English • Deutsch • Español • Français • Italiano • ??? • Polski • Português • ??????? • Svenska • ???(????)? • ???(??)? •

Contents

NVRAM

NVRAM (non-volatile RAM) also called flash space is the place where the permanent settings are stored. This includes

◊ DD-WRT settings that you normally change using Web Interface
◊ settings for user Startup Scripts

You can run the following commands using Command Line:

- Show the whole NVRAM content:

```
nvram show
```

- Show NVRAM content containing the pattern <search_pattern> (useful for quickly finding things in NVRAM):

```
nvram show | grep <search_pattern>
```

Replace <replace_pattern> with what you are actually looking for. This can simply be a word or a regular expression

- Show value of a certain variable:

```
nvram get <variable_name>
```

- Change variable value *in RAM only*:

```
nvram set <variable_name>="<value>"
```

(Quotes not needed for numeric/boolean values, needed for texts like `192.168.1.1` or `443 8080` or `sometext`).
Now you can play with the new settings or test user scripts but this new value will be lost after reboot unless you do `nvram commit`.

- Delete a variable (and its value). Useful when you made a typo in `<variable_name>` above:

```
nvram unset <variable_name>
```

- Save all changed variables to NVRAM:

```
nvram commit
```

Reboot the router for new settings in NVRAM to go into effect.

- To reset or erase nvram, please refer to the Reset And Reboot or Hard reset or 30/30/30 wikis.

Boardflags

/* boardflags */

1. define BFL_BTCOEXIST 0x0001 /* This board implements Bluetooth coexistance */
2. define BFL_PACTRL 0x0002 /* This board has gpio 9 controlling the PA */
3. define BFL_AIRLINEMODE 0x0004 /* This board implements gpio13 radio disable indication */
4. define BFL_ENETROBO 0x0010 /* This board has robo switch or core */
5. define BFL_CCKHIPWR 0x0040 /* Can do high-power CCK transmission */
6. define BFL_ENETADM 0x0080 /* This board has ADMtek switch */
7. define BFL_ENETVLAN 0x0100 /* This board has vlan capability */
8. define BFL_AFTERBURNER 0x0200 /* This board supports Afterburner mode */
9. define BFL_NOPCI 0x0400 /* This board leaves PCI floating */
10. define BFL_FEM 0x0800 /* This board supports the Front End Module */
11. define BFL_EXTLNA 0x1000 /* This board has an external LNA */
12. define BFL_HGPA 0x2000 /* This board has a high gain PA */
13. define BFL_BTCMOD 0x4000 /* This board' BTCOEXIST is in the alternate gpios */

Overclocking

BCM3303 v0.7 clock frequencies: (todo...)

MIPS/CPU	Backplane	Comment

BCM3303 v0.8 clock frequencies:

MIPS/CPU	Backplane	Comment
183	91	
188	94	
197	98	
200	100	

206	103
212	106
216	108
217	109
225	113
238	119
240	120
250	125

Flow Acceleration, SFE and Cut-Through Forwarding

As Internet Bandwidth speeds available to the end users have continued to increase, manufacturers have had to increase speed and power of routers to keep up with the higher network speeds. This has 2 negative consequences, first is that powerful router CPU's at high clock speeds are much more expensive, and second that more powerful hardware increases heat production which shortens the life of the equipment. Adding a cooling fan can help this but it decreases router reliability because the fan can fail causing the router to overheat, and new SOHO router prices need to be maintained at approximately 25% of prices of a new Windows desktop otherwise the price/performance of using something like pfsense on an Intel-based PC, or the commercial Untangle on a custom PC is better than any hardware router. (although the reliability is much less and power consumption of the PC is much higher)

As a result research began on seeing if it would be possible to increase packet-handing efficiency from the standard netfilter-based NAT forwarding in Linux and thus maintain the same embedded CPU architecture while accelerating packet throughput.

A number of researchers began to analyze the Netfilter NAT code in Linux and found it to be very inefficient and a number of proposals were made to bypass the Linux TCP/IP stack. These are discussed in the "Related Work' section in the Quick NAT research paper [1] The researchers of that paper identified the most inefficient parts of the Linux NAT and rewrote them as follows - from the paper:

"In order to improve the performance of NAT on commodity platforms, we design Quick NAT system built on DPDK. Quick NAT Search (QNS) algorithm is designed to look up NAT rules with complexity of O(1). In addition, Quick NAT system leverages the lock-free hash table to reduce the expense of locks when sharing NAT mapping records among CPU cores on multicore commodity servers. Moreover, Quick NAT achieves full zero-copy in the process of NAT to cut down the overhead of copy."

The paper made the bold statement:

"the performance of Quick NAT scales out linearly with the number of cores and achieves an improvement of more than 860% compared to that of Linux Netfilter."

Although it may have been adjusted on a test bed optimized for their code, if even 1/2 of what they claimed was true it would indicate a spectacularly inefficient NAT implementation in Linux with enormous room for

improvement by bypassing the Linux network stack.

The earliest acceleration effort in the SOHO wireless router market appeared around 2010 as an experimental "binary blob" module in Broadcom's SDK named "fast NAT" and marked experimental. Although this code wasn't a port of Quick NAT it was undoubtedly based on ideas from earlier researchers that were also used by Quick NAT. Fast NAT was incorporated into TomatoUSB firmware Build 47 in June 2010. It was then disabled in TomatoUSB firmware in Build 48 the following month due to instability. This was the predecessor to the CTF binary blob module from Broadcom. It is purely software NAT acceleration and worked with MIPS processors.

The next acceleration effort was SFE. This was originally named "Fast Classifier" (Fast Crashifier by one wag) It was first posted to GitHub (2 years after the Qualcomm acquisition Atheros in 2011), by Qualcomm in 2013 Qualcomm Shortcut Forwarding Engine and per it's README:

"It was written as a demonstration of what can be done to provide high performance forwarding inside the kernel. There were two initial motivations:

1) To provide a platform to enable research into how QoS analysis systems can offload work and avoid huge Linux overheads.

2) To provide a tool to investigate the behavior of various processors, SoCs and software sets so that we can characterize and design new network processor SoCs."

Not to be outdone Broadcom released it's own version named CTF that appeared in a few commercial firmwares. An early analysis of this by the OpenWRT community was posted Here

For the next 4 years Broadcom continued to work on Fast NAT (CTF) with it making occasional appearances (disabled by default) in different forks of Tomato as bcm_nat, and steadily spreading through manufacturer's K26-based firmware for Broadcom based devices by Netgear, Linksys, and others. Most manufacturer's firmware DID NOT contain Linux kernels with many enhanced features, (such as packet inspection) thus Fast Nat was able to work with them, while the various open projects (Tomato, Openwrt, and all their forks, as well as dd-wrt) used more complete (and modern) Linux kernels so the Broadcom CTF module was unstable or impacted various services (like port forwarding)

There was also parallel work on Fast Classifier (SFE) being done, it was incorporated by Google into the ChromeOS 3.x source code ChromeOS 3.18 SFE source.

Around this time dd-wrt also experimented with a software based NAT speedup called "ddtb" (DD-WRT Turbo Boost) which ultimately proved to have too many problems for production use and was abandoned.

During this period a series of unfavorable comparisons in the media to manufacturer's firmware were made and posted, and users also pushed for CTF to be included (with often disappointing results when enabled) into their preferred open source router firmware. For example the ctf.ko module Broadcom released for the K2.6 kernels was tested back by kong around 2014 in dd-wrt and discovered to be very buggy so dd-wrt never implemented it.

Finally, SFE was ported into OpenWRT LEDE in June 2017 by gwlim Qualcomm Fast Path For LEDEand into dd-wrt by BrainSlayer a month later. Kong's SFE announcement This really helped the cause of high-speed link usage on open source firmware on routers as once the code was part of the source code of these flagship distributions, every other fork and project started incorporating SFE (and very likely a number of hardware manufacturers as well) Today, SFE is enabled by default in the dd-wrt interface.

The Quick NAT project eventually released their source code to QuickNAT, designated QNAT, on Github here [2] This code was developed under Centos and claims up to 10Gbt packet handling ability on x86 hardware.

Broadcom had not been idle with CTF, either. By 2012 they had concentrated on the Northstar line of CPUs and in addition to the software NAT acceleration, introduced hardware acceleration in later Northstar SoCs named Flow Acceleration. CTF was handled by a proprietary software module ctf.ko to create Layer 1 acceleration, while Flow Acceleration (FA) was handled by adding special hardware into it's devices with gigabit Ethernet ports that helps distribute the processing of packets in the router to create Layer 2 acceleration. With Broadcom:

Level 1=CTF Only

Level 2=FA + CTF

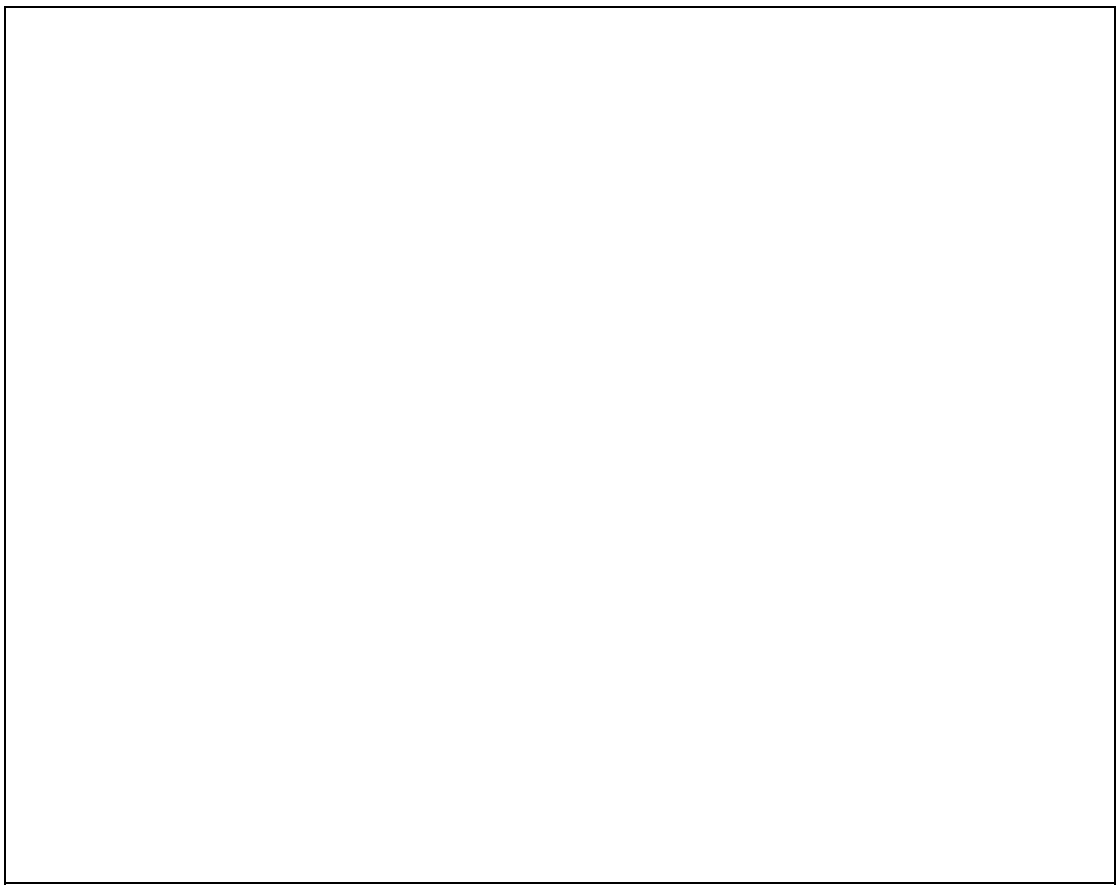
With Layer 2, instead of the primary CPU handling all packet delivery within the router, many packets can be moved directly from interface to interface without the primary CPU being involved in all of the packet handling. While nowadays it is pretty well understood how CTF operates (since the Qnat sourcecode can be compiled by anyone and used to show just how inefficient the Linux Netfilter NAT code is, and it's obvious that CTF works the same way) FA is still somewhat of a black box. Some speculation is that it uses a specialty NAT cpu incorporated into the ethernet switch block of the SoC, and checks every incoming packet into the router to see if it can be diverted into this CPU.

FA was officially incorporated into dd-wrt in May of 2021. (after years of waiting for Broadcom to port it to the K4 kernels and release a binary) CTF works with many Broadcom chips while FA works with some of the Northstar line of Broadcom chips, and SFE works with Atheros, Broadcom, and all other chips (including Intel).

In summary, CTF and SFE are software optimization techniques to accelerate NAT, while FA is a Hardware NAT acceleration mechanism. SFE works on all dd-wrt routers while CTF requires Broadcom CPU's and FA requires both Broadcom CPUs and specialty hardware in the Broadcom SoC

Once dd-wrt is loaded on your router, click Status and look at the router CPU. If it is a newer Broadcom CPU you should have CTF. If it is Broadcom and an ARM chip then you can check this list to see if your CPU is a Northstar CPU If it is, you will get both FA and CTF. Note that since a few Northstar chips do NOT have the FA hardware, there is an autodetection in dd-wrt that will disable the FA selection from appearing if the hardware is NOT detected. This autodetection is not active on the first boot AFTER a NVRAM wipe. So IF your router hardware supports FA, it is necessary on the first boot after a factory reset/nvram erase to configure the WAN interface, save and apply the configuration, then reboot the router. Then login again and see if the FA option is present.

Note the following screenshot of a Linksys EA6900 with both options. CTF is accessible from the dropdown for Shortcut Forwarding Engine (SFE) while the Flow Acceleration is listed below it. If your CPU is not a Northstar CPU but does have CTF then you will have the dropdown for CTF in the Shortcut Forwarding Engine but not the Flow Acceleration selection. If it's not Broadcom you will only have SFE



Note that for Flow Acceleration to be visible and selectable, the WAN interface of the router MUST be configured EITHER with DHCP or with a Static IP address, it must also have QoS disabled, and AFTER the configuration is saved and applied, the router MUST be rebooted for the option to appear. CTF Level 1 does not require QoS to be disabled, however, so if you need QoS you can still use it however packets going through the CTF bypass may ignore QoS settings.

Which scheme is faster

From a discussion on Build r46690

"ctf is much faster. with sfe i reach about 700 mbit with lan->wan tests. with ctf i get full gbit speed. so 950 - 980 mbit and lower cpu usage.... i'm running it here for 2 weeks now on several routers (r7000, ac66, n66)" --BrainSlayer

What kind of speed boost does this give me?

If you have a fairly fast router CPU such as 300Mhz or above, and a slower cable or DSL line, then you probably won't see any difference. If you have a 500Mbt or above line, you can see a big difference. For example without FA you might max out at 700Mbt on a gigabit Internet line.

For testing throughput, a baseline should be established with Shortcut Forwarding Engine (SFE) disabled and Flow Acceleration disabled. Then enable SFE and test, then switch to CTF and test, then switch to FA and CTF. Test results are going to be specific to the dd-wrt version and the router model.

Caveats:

- While SFE is open source, CTF and FA are binary blob drivers from Broadcom. NOBODY other than Broadcom knows exactly how they work or what exactly they are doing, the links below are all speculations on what is actually going on. The fact that CTF requires Broadcom devices is likely due to tests for Broadcom CPUs inside of the CTF module - Broadcom spent considerable time developing the CTF code as a sales tool to sell Broadcom SoCs. We also know for a fact that FA uses specialty hardware not just because FA requires Northstar and ARM devices but Broadcom lists Flow Acceleration hardware in their sales literature for their SoCs.

- Packets that are modified by the router (such as packets being stuck into an OpenVPN tunnel) cannot go through SFE, CTF or FA and the router will continue to process them as before.

- It is possible that some types of packets (such as VoIP packets) can be disrupted by having the CTF and FA options turned on. (SFE generally is not a problem)

- Traffic Monitoring, (such as YAMon) PPPoE, and NAS fileserving don't work with these options.

- Port forwards of some protocols may be interfered with.

- Because the FA and CTF code is not published it is impossible for a 3rd party to audit it for security.

- Most of these apply to wired ethernet to ethernet connections. Of course, Gigabit wifi speeds require an 802.11ac or above router (Netgear Nighthawk, etc.). The CTF module attaches to the Wireless interface as well as the LAN interfaces so it may accelerate wifi connections.

Broadcom docs:

Youtube video of Northstar Flow Accelerator Computex 2012 Demo This one is fascinating almost as much for the lack of identification of who is speaking, who he works for, what router model is sitting there taken apart, what router software is in use and what testing methodology is being used. The group filming the video implies it is a Broadcom representative.

It appears that Broadcom uses the same technology in their higher end hardware, under the names TruFlow and vSwitch acceleration.

Datasheet on the BCM957414A4142CC The BCM957414 is a Dual-Port 25 Gb/s SFP28 Ethernet PCI Express 3 x8 Network Interface Card that mentions the "TruFlow? flow processing engine" and "vSwitch acceleration" features

Datasheet on the Stingray PS250 This sheet expands on TruFlow, stating:

"The powerful TruFlow? configurable flow accelerator adds a powerful packet inspection and processing capability in the hardware, moving common flow-processing workloads into hardware and freeing CPUs for application workloads"

Datasheet for the bcm58712 CPU that states:

"Broadcom?s TruFlow? packet flow processing technology enables Open vSwitch acceleration through CPU offload"

Datasheet for the bcm4707, etc. that states:

"Hardware Flow Accelerator in the Overview"

OpenWRT investigations on the older K26 CTF module

OpenWRT Technical Explanation of ctf.ko module and claims

Initial OpenWRT discovery and speculation on CTF

OpenWRT discussion of profiling the acceleration module

Speculation of symbols in the ctf.ko module and what they might mean

Hardware Modifications

Hardware Modifications (mods) are not necessarily DD-WRT specific or compatible.

- Serial Port
- Additional external SMA-R connector
- Dual Serial Port
- SD-Card
- Temperature sensor for WRT54G 2.x and WRT54GS with DS1820
- DIR-320 64 Mb RAM Upgrade