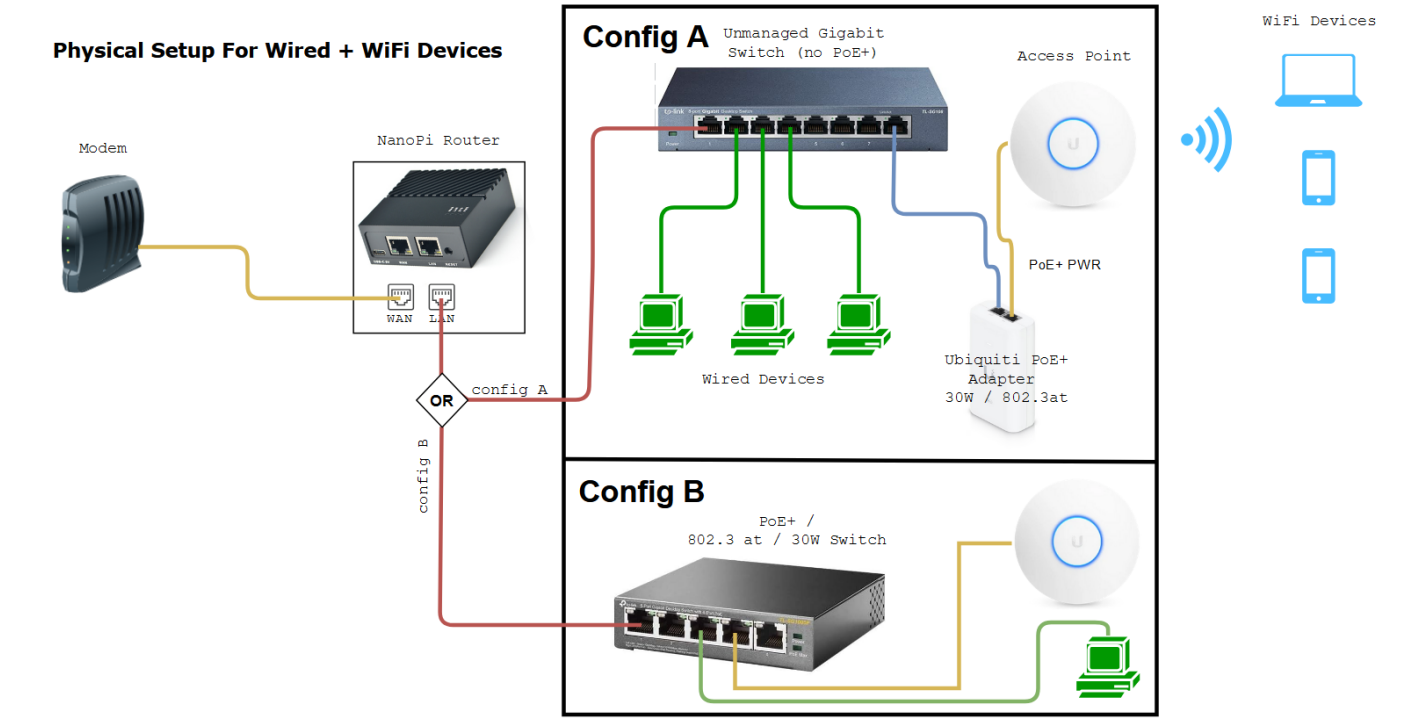
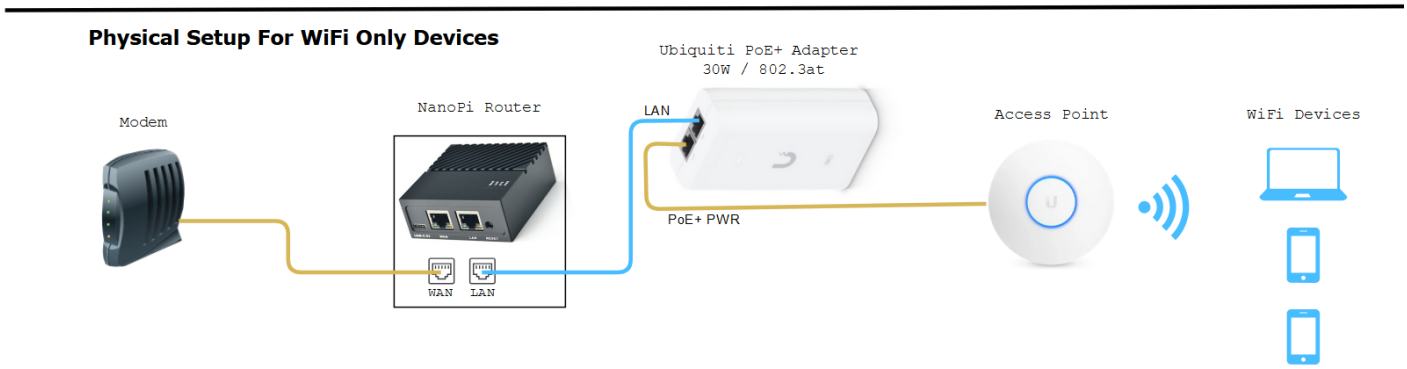
NanoPi R6S R4S for Gigabit SQM With OpenWrt

I made a new significant discovery on 2024.01.07 that makes the NanoPi R6S capable of pushing past 1400+ Mbps with cake on!!! Hooray! See here. On 2024.01.11 I discovered the same for the NanoPi R4S which is now able to push up to 800 Mbps with cake SQM.





Pictured Gigabit Switch: TP-Link 8-Port Gigabit Ethernet Switch (**Amazon Referral Link)**

Pictured Access Point - Ubiquiti Unifi 6 Pro (**Official Link)** Note: If Ubiquiti is out of stock you or if you don't like them, I heard that the TP-Link EAP670 **(Amazon Referral Link)** from their Omada lines work just as great. I just have never tried TP-Link EAPs myself as I've been using Ubiquiti APs.

Note: The Unifi 6's don't include power. So you need to buy the (30W aka 802.3at or PoE+) Injector to power it as shown in the picture. Alternative you can get a switch that supports PoE+, 802.3at, 30W).

Pictured OpenWrt Device: NanoPi R4SE **(Official Link).** As for the power supply I'd recommend the CanaKit Raspberry Pi power supply **(Amazon Referral Link)** over the one FriendlyElec provides. The SE version has eMMC which means you can run off the devices storage instead of running on the MicroSD like it's predecessor the R4S. Cake SQM limit is at about **630Mbps**

The new one is the NanoPi R6S **(Official Link)** which has a better CPU, the RK3588S. You may need to source your own 18W PD and USB-C to USB-C PD Cable. It has 2x 2.5Gbps ports and 1 Gbps port. The alternative, NanoPi R6C **(Official Link)** costs less and has the same CPU however it only has 1x 2.5 Gbps port. Both can do cake SQM at **1400+Mbps**.

Note: The NanoPis from Amazon are not from official sources. FriendlyElec would be direct from manufacturer.

> **R4SE - Can do cake SQM up to 800 Mbps (w/ performance tweak).** Also has official OpenWrt software available.
>
> **R6S - Can do cake SQM up to 1500 Mbps (w/ performance tweak)**. Only has official FriendlyWrt software for now.

If you're interested in using cake SQM on an x86 machine please refer to this page instead: https://wiki.stoplagging.com/books/technical-guides/page/sqm-for-beyond-1-gbps-lines-with-openwrt

1.1 Introduction and Why?

The diagram above demonstrates how you would install a more powerful ARM PC, the nanoPi as a router into your network. Building your home network infrastructure like this, is more reliable and better than consumer routers which try to put the modem, routing, and wireless all in one.

The reason why we would want to do this is so we can stop bufferbloat at higher bandwidths with SQM (Smart Queue Management) turned on. Currently consumer routers usually can't push past 350 Mbps with cake or fq_codel SQM they are limited by their CPU power. Most consumer routers have underpowered CPUs so that's why the NanoPis are a solid choice. They are low power usage,

small and have solid CPU that can handle cake at **800 Mbps** (R4SE) and **1400+ Mbps** (R6S w/ CPU fixes)

What is Bufferbloat and why stop it?

It is lag or ping spikes in video games or zoom calls that is caused when you or someone else uses up all your bandwidth. It could be torrenting, 4k streaming, bulk downloads, or even a speedtest. SQM algorithms (fq_codel or cake) which are available on OpenWrt, can completely mitigate these pings and ensures low latency even under full load. Overall, you do sacrifice a little max speed 5-10% for guaranteed low latencies.

NanoPi R4S / R4SE Performance

By default, the FriendlyWrt firmware on R4S doesn't optimally single out the use of it's faster A72 cores for Queues. This causes the cap to be around **630 Mbps**.

Do the performance tweak here and the improvement for cake SQM jumps from **630 Mbps** up to **800 Mbps** afterwards.

Note: Don't copy me here and set max bandwidth as 920000 Mbps I was just testing the limits. You should be setting 90-95% of you max bandwidth for best performance! The video is just to demonstrate it's possible to do up to 800 Mbps cake.

NanoPi R6S Performance

By default, the FriendlyWrt firmware on R6S doesn't optimally single out the use of it's faster A76 cores for Queues. This causes the cap to be around **800 Mbps**.

Do the performance tweak here and the improvement for cake SQM jumps from **800 Mbps** to **1400+ Mbps** afterwards.

NanoPi Software Installation

Installation is easy. You just need to flash a microSD card with friendlyWrt. They have a tutorial here for **R4SE**: https://wiki.friendlyelec.com/wiki/index.php/NanoPi_R4S#Install_OS

And a tutorial here for **R6S**:
https://wiki.friendlyelec.com/wiki/index.php/NanoPi_R6S#Install_OS_to_eMMC

All you have to do to install is...
1. Plug in a microSD card to your computer.

2. Download the appropriate image (usually the eflasher) from the FriendlyWrt wiki
3. Get win32diskimager and launch it.
4. On win32diskimager select your image file that you downloaded and select your microSD drive letter. Then flash!
5. After flashing is done eject microSD and unplug.
6. Plug in microSD into your NanoPi and wait for it to flash (LEDs pictured below for reference)

4.5.3.1 Option 1: Install OS via TF Card

This method firstly boots a mini Linux from a TF card and then automatically runs an EFlasher utility to install the OS to eMMC. You can connect your system to an HDMI monitor and watch its progress.
This is optional. You can watch its progress by observing its LEDs as well:

Progress	SYS LED(Red)	LAN LED(Green)	WAN LED(Green)
Power On	Solid On	Off	Off
System Boot	Slow Flashing	Off	Off
Installation in Progress	Fast Flashing	Off	Off
Installation Done	Slow Flashing	Solid On	Solid On

By default, flashing starts automatically upon power-up, so be sure to back up the data in eMMC. If you don't want it to start automatically, you can use image file with a filename containing the words 'multiple-os' and manually select the OS you want to flash on the interface.

7. Hook up WAN to your modem. Hook up LAN to either your switch which connects to a computer or hook up LAN directly to your computer.
8. Power on. Wait about 3 minutes.
9. On the computer that is connected to the switch or NanoPi's LAN port. Go to web browser and enter in http://192.168.2.1 to access your router

That's it! All that is left is to configure SQM with fq_codel as shown . There's no need to install luci-app-sqm because the FriendlyWrt image has everything already! You just need to enable SQM via the official openWrt guide or my guide.

Either way feel free to improve it further with the advanced cake config section of this page

If you want to fine tune cake further you can see the section below this page:
https://wiki.stoplagging.com/books/technical-guides/page/sqm-for-up-to-800-mbps-lines-with-openwrt#bkmrk-1.4-advanced-cake-co

1.3 What Access Point to Get?

I keep hearing raving reviews about the Ubiquiti APs and use one myself. I have extremely stable WiFi with these and never have to reboot them. Ubiquiti also advertises up to 200 concurrent users as well! If you have a recommendation better than these I'd like to know.

Ubiquiti Unifi 6 Pro (**Official Link)**

If you plan on only having one Ubiquti AP I recommend installing via the phone so you don't have to bother with more complicated things like AP Controllers.

If you're on a budget and can't buy a dedicated AP. You **can try turning your old router** into an **access point** by putting it into AP mode instead of routing mode. This is important because you should be letting the OpenWrt device do the routing to prevent bufferbloat not your old router.

Another option you could try that I've heard are good are the TP-Link EAP670 **(Amazon Referral Link)**. I have no real world experience with these as I don't own any, but I heard they are solid products in the /r/homenetworking community.

Facts about WiFi

If you need more coverage you should get more APs not one single AP with a bunch of antennas, because those are marketing gimmicks.

WiFi has limited range due to the physics of their frequency bands.

5Ghz can handle more bandwidth, but will usually be about half the range of 2.4Ghz.

1.4 Advanced Cake Configuration

This section is for my own reference and these were recommended by the official docs:
https://openwrt.org/docs/guide-user/network/traffic-shaping/sqm-details#sqmqueue_discipline_tab

It's not necessary to do this but if you want even further ping stability under load it might be worthwhile!

Under the Queue Discipline tab of SQM.
Enable the checkmark for advanced configuration and save& apply.

This turns on squash_dscp, squash_ingress, ECN on ingress and NOECN on egress. Leave them as defaults as they are good the way they are. (If you have symmetrical fiber then ECN can be enabled on egress.

Next checkmark and enable "Dangerous Configuration" which is below the "Advanced Configuration" section. We are going to disable triple-isolate and enable per host isolation... Here's a short explanation.

To quote the docs, by default, cake will use triple-isolate: “which will first make sure that no internal or internal host will hog too much bandwidth and then will still guarantee for fairness for each host. In that mode, Cake mostly does the right thing. It would ensure that no single stream and no single host could hog all the capacity of the WAN link. However, it can’t prevent a BitTorrent client – with multiple connections – from monopolizing most of the capacity.” You can enable per host isolation, which will identify all source/destination information.

To enable that, Add the following to the “Advanced option strings” (in the Interfaces → SQM-QoS page; Queue Discipline tab, look for the Dangerous Configuration options):

For queueing disciplines handling incoming packets from the internet (internet-ingress): nat dual-dsthost ingress

For queueing disciplines handling outgoing packets to the internet (internet-egress): nat dual-srchost

For me that means Qdisc options (ingress) I wrote in "nat dual-dsthost ingress" while for Qdisc options (egress) I wrote in "nat dual-srchost"

1.5 Performance Tweaks for R4S and R6S

- https://github.com/StarWhiz/NanoPi-R6S-CPU-Optimization-for-Gigabit-SQM/tree/main for NanoPi R6S
  - Performance improvement for cake SQM jumps from 800 Mbps to 1400+ Mbps afterwards
- https://github.com/StarWhiz/NanoPi-R6S-CPU-Optimization-for-Gigabit-SQM/tree/main/R4S%20CPU%20Optimization for NanoPi R4S
  - Performance improvement for cake SQM jumps from 630 Mbps up to 800 Mbps afterwards.

Deprecated way to fix this

```
## Step1: Get CPU Frequencies to confirm that cores 4, 5, 6, and 7 are the faster cores. CPU0 starts from the top.
cat /sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_max_freq

## Step2: Get IRQ ##s (In my example below they are 31 for eth0 and 88 for eth1 yours may be different)
grep eth /proc/interrupts

## My output of Step2 yours may be different
grep eth /proc/interrupts
 74:          0          0          0          0          0          0          0
0     GICv3 266 Level     eth0
 75:          0          0          0          0          0          0          0
0     GICv3 265 Level     eth0
128:          0          0      61981          0          0          0          0
2335426   ITS-MSI 570949632 Edge      eth1-0
144:          0          0    1236903          0          0          0          0
0   ITS-MSI 570949648 Edge      eth1-16
146:          0          0          0          0          0          0          0
0   ITS-MSI 570949650 Edge      eth1-18
149:          0          0          0          0          5          0          0
3   ITS-MSI 570949653 Edge      eth1-21
160:          0          0          0     148716    4732058          0          0
0   ITS-MSI 428343296 Edge      eth2-0
176:          0          0          0    1559148          0          0          0
0   ITS-MSI 428343312 Edge      eth2-16
178:          0          0          0          0          0          0          0
0   ITS-MSI 428343314 Edge      eth2-18
181:          0          0          0          0          0          0          0
7   ITS-MSI 428343317 Edge      eth2-21

## What is your IRQ ##?
By default eth2-0 is the WAN port and eth1-0 is the 2.5gbps LAN port on the R6S. So the lines of interest are below:
160:          0          0          0     148716    4732058          0          0
0   ITS-MSI 428343296 Edge      eth2-0
128:          0          0      61981          0          0          0          0
2335426   ITS-MSI 570949632 Edge      eth1-0

Your IRQ number might be different from mine whcih is 160 for 2.5 Gbps WAN and 128 for 2.5Gbps
```

```
LAN

## Optional Step: List CPU Cores Assigned to Current IRQs
cat /proc/irq/160/smp_affinity
cat /proc/irq/128/smp_affinity

## Optional Step: List CPU Cores Assigned Current Queues
cat /sys/class/net/eth0/queues/rx-0/rps_cpus
cat /sys/class/net/eth1/queues/rx-0/rps_cpus

The optional steps above are to see what the values are currently. Nowe we will change them!

## Step3: The Performance Tweaks. The idea here is to put IRQ cpu affinities on Faster A76
Cores.
And assign all CPU cores to the queues.

#ETH0 irq on core 4,5 (a76 core) replace 160 with your actual IRQ number for WAN
echo -n 30 > /proc/irq/160/smp_affinity

#ETH1 irq on core 6,7 (a76 core) replace 128 with your actual IRQ number for WAN
echo -n c0 > /proc/irq/128/smp_affinity

#ETH0 queues on all CPU cores
echo -n ff > /sys/class/net/eth0/queues/rx-0/rps_cpus

#ETH1 queues on all CPU cores
echo -n ff > /sys/class/net/eth1/queues/rx-0/rps_cpus

#ETH2 queues on all CPU cores
echo -n ff > /sys/class/net/eth1/queues/rx-0/rps_cpus
```

> If you restart Smart Queue Management or change SQM settings, it will **reset** the CPU affinity and you will **need** to re-apply the performance tweaks again. I recommend doing it the new way

Optional understanding of CPU Affinity

This section below is **optional** and is for my understanding of how the hex value selects certain CPU cores.

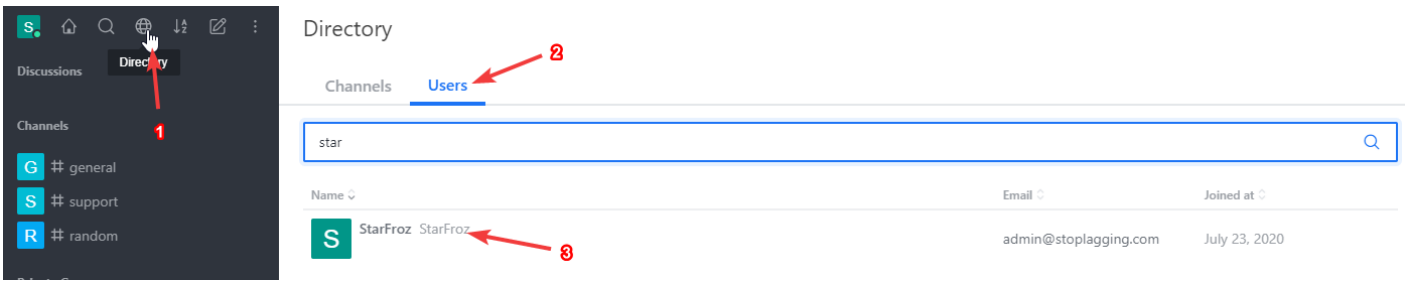
```
## Performance Tweaks Quick Reference R6S
Binary   = hex ## = cpu core
00000001 = hex 1 = cpu core 0 (A55) selected
00000010 = hex 2 = cpu core 1 (A55) selected
00000100 = hex 4 = cpu core 2 (A55) selected
00001000 = hex 8 = cpu core 3 (A55) selected
00010000 = hex 10 = cpu core 4 (A76) selected
00100000 = hex 20 = cpu core 5 (A76) selected
01000000 = hex 40 = cpu core 6 (A76) selected
10000000 = hex 80 = cpu core 7 (A76) selected

## Examples (Note CPU0 starts on the right. CPU7 ends on the left. Read from right to left)
00001111 = hex 0f = cpu cores 0, 1, 2 and 3 selected
11111111 = hex ff = all cpu cores selected
11110000 = hex f0 = cpu cores 4, 5, 6, 7 selected
00110000 = hex 30 = cpu cores 4, 5 selected
11000000 = hex c0 = cpu cores 6, 7 selected

Just use binary and covert it to hex, 1 = select that cpu core and 0 = unselect that cpu core.
```

1.6 Contact

If you need help or consultation please join my rocket.chat server at https://chat.stoplagging.com/invite/zaMu6X you can message me @Starfroz by looking me up under the globe icon after registering and logging in.



External Resources for Nano Pi R4S

R4S Benchmarks by Van Tech Corner on Youtube:

https://www.youtube.com/watch?v=t5xuTy1xn64

R6S Benchmarks by Van Tech Corner on Youtube:

https://www.youtube.com/watch?v=2bCf8Xchrfc

R4S Performance Tweaking: https://forum.openwrt.org/t/nanopi-r4s-rk3399-4g-is-a-great-new-openwrt-device/79143/406

Revision #164
Created 6 January 2021 19:48:19 by Admin
Updated 24 December 2024 07:17:37 by Admin