Knot DNS Resolver

Modulární rekurzivní resolver

Karel Slaný • karel.slany@nic.cz • 13. 11. 2015

CZ.NIC | SPRÁVCE DOMÉNY CZ

Obsah

- Co je KNOT Resolver
- Části resolveru
- Funkce a konfigurace
- Integrační testování

Co je Knot DNS Resolver

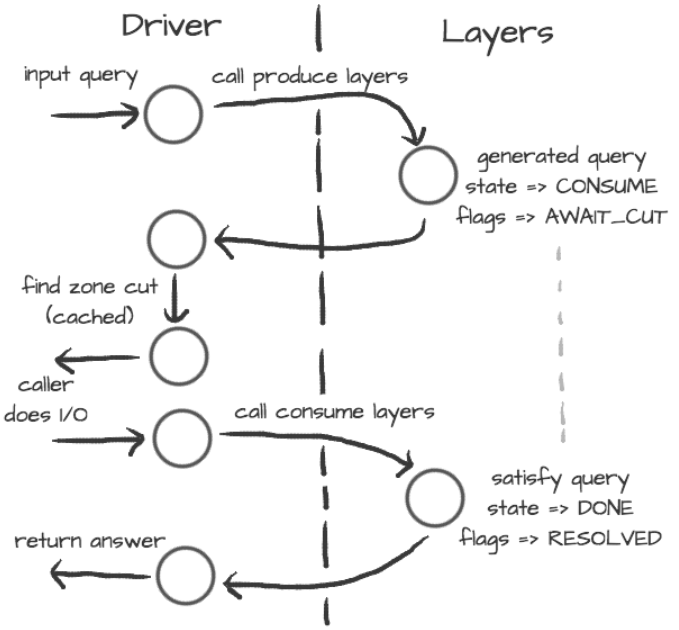
- Minimalistický open-source resolver
- Vychází z Knot DNS
  - libknot, libdnssec
- Modulární architektura
  - Kompaktní jádro
  - Vrstvy pro zpracování dotazů
- Knihovna a daemon

Knihovna libkresolve

- Základní služby pro resolvování
  - Využívá procesní API knihovny libknot
  - Řízení stavovým automatem
  - Data předávána pro zpracování vrstvám
- Poskytuje
  - Rozhraní pro jednotlivé vrstvy
  - Sdílená cache
  - Plán resolvování
  - Reputační databáze jmenných serverů

```
// Create request and its memory pool
struct kr_request req = {
    .pool = { .ctx = mp_new (4096),
              .alloc = (mm_alloc_t) mp_alloc }
};

// Setup and provide input query
int state = kr_resolve_begin(&req, ctx, final_answer);
state = kr_resolve_consume(&req, query);

// Generate answer
while (state == KNOT_STATE_PRODUCE) {

    // Additional query generate, I/O and pass back answer
    state = kr_resolve_produce(&req, &addr, &type, query);
    while (state == KNOT_STATE_CONSUME) {
        int ret = sendrecv(addr, proto, query, resp);

        // If I/O fails, make "resp" empty
        state = kr_resolve_consume(&request, addr, resp);
        knot_pkt_clear(resp);
    }
    knot_pkt_clear(query);
}

kr_resolve_finish(&request, state);
```

Vrstvy



- Funkce vrstev:
  - Sledování procesu zjišťování odpovědi
    - Logování, statistiky
  - Modifikace procesu zjišťování odpovědi
    - Produkce dalších dotazů
    - Konzumace a zpracování odpovědi
- Komunikují mezi sebou a jádrem pomocí příznaků.
- Mohou být za běhu nahrávány jako moduly.

Daemon kresd

- Implementace resolveru využívající libkresolve
- Perzistentní cache
- DNSSEC
  - Automatická aktualizace kořenů důvěry (RFC 5011)
  - Negativní kořeny důvěry (RFC 7646)
- Interaktivní rozhraní v příkazové řádce
  - Interpret jazyka Lua
  - Implementována základní nápověda
- Vrstvy lze implementovat v C/C++, Go, Lua

Škálovatelnost

- Lze spustit více procesů současně (Linux – SO_REUSEPORT).
  - V závislosti na vytížení lze spouštět/ukončovat procesy.
- Jednotlivé instance mohou obsluhovat odlišná rozhraní.
  - Mohou nadále sdílet pracovní adresář a cache.
- Ke konzoli běžícího procesu se lze připojit za běhu.
- Zatím není řešení pro víceuzlové clustery.
  - Připravovaný modul hive

Konfigurace

- Daemon použitelný přímo bez konfigurace
- Konfiguraci lze provádět za běhu.
- Konfigurační soubor používá syntaxi Lua.
  - Dynamická konfigurace.
  - Skriptovatelné události.

```
-- every 5 minutes
event.recurrent(5 * minute, function()
        cachectl.prune()
end)
```

```
modules = {
   'hints  > iterate', -- Hints AFTER iterate
   'policy > hints',   -- Policy AFTER hints
   'view   < rrcache'  -- View BEFORE rrcache
}
modules.list() -- Check module call order

if hostname() == 'hidden' then
        net.listen(net.eth0, 5353)
else
        net = { '127.0.0.1', net.eth1.addr[1] }
end

for name, addr_list in pairs(net.interfaces()) do
        net.listen(addr_list)
end
```

Moduly

- Static hints – statické hinty
- Statistics collector – čítače a metriky
- Query policies – blokování nebo úpravy dotazů
- Views and ACLs – filtry, řízení přístupu, pohledy
- Prefetching – obnovování expirujících záznamů
- Graphite – vizualizace metrik
- Memcached, Redis – úložiště cache
- Etcd – hlídání změn konfigurace
- Cache control – manipulace cache
- Web interface – informace o výkonu a stavu
- DNS64 – překlad IPv4 na IPv6 (RFC 6147)

Integrační testování

- Deckard – testovací prostředí v Pythonu pro řízené testování
- Testy jsou popisovány ve scénářích
  - Konfigurace
  - Komunikační scénář
  - Kontrolují se odchylky od scénáře
- Univerzální použitelnost
  - Knot DNS Resolver
  - Power DNS Recursor

Podporované platformy

Část	Podporované platformy	Kompatibilita
daemon	UNIX-like, MS Windows	C99, libuv pro abstrakci I/O
knihovna	UNIX-like, MS Windows	MinGW (ne MSVC)
moduly	závisí na modulu	
integrační testy	UNIX-like	

Překlad a instalace

- Závislosti
  - libknot 2.0+
  - LuaJIT 2.0+
  - Libuv 1.7+
  - Pkg-config
- Překlad
  - make PREFIX="/usr/local" install
  - make lib lib-install

- docker run cznic/knot-resolver

Dokumentace

- https://gitlab.labs.nic.cz/knot/resolver
- http://knot-resolver.readthedocs.org/

- https://gitlab.labs.nic.cz/knot/deckard

Děkuji za pozornost

Karel Slaný • karel.slany@nic.cz

CZ.NIC | SPRÁVCE DOMÉNY CZ