Honey, I shrunk DNSdist

Remi Gacogne

FOSDEM 25 - Feb. 1st, 2025

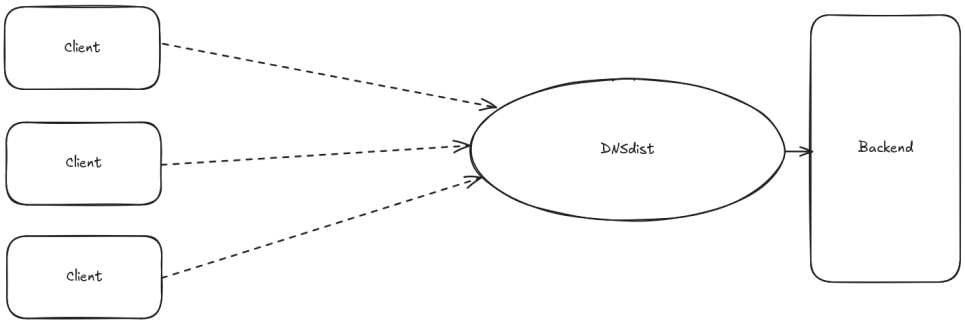
1 Initial goal and misconceptions

What is DNSdist

A DNS proxy:

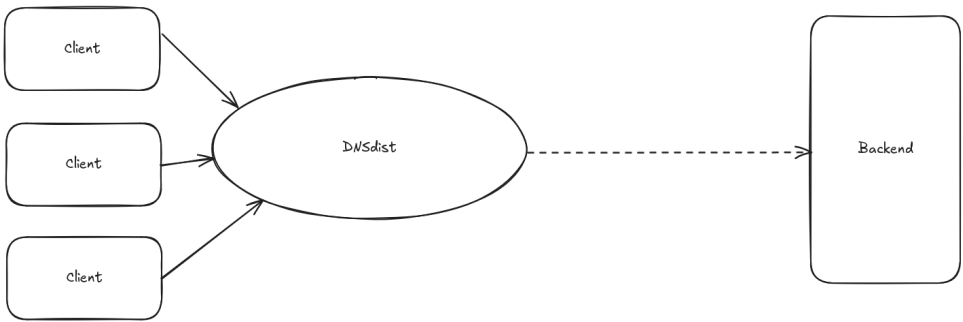
- Do53, DNSCrypt, DNS over TLS, HTTPS, QUIC and HTTP/3
- Multiple backends with load-balancing options
- Health-checks
- Efficient packet-cache
- DoS protection

DNSdist as a reverse proxy



Mostly used as a reverse proxy

DNSdist as a proxy



But as a proxy as well

The goal

Around 2021:

Several people: “Hey, DNSdist would be a great fit for DNS encryption on OpenWrt-powered CPEs”
Us: “Sure, it’s a tiny program, with a small memory footprint”
Them: “About that..”

The goal - OpenWrt supported devices

In 2022, 19.07.10 supported devices with:

- Minimum amount of flash: 4 MB
- Minimum amount of RAM: 32 MB

The (reasonable) goal

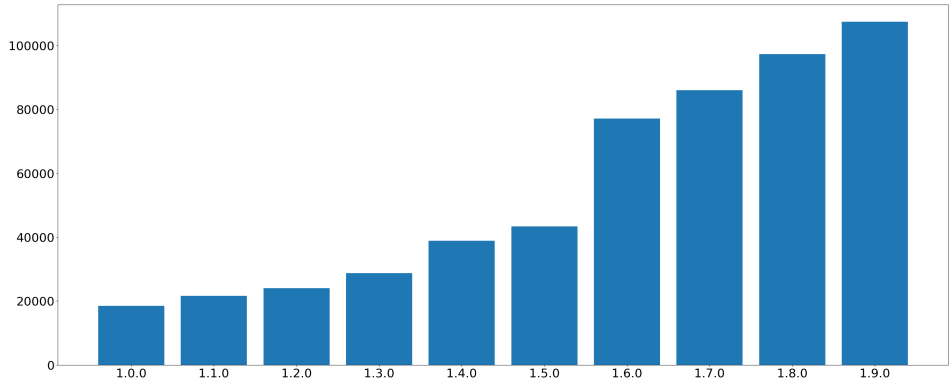
Let’s pick a target that is, at least in theory, capable of handling encrypted DNS:

TP-Link Archer C7 AC1750 v2+ with 16 MB of flash, 128 MB of RAM.

We should be able to have a binary size of a couple MB, and a memory footprint smaller than 4 MB, right?

How far are we?

DNSdist has got a fair amount of features over the years, and binary size has grown: 9 MB on x86_64



How far are we? Our misconceptions

In the OpenWrt world, what matters is:

- The compressed size on the flash
- The proportional set size (PSS): Private memory of that process plus the proportion of shared memory with one or more other processes

Proportional set size (PSS)

On Linux, the PSS of a process can be found via:

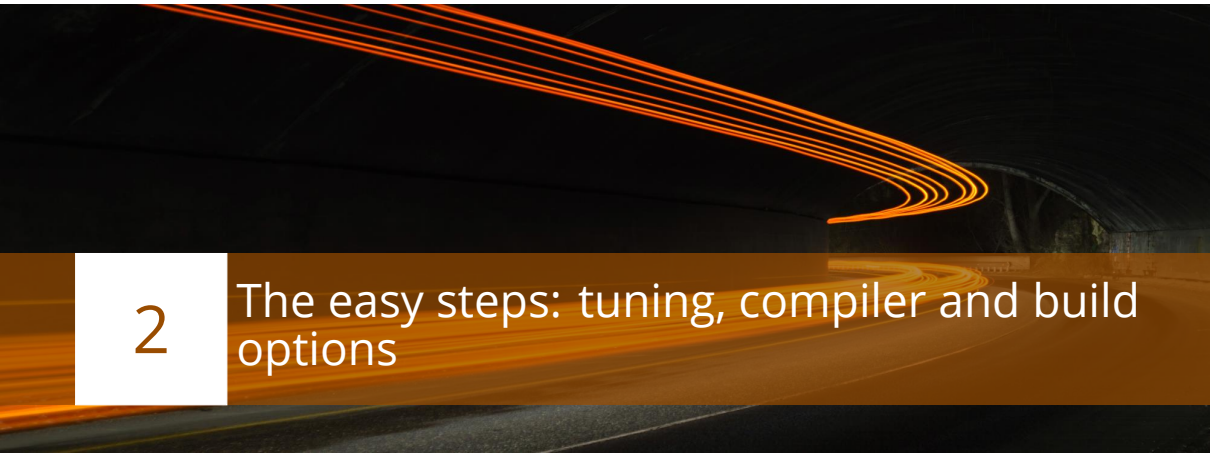
- */proc/<pid>/smaps* with per VMA entries
- */proc/<pid>/smaps_rollup* with pre-summed values (slightly more accurate) plus:, plus *Pss_Anon*, *Pss_File* and *Pss_Shmem*
- *Pss_Dirty*: memory that has been altered
- *Pss_Anon*: anonymous memory, mostly the heap
- *Pss_File*: memory associated with files that has not been modified (can technically be discarded)
- *Pss_Shmem*: memory shared with other processes

Proportional set size (PSS) for DNSdist

For DNSdist on OpenWrt, at this point, that means:

- 1.8 MB of DNSdist
- 1.7 MB of libcrypto
- 440 kB of libssl
- 850 kB of libstdc++
- ...
- 1.9 MB of heap

Which is waaaaaaaay over our target. Crap, it's not going to be easy.



2 The easy steps: tuning, compiler and build options

Tuning the configuration

```
1 setMaxUDPOutstanding(50) -- 50 concurrent, in-flight, UDP queries
2 setMaxTCPClientThreads(1) -- a single TCP worker thread for TCP/DoT/DoH
3 setOutgoingDoHWorkerThreads(1) -- a single TCP worker thread for outgoing DoH
4 setRingBuffersSize(300, 1) -- we keep track of the last 300 queries
5 setRingBuffersOptions({recordResponses=false}) -- and no responses
6 setMaxTCPQueuedConnections(10) -- reduce the queue for pending TCP connections
7 setOutgoingTLSSessionsCacheMaxTicketsPerBackend(5) -- only keep 5 TLS tickets per
  ↪ backend
8 setTCPInternalPipeBufferSize(0) -- reduce the size of the buffer when dispatching
  ↪ TCP connections
```

Build options

Start with the obvious step: **disable non-essential features** in *dnsdist*, and move them to *dnsdist-full*

- carbon
- CDB
- console (including help messages!)
- dnstap
- ebpf filtering
- ipcipher
- LMDB
- prometheus
- re2
- SNMP
- ...

Compiler and linker flags

And tell the compiler and linker to help us:

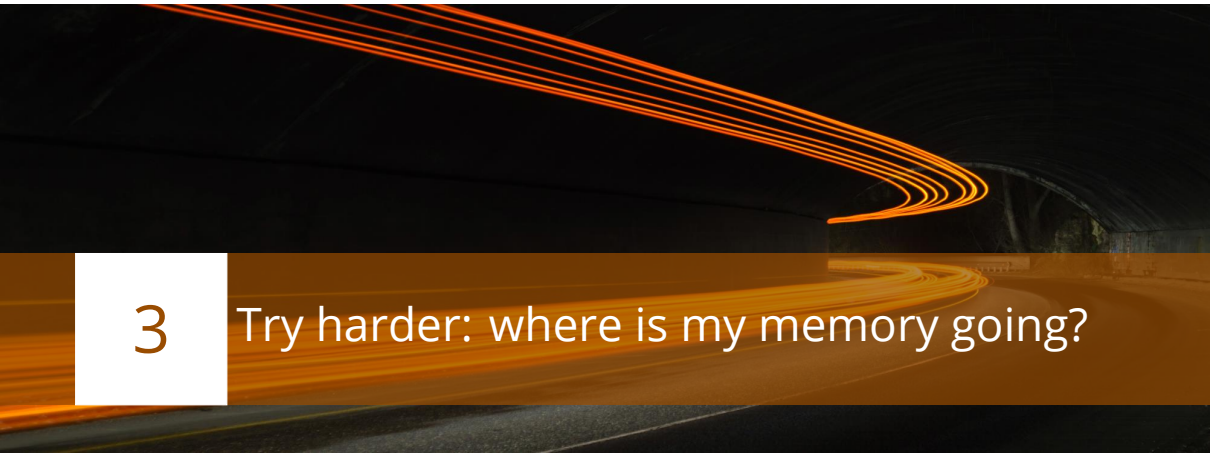
- Optimize for the size of the binary: **-Os**
- Hide symbols to external modules by default: **-fvisibility=hidden**
- Enable Link-Time Optimizations, which amongst other things remove code that is not actually needed (depending on the options enabled: **-flto**
- **-fno-ipa-cp** works around a bug in the LTO of the specific GCC version used by the OpenWrt 19 toolchain
- Remove assertions (**-DNDEBUG**) and strip debug symbols
- Disable **PIE**: it reduces the size of the binary: we can enable it on devices that have enough memory

Results

That’s a great start:

- Compressed binary size is now below 2 MB
- PSS is automatically lower, but still over target: around 6 MB

But we are not there yet.



3 Try harder: where is my memory going?

Looking at the heap

Let's start with our direct memory usage, on the heap:

- Looking at the values in *smaps_rollup*, we see almost 2 MB
- Surely we can do better, but where does that memory usage come from?
- Hard to investigate on the device itself, but we can do that in a emulator or extrapolate what we see on Linux amd64

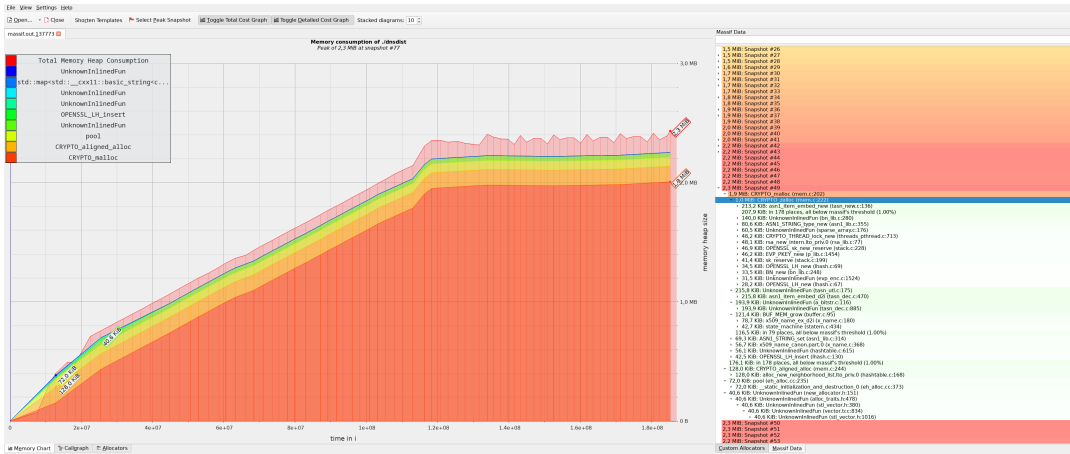
Massif: a heap profiler

Massif[1] is the heap profiler of the Valgrind suite:

- can profile page allocations (pool allocators not using *malloc*)
- can profile stack allocations
- results can be visualized with *massif-visualizer*
- reaaaaaaally slow

[1] https://valgrind.org/docs/manual/ms-manual.html

Massif: a heap profiler



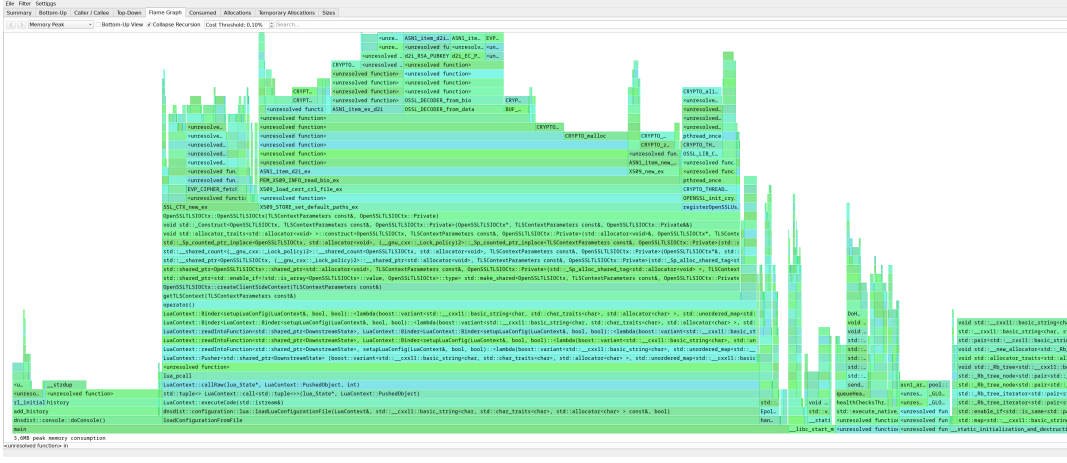
Heaptrack: a heap memory profiler for Linux

Heaptrack[2] is the heap profiler of the KDE project:

- keeps more data than massif, which aggregates data early in the process
- much more efficient: doesn't slow the process that much, and doesn't consume a lot of memory

[2] https://github.com/KDE/heaptrack

Heaptrack: a heap memory profiler for Linux



Memory usage improvements in DNSdist

This led to multiple improvements in our code:

- Don’t load useless ciphers and digests[3]
- Don’t load OpenSSL error messages[4]
- Don’t waste memory by allocating then shrinking[5]

[3] https://github.com/PowerDNS/pdns/pull/11166
[4] https://github.com/PowerDNS/pdns/pull/11988
[5] https://github.com/PowerDNS/pdns/pull/11171

Memory usage improvements in libh2o-evloop

And in *libh2o-evloop* for *OpenWrt*[6]:

- Drop the libyaml dependency
- Remove a MIME map only used to serve static files, and using a fair amount of memory
- Reduce the initial size of the HTTP/2 buffer from 80 kB to 8 kB which is more than enough for most DNS queries

Note that since 1.9.0 DNSdist no longer uses h2o, using nghttp2 for incoming queries instead.

[6] https://github.com/openwrt/packages/pull/21365

Things that did not help in this round

Using wolfSSL[7] instead of OpenSSL:

- surprisingly easy, nice OpenSSL compatibility layer
- did not reduce the memory usage significantly in our tests
- actually hurts as soon as a second user of OpenSSL shows up
- OpenWrt 21 switched wolfSSL as the default TLS provider, which should have helped but we did not see any major change in our case

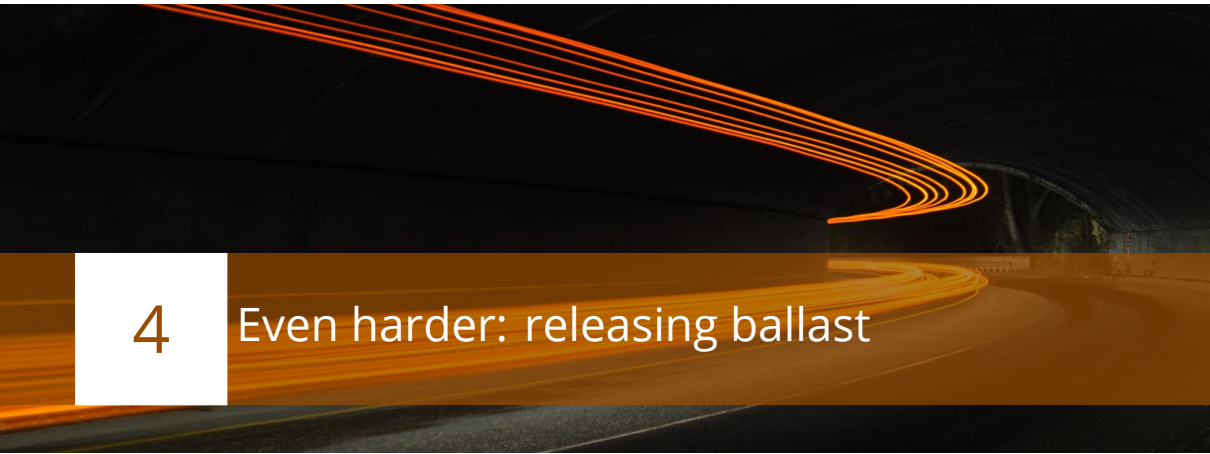
[7] https://www.wolfssl.com/

Things that did not help in this round (bis)

Using a packer (UPX[8]):

- reduces binary size
- no visible change to the PSS since the compressed parts still have to be uncompressed in memory

[8] https://upx.github.io/



4 Even harder: releasing ballast

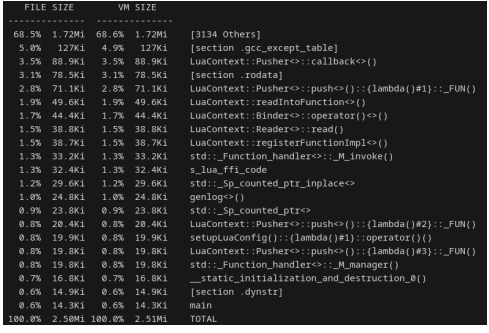
Can we reduce our binary size further without removing features?

Finding out what is inside our binary with Bloaty[9]:

- Compile our binary with the features we want
- Copy the binary, strip it
- Run *bloaty <stripped copy> –debug-file=<original binary> -d symbols*

[9] https://github.com/google/bloaty

Can we reduce our binary size further without removing features?



FILE SIZE		VM SIZE		
68.5%	1.72Mi	68.6%	1.72Mi	[3134 Others]
5.0%	127Ki	4.9%	127Ki	[section .gcc_except_table]
3.5%	88.9Ki	3.5%	88.9Ki	LuaContext::Pusher<>::callback<>()
3.1%	78.5Ki	3.1%	78.5Ki	[section .rodata]
2.8%	71.1Ki	2.8%	71.1Ki	LuaContext::Pusher<>::push<>()::{lambda()#1}::_FUN()
1.9%	49.6Ki	1.9%	49.6Ki	LuaContext::readIntoFunction<>()
1.7%	44.4Ki	1.7%	44.4Ki	LuaContext::Binder<>::operator()<>()
1.5%	38.8Ki	1.5%	38.8Ki	LuaContext::Reader<>::read()
1.5%	38.7Ki	1.5%	38.7Ki	LuaContext::registerFunctionImpl<>()
1.3%	33.2Ki	1.3%	33.2Ki	std::_Function_handler<>::_M_invoke()
1.3%	32.4Ki	1.3%	32.4Ki	s_lua_ffi_code
1.2%	29.6Ki	1.2%	29.6Ki	std::_Sp_counted_ptr_inplace<>
1.0%	24.8Ki	1.0%	24.8Ki	genlog<>()
0.9%	23.8Ki	0.9%	23.8Ki	std::_Sp_counted_ptr<>
0.8%	20.4Ki	0.8%	20.4Ki	LuaContext::Pusher<>::push<>()::{lambda()#2}::_FUN()
0.8%	19.9Ki	0.8%	19.9Ki	setupLuaConfig()::{lambda()#1}::operator()()
0.8%	19.8Ki	0.8%	19.8Ki	LuaContext::Pusher<>::push<>()::{lambda()#3}::_FUN()
0.8%	19.8Ki	0.8%	19.8Ki	std::_Function_handler<>::_M_manager()
0.7%	16.8Ki	0.7%	16.8Ki	__static_initialization_and_destruction_0()
0.6%	14.9Ki	0.6%	14.9Ki	[section .dynstr]
0.6%	14.3Ki	0.6%	14.3Ki	main
100.0%	2.50Mi	100.0%	2.51Mi	TOTAL

POWERDNS

Reducing the size of important structures: inefficient padding

```
(gdb) ptype /o Rings::Response
/* offset      |    size */  type = struct Rings::Response {
/*      0      |      16 */    struct timespec {
[...]
/* XXX 12-byte padding   */
                               } requestor;
/* XXX  4-byte hole      */
/*     72      |       2 */    uint16_t qtype;
/* XXX  2-byte hole      */
[...]
/* XXX  4-byte padding   */
[...]
/* total size (bytes):  128 */
```

After the PR*https://github.com/PowerDNS/pdns/pull/10381* fixing it: “total size (bytes): 120”

Giving up on false sharing

False sharing[10]:

- degrades performance of threaded programs when atomic variables are aligned next to each other
- is mitigated in PowerDNS by aligning atomics on the size of a cache line
- which uses significant memory
- so we added an option to disable this alignment on OpenWrt

[10] https://en.wikipedia.org/wiki/False_sharing

Reducing the number of threads

DNSdist uses several threads to handle queries and responses, for maximum performance:

- 2 threads per listening address and port, accepting UDP queries and TCP/DoT/DoH connections from clients
- 1 thread per backend, accepting UDP responses from backend
- several TCP workers, handling TCP/DoT connections from clients as well as TCP/DoT connections to backends (event-based)
- several DoH workers, handling DoH connections to backends (event-based)

Reducing the number of threads (2)

Implemented a collapsed model[11] for OpenWrt, reducing CPU and RAM needs:

- 1 thread handling incoming UDP queries
- 1 thread handling TCP/DoT connections, both incoming and outgoing, plus incoming DoH
- 1 thread handling outgoing DoH connections
- 1 thread per backend, accepting UDP responses from backend

[11] https://github.com/PowerDNS/pdns/pull/12003

Memory fragmentation

musl libc 1.1.24's memory allocator, used in OpenWrt 19 and 21, is known to have fragmentation issues[1][2]

- getting rid of a few short-lived allocations (outside of the hot path) thanks to heaptrack
- triggering the Lua garbage collector (*collectgarbage()*) at strategic points during the configuration also helped a bit
- reducing the number of threads also reduced fragmentation

[1][2] //www.openwall.com/lists/musl/2018/04/20/1

Things that did not help in this round

Statically linking *libstdc*++ in:

- slightly reduces the PSS
- means we would have to recompile DNSdist in case of a security issue/bad enough bug in libstdc++
- will actually hurt the PSS if a second C++ program is loaded (bigger binary, not shared)

Time to call it

- still slightly over the target that we initially hoped to reach
- but return on investment of further investigations was falling very fast
- so it was time to contribute our improvements

5 Putting it all together

Current status

As of OpenWrt 23.05.5, for the *mipsel_24kc* architecture:

- Compressed size of dnsdist-full: 1.5 MB (5.5 MB uncompressed)
- Compressed size of dnsdist: 630 KB (1.7 MB uncompressed)
- PSS is roughly 4 MB with several backends, incoming Do53, DoT and DoH, outgoing DoT and DoH

UCI integration

In our own OpenWrt repository[13] we also offer *dnsdist-maindns*, which replaces dnsmasq with DNSdist as the DNS provider on the CPE.

It also includes UCI[14] integration to make DNSdist easy to use, including via the Web interface. We have contributed[15] this to OpenWrt a few months ago but it has not been accepted yet.

[13] https://repo.powerdns.com/openwrt/
[14] Unified Configuration Interface
[15] https://github.com/openwrt/packages/pull/25398

UCI integration example I

```
config dnsdist general
  option 'enabled' '1'
  # private key to use for incoming DoT and DoH
  option 'tls_key' '/etc/dnsdist.key'
  # certificate to use for incoming DoT and DoH
  option 'tls_cert' '/etc/dnsdist.pem'
  # Number of entries in the domain cache. 0 means caching is disabled, the maximum value is 2^32-1 entries
  # but is likely limited by the amount of RAM available.
  option 'domain_cache_size' '100'
  # The DNS suffix, or list of suffixes, identifying local domain names
  option 'local_domains_suffix' 'lan'
  # Port on which to accept Do53 queries on (UDP and TCP)
  option 'do53_port' '53'
  # Port on which to accept DoT queries on
  option 'dot_port' '853'
  # Port on which to accept DoH queries on
  option 'doh_port' '443'
  # Whether upstream resolvers learned via the system should be checked for DoT/DoH support
  option 'auto_upgrade_discovered_backends' '1'
  # Whether the Do53 version of auto-upgraded resolver should be kept as fallback
  option 'keep_auto_upgraded_backends' '1'
  # User to switch to, after the configuration has been set up. Default is to run as user 'root'
  option 'user' 'dnsdist'
  # Group to switch to, after the configuration has been set up. Default is to run as group 'root'
  option 'group' 'dnsdist'
```

UCI integration example II

```
config interface
  option name 'default_interface'
  # Whether dnsdist will listen on this interface
  option enabled 1
  # Whether dnsdist will accept Do53 queries, UDP and TCP, on this interface
  option do53 1
  # Whether dnsdist will accept DoT queries on this interface
  option dot 1
  # Whether dnsdist will accept DoH queries on this interface
  option doh 1
  # Whether local domain resolution will be enabled for queries received on this interface
  option local_resolution 1
  # Whether dnsdist will advertise DoT and/or DoH support, if available, in response to DDR queries received on this interface
  option advertise 1
```

UCI integration - DDR advertisement

To let local devices know that our CPE supports DoT and DoH, allowing them to automaticall upgrade to a secure transport:

```
Reply to question for qname='_dns.resolver.arpa.', qtype=SVCB
Rcode: 0 (No Error), RD: 0, QR: 1, TC: 0, AA: 0, opcode: 0
0       _dns.resolver.arpa.     60      IN      SVCB    1 _dns.resolver.arpa. mandatory=port alpn=dot no-default-alpn\
    port=8443 ipv4hint=127.0.0.1 ipv6hint=::1
0       _dns.resolver.arpa.     60      IN      SVCB    2 _dns.resolver.arpa. mandatory=port alpn=h2 port=10443 \
        ipv4hint=127.0.0.1 ipv6hint=::1 key7="/dns-query"
2       _dns.resolver.arpa.     60      IN      A       127.0.0.1
2       _dns.resolver.arpa.     60      IN      AAAA    ::1
```