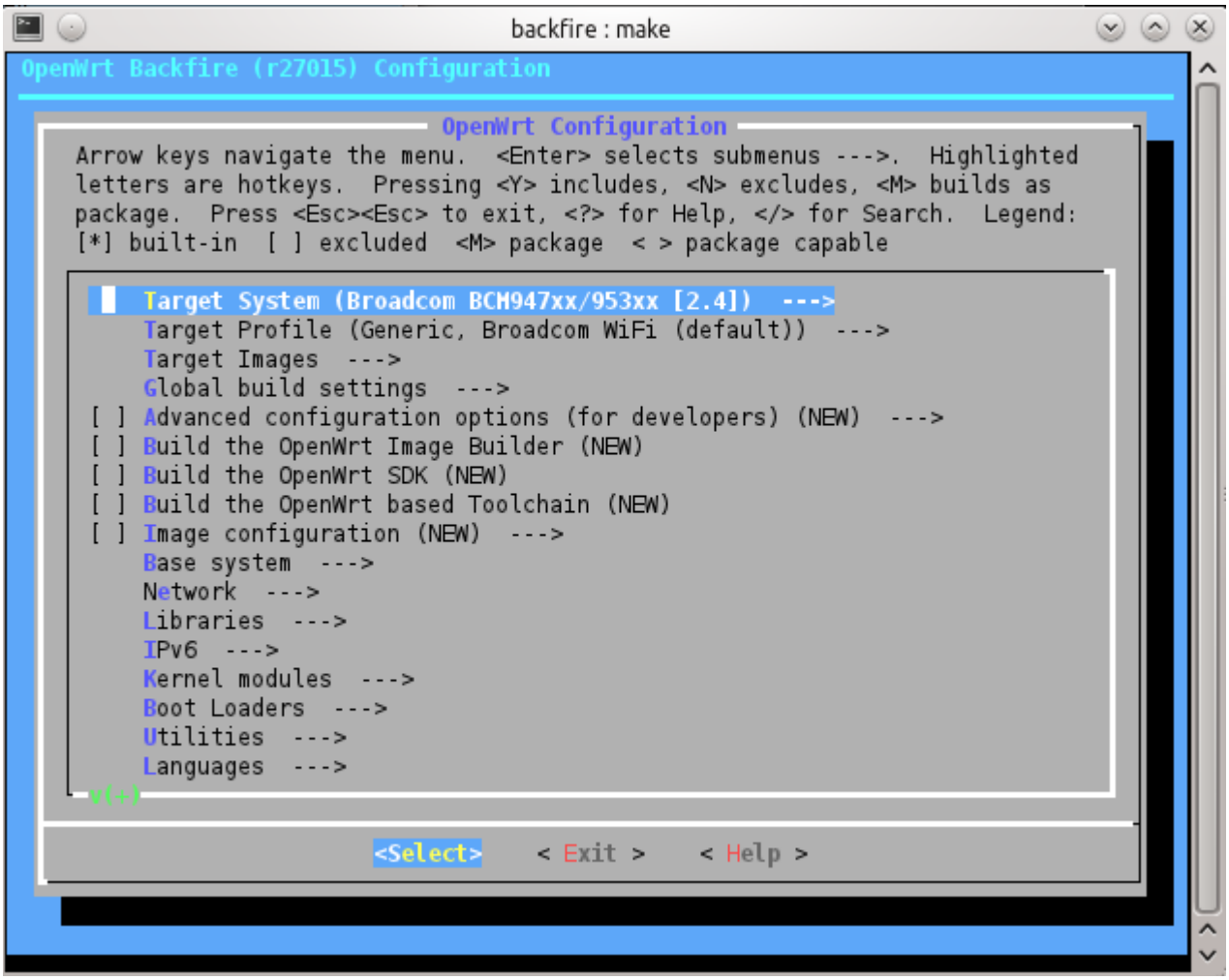


figure 1. Building firmware for particular hardware.

Selecting *target system* is the main process, the architecture of the chipset is defined here. Other important things to look for are the *Target profile, Image configuration, Base system, Libraries and Utilities* as per user requirement.

In this project, we were supposed to work with **Ubiquiti dual band routers(particularly UAP-PRO and UAP-AC)**, which uses MIPS architecture. The hardware specs are shown in figure 2.

Hardware

Version	LITE	LR	PRO	EDU
W Instruction set:	W MIPS	W MIPS	W MIPS	?
Vendor:	Qualcomm / Atheros	Qualcomm / Atheros	Qualcomm / Atheros	?
Bootloader:	UBoot	UBoot	UBoot	?
System-On-Chip:	QCA9563-AL3A	?	QCA9563-AL3A	?
CPU Frq:	775 MHz	775 MHz	775 MHz	?
BogoMIPS:	385.84	385.84	385.84	?
Flash-Chip:	Macronix MX25L12835FMI	?	Macronix MX25L12835FMI	?
Flash size:	16 MiB	?	16 MiB	?
RAM-Chip:	Winbond W971GG6KB-25	?	Winbond W971GG6KB-25	?
RAM size:	128 MiB	128 MiB	128 MiB	?
Wireless 1 (2.4 GHz):	QCA9563 built-in	?	QCA9563 built-in	?
Wireless 2 (5 GHz):	Ubiquity U-AME-G1-BR4A	?	QCA9880-BR4A	
Eth Phy:	Atheros AR8033-AL1A	?	QCA8334-AL3C	?
USB:	possibly unpopulated	?	USB 2.0	?
Serial:	Yes, for U-Boot	Same as the AC Lite	?	?
JTAG:	N/A	?	?	?

figure 2. Hardware configuration of some of dual band Ubiquiti routers

3.Raspberry Pi: an open source limitless hardware

- The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

- What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work.

Make Raspberry Pi a WiFi Router

The Raspberry Pi is single-board computer without an Ethernet Network Switch. OpenWrt for Raspberry Pi 2 is compiled with the ARM hard-float kernel ABI (armhf). This means that non-integer math is done in hardware instead of in software. Raspberry Pi 1 (and Zero) support only soft-float (armel) which is slower than hard float. The Raspberry Pi is supported in the brcm2708 target. Here, we are using a *WiFi dongle* with Raspberry Pi 2 to make it as a wireless router/Access point.

Flash OpenWrt to an SD card

On a Linux desktop, insert your SD card and run: dmesg (to see the latest kernel messages)
At the time of this project the latest releases of OpenWrt were:
openwrt-brcm2708-bcm2708-sdcard-vfat-ext4.img
openwrt-brcm2708-bcm2709-sdcard-vfat-ext4.img
To copy the image to sd card:
dd if=/home/username/Downloads/openwrt-brcm2708-bcm2709-sdcard-vfat-ext4.img of=/dev/sdX bs=2M conv=fsync

4.Communications between VM or Pi as a client to server

Two types of connections are possible here: SSH and LuCI.
Telnet and ftp can also be used, but above two methods are more secure.

- **SSH:**

  Secure Socket Shell, is a network protocol that provides administrators with a secure way to access a remote computer. Secure Shell provides strong authentication and secure encrypted data communications between two computers connecting over an insecure network such as the Internet.
  SSH is widely used by network administrators for managing systems and applications remotely, allowing them to log in to another computer over a network, execute commands and move files from one computer to another.

- **LuCI:**

  LuCI is the main web configuration interface for OpenWrt. It is an easy way to edit the device's configuration files. The interface looks as shown below in figure 3. To access the interface, put router's assigned address in the browser(ex: 192.168.1.1).

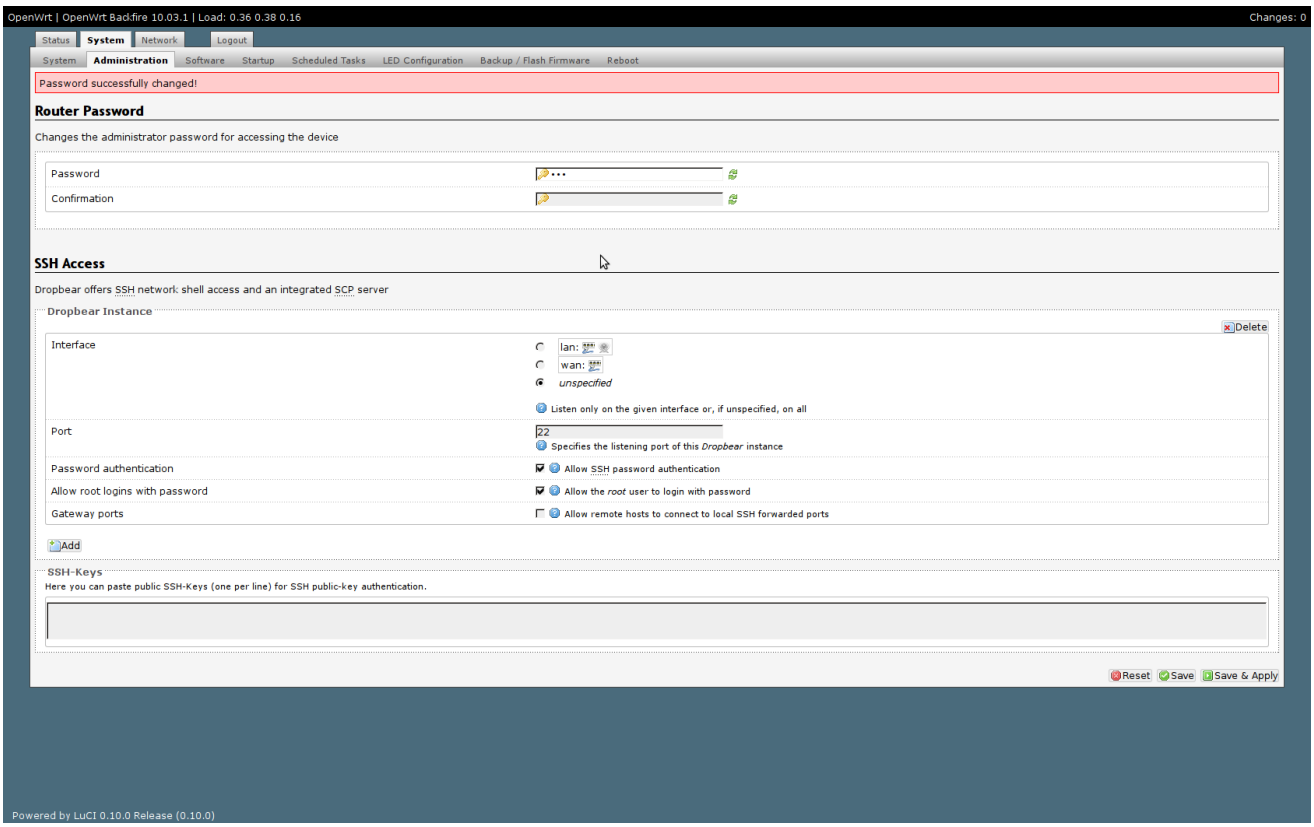


figure 3. Administration interface of LuCI web interface. It's a powerful tool to see all the statistics of the network, all changes that can be done to router externally

Connecting to internet

1. VMware as Virtual machine: To directly boot into the system, we use qemu to convert to vmdk file, ex:
qemu-img convert -f raw openwrt-x86-generic-combined-ext4.img -O vmdk openwrt-x86-generic-combined-ext4.vmdk

After setting it to work in VM, interface of system can be seen. (figure 4)

figure 4. OpenWrt interface.

we use eth0 as default to connect to internet:
udhcpc eth0
vim etc/config/network (to configure the network and set it to dhcp shown in figure 5)

figure 5. configure network as DHCP

```
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config interface 'lan'
        option ifname 'eth0'

        option proto 'dhcp'
        option hostname 'barrier'

config interface 'wan6'
        option ifname '@wan'
        option proto 'dhcpv6'

config globals 'globals'
        option ula_prefix 'fdfc:5100:e0ad::/48'
```

To check the network interfaces, *ifconfig* (figure 6)can be used which tells the address assigned either manually or by dhcp server.
Now to check internet is connected to system, *ping* command is used (figure 7).

```
root@OpenWrt:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:EA:D1:B2
          inet addr:172.16.80.30  Bcast:172.16.83.255  Mask:255.255.252.0
          inet6 addr: fe80::20c:29ff:feea:d1b2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:91293 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6228 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20271919 (19.3 MiB)  TX bytes:586076 (572.3 KiB)
          Interrupt:18 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1560 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1560 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:244001 (238.2 KiB)  TX bytes:244001 (238.2 KiB)

root@OpenWrt:/# _
```

figure 6. network configurations

```
root@OpenWrt:/# ping google.com
PING google.com (216.58.220.46): 56 data bytes
64 bytes from 216.58.220.46: seq=0 ttl=55 time=86.935 ms
64 bytes from 216.58.220.46: seq=1 ttl=55 time=45.816 ms
64 bytes from 216.58.220.46: seq=2 ttl=55 time=51.412 ms
64 bytes from 216.58.220.46: seq=3 ttl=55 time=76.676 ms
64 bytes from 216.58.220.46: seq=4 ttl=55 time=78.973 ms
64 bytes from 216.58.220.46: seq=5 ttl=55 time=48.476 ms
64 bytes from 216.58.220.46: seq=6 ttl=55 time=51.981 ms
64 bytes from 216.58.220.46: seq=7 ttl=55 time=33.812 ms
64 bytes from 216.58.220.46: seq=8 ttl=55 time=53.164 ms
64 bytes from 216.58.220.46: seq=9 ttl=55 time=95.601 ms
_
```

figure 7. pinging google.com, thus confirming internet connection

Connecting to internet

2. Raspberry Pi: As we have flashed OpenWrt into SD card, its time to boot it up. During first time we need to set it for SSH or FTP connection for communication using *passwd*.

- After this Pi can be "SSHed" to server.
- The procedure is same as VMware discussed above.

```
root@(none):/# passwd
Changing password for root
New password:
Retype password:
Password for root changed by root
root@(none):/#
```

figure 8. changing the passwrord during first time login

Connecting the Wi-Fi dongle and its necessary drivers:
In OpenWrt, all the packages can be updated by *opkg update.*
In Raspbian, it is done via sudo apt-get update.

For OpenWrt, the procedure for connecting to internet is same as of VMware. But in Raspbian system, we need to do *sudo nano /etc/network/interfaces* as shown below:

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0


iface wlan0 inet dhcp
        wpa-ssid "ssid"
        wpa-psk "password"
```

figure 9. before and after changing the network interfaces.

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

auto wlan0
allow-hotplug wlan0
iface wlan0 inet dhcp
   wpa-scan-ssid 1
   wpa-ap-scan 1
   wpa-key-mgmt WPA-PSK
   wpa-proto RSN WPA
   wpa-pairwise CCMP TKIP
   wpa-group CCMP TKIP
   wpa-ssid "My Secret SSID"
   wpa-psk "My SSID PSK"

iface default inet dhcp
```

Thus, when pinged, Pi connects to internet(figure 7)

5.Forming Ad-hoc networks

Wireless ad hoc network

- A wireless ad hoc network (**WANET**) is a decentralized type of wireless network.
- The network is ad hoc because it does not rely on a pre existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks.
- Each node participates in routing by forwarding data for other nodes, so the determination of which nodes forward data is made dynamically on the basis of network connectivity.
- In addition to the classic routing, ad hoc networks can use flooding for forwarding data.

Making ad-hoc network in Raspberry Pi

Debian Method: In this Debian method, two wireless LAN clients, namely node A and node B as shown in figure 10 will be configured as ad-hoc network nodes with static IP addressing. Open in each Pi, with following command: *sudo vi /etc/network/interfaces*

Node A

```
auto wlan0
iface wlan0 inet static
    address 192.168.1.1
    netmask 255.255.255.0
    wireless-channel 1
    wireless-essid MYNETWORK
    wireless-mode ad-hoc
```

Node B

```
auto wlan0
iface wlan0 inet static
    address 192.168.1.2
    netmask 255.255.255.0
    wireless-channel 1
    wireless-essid MYNETWORK
    wireless-mode ad-hoc
```

figure 10. configuring the nodes in Pi as node A and B and assigning same network as MYNETWORK in ad-hoc mode.

After raising the network, using *ifup wlan0,* we can test for ping thus connecting each other, shown in figure 11.

```
pi@raspberrypi:~ $ ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.863 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.768 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.769 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.776 ms
```

```
pi@raspberrypi:~ $ 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=0.933 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.696 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.671 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=0.808 ms
```

figure 11. pinging from node A to node B. Similarly, ping can also be done from node B to node A.

- This proves the ad-hoc network connection. *ping* and *traceroute* (figure 12.) methods are most useful for measuring transit delays of packets across an Internet Protocol.

```
pi@raspberrypi:~ $ traceroute 192.168.1.2
traceroute to 192.168.1.2 (192.168.1.2), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  1.168 ms  1.140 ms  1.473 ms
```

```
pi@raspberrypi:~ $ traceroute 192.168.1.2
traceroute to 192.168.1.2 (192.168.1.2), 30 hops max, 60 byte packets
 1  raspberrypi (192.168.1.2)  0.178 ms  0.098 ms  0.066 ms
```

figure 12. traceroute, recording the route of nodes which are connected to it.

6.Using http and rtsp protocols for transmitting data in real time

http or **Hypertext Transfer Protocol (HTTP)** is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

- HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the client and an application running on a computer hosting a web site may be the server.
- The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client.
- The response contains completion status information about the request and may also contain requested content in its message body.

Here, in our scenario of transmitting data, the completion status of response takes time, so the delay becomes too high when streaming data or receiving camera feeds for example.

rtsp or **Real Time Streaming Protocol** is a network control protocol designed for use in entertainment and communications systems to control streaming media servers.

- The protocol is used for establishing and controlling media sessions between end points.
- Clients of media servers issue VCR-style commands, such as play, record and pause, to facilitate real-time control of the media streaming from the server to a client (Video On Demand) or from a client to the server (Voice Recording).

Benchmarks and tests

- In this project, we used Raspbian operating system with vncserver.
- Using scp to transfer files or any media.

Ex:

scp/sourcelocation/source.file pi@destaddr:/home/destlocation/dest.file(optional)

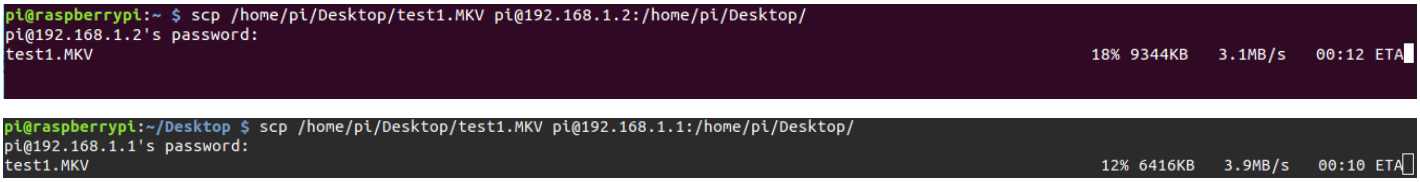


figure 13. using scp in both nodes. Observe the transfer speeds, thus proving the network's real time capabilities.

Streaming media and camera feeds

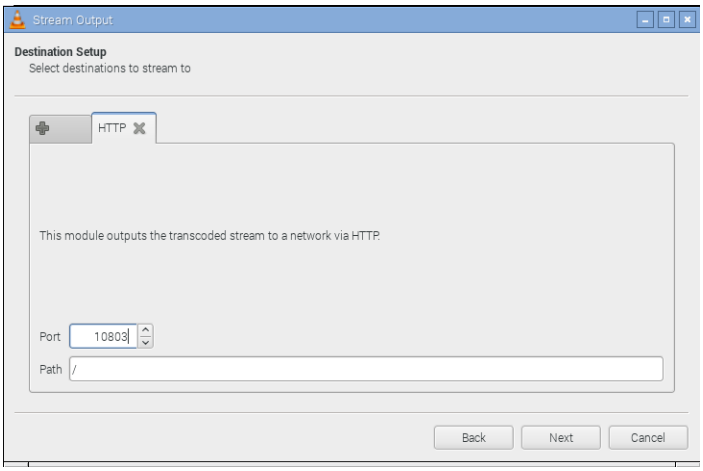
- VLC media player to stream between two nodes: one was 192.168.1.1 as node A and 192.168.1.2 as node B.
- **Video stream:** For http the default ports is 8080, but here we've allocated port as 10803(figure 14.)
- For rtsp 8554 is the default port, but we've used a range from 8100 to 8500.(figure 16.)

As shown in figure 15, the speeds of transfer are very slow, probably because of:

- observing through **vncserver**, a GUI to access pi, which takes significant bandwidth
- congested ports maybe one reason, but can be improved by changing it.

Camera Feeds:

- Same as video stream, camera feeds are laggy for suspected same reasons above.(figure 17.)



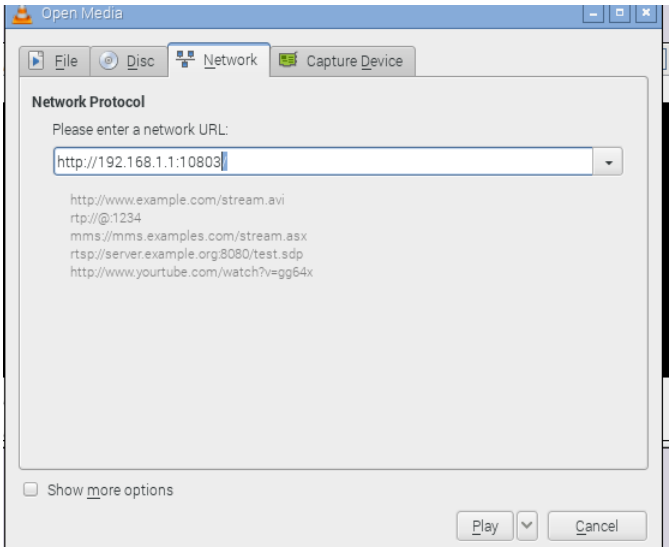
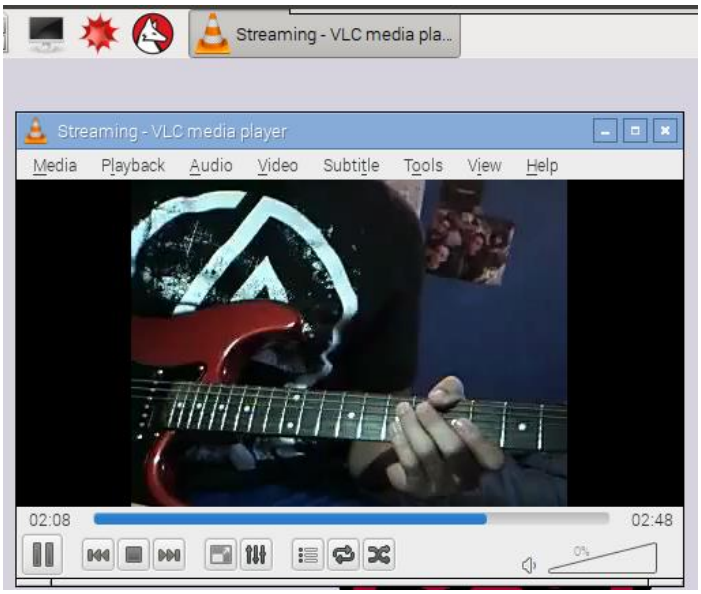


figure 14. assigning http ports as 10803 in the left and obtaining the stream at right



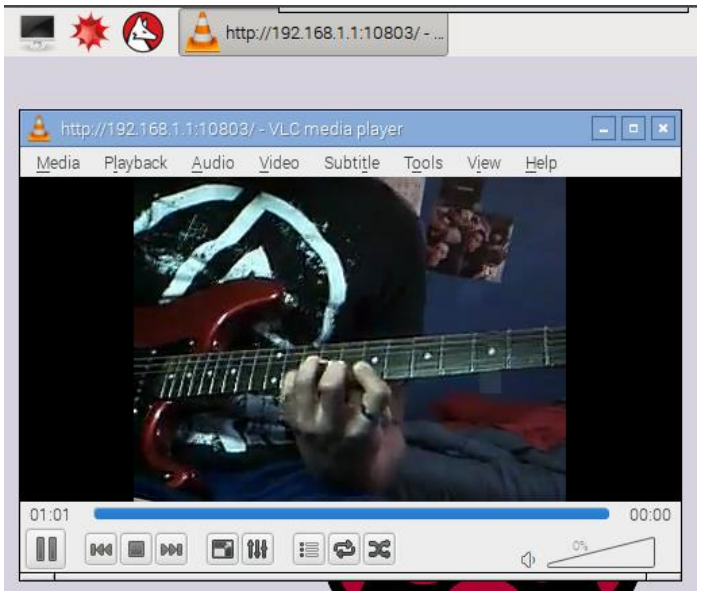
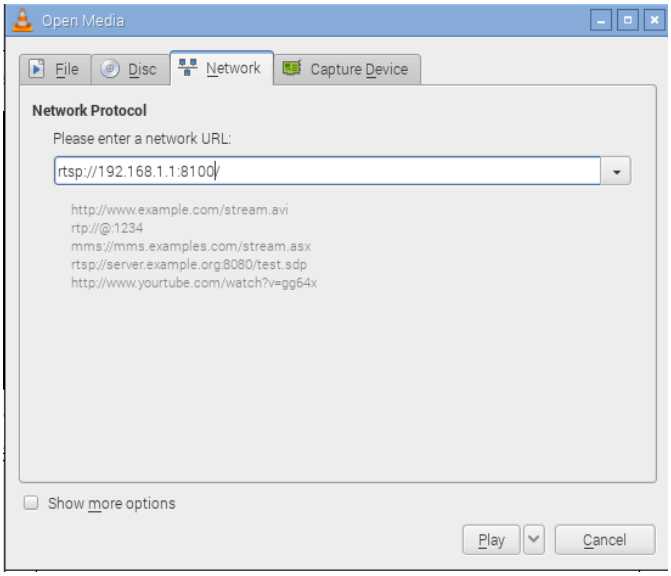


figure 15. Notice the time stamps in both streams, this difference is because of handshaking between http and the bandwidth consumption of vncserver.



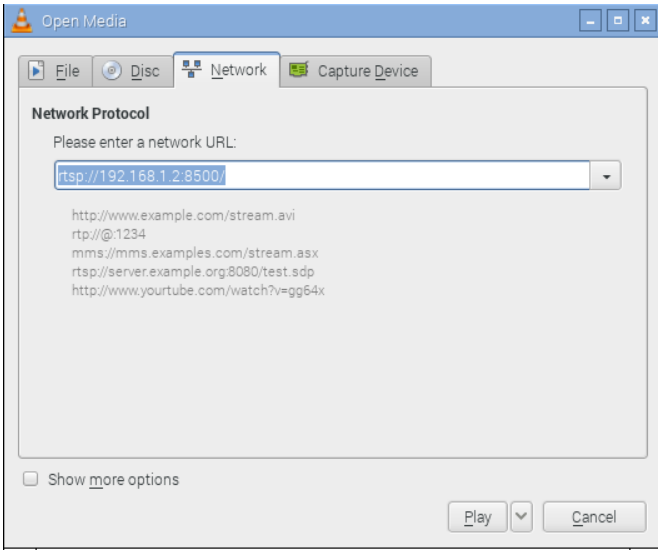
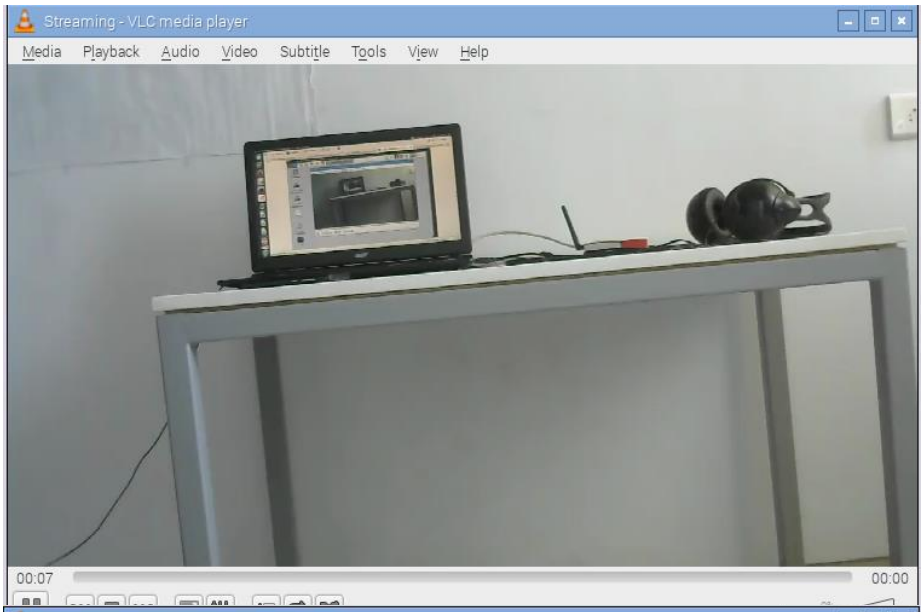


figure 16. assigning http ports as 10803 in the left and obtaining the stream at right



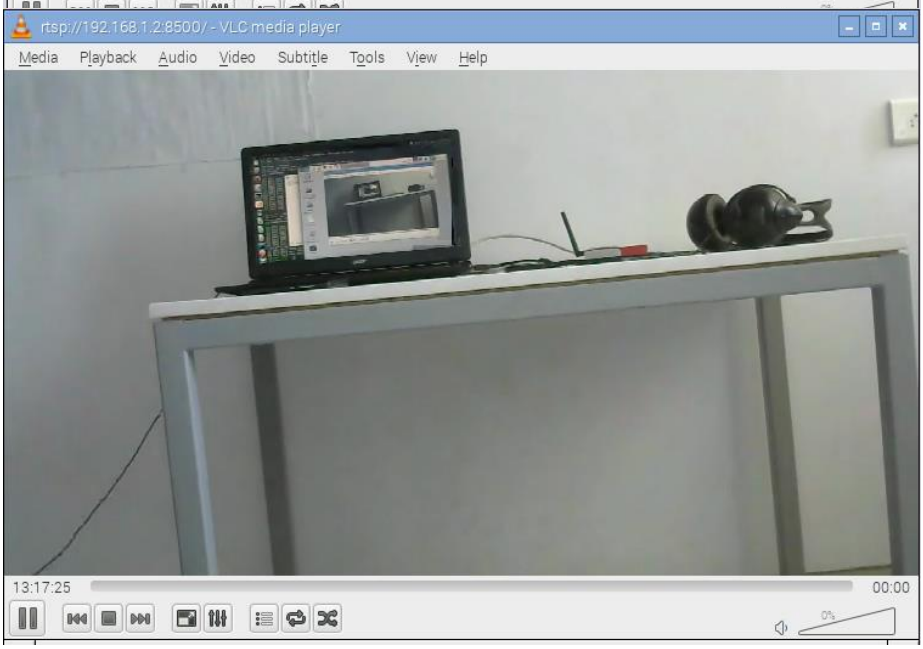


figure 17.
camera streaming in real time. Notice the monitor of Laptop; the change can be seen. Both frames are captured at the exact time instant. This delay is however lesser than http protocol. About 4-5 seconds of delay are observed in this process.

7. Applications

- A mobile ad hoc network (MANET) is a continuously self-configuring, infrastructure-less network of mobile devices connected without wires.

- Vehicular ad hoc networks (VANETs) are used for communication between vehicles and roadside equipment. Intelligent vehicular ad hoc networks (InVANETs) are a kind of artificial intelligence that helps vehicles to behave in intelligent manners during vehicle-to-vehicle collisions, accidents.

- Smartphone ad hoc networks (SPANs) leverage the existing hardware (primarily Bluetooth and Wi-Fi) in commercially available smartphones to create peer-to-peer networks without relying on cellular carrier networks, wireless access points, or traditional network infrastructure.

- Internet-based mobile ad hoc networks (iMANETs) are ad hoc networks that link mobile nodes and fixed Internet-gateway nodes. One implementation of this is Persistent System's CloudRelay.

- Military/Tactical MANETs are used by military units with emphasis on security, range, and integration with existing systems

8. Challenges and future work

- In this project, ad-hoc network was wireless. Thus, signal interference and strength should be taken into account.
- Real time protocols work over shell remarkably well, but not so much in any software generated graphical interface. One, possible solution is to observe through monitor, rather than software imitating the process, thus reducing the delay in feeds and stream.

- **Future work** will be doing the same project in Ubiquiti line of routers, and also transmitting internet data packets through the network.

9. References

- Introduction to OpenWrt: https://en.wikipedia.org/wiki/OpenWrt
- Need for OpenWrt: http://www.makeuseof.com/tag/what-is-openwrt-and-why-should-i-use-it-for-my-router/
- https://www.reddit.com/r/linux/comments/3knsd9/eli5_why_is_openwrt_better_than_my_regular_home/
- OpenWrt on VMware: https://wiki.openwrt.org/doc/howto/vmware
- Building a custom OpenWrt: https://www.pacificsimplicity.ca/blog/routerstation-pro-openwrt-image-howto
- OpenWrt on Ubiquiti routers: https://wiki.openwrt.org/toh/ubiquiti/unifiac
- Installing OpenWrt on Raspberry Pi: https://wiki.openwrt.org/toh/raspberry_pi_foundation/raspberry_pi
- SSH: http://searchsecurity.techtarget.com/definition/Secure-Shell
- Configuring OpenWrt first time: https://wiki.openwrt.org/doc/howto/firstlogin
- Wi-Fi dongle setup for Pi: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-3-network-setup/setting-up-wifi-with-occidentalis
- Ad-Hoc Network: https://wiki.debian.org/WiFi/AdHoc
- Secure copy: http://www.hypexr.org/linux_scp_help.php
- TightVNC Server: http://www.tightvnc.com/vncserver.1.php