Le serveur DNS Knot

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 4 janvier 2014

https://www.bortzmeyer.org/knot.html

Curieusement, des tas d'endroits où tourne un serveur DNS faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html>, font tourner BIND. Ce qui est curieux, ce n'est pas ce choix (BIND est un excellent logiciel), c'est le fait qu'aucune alternative n'a été considérée. Peut-on vraiment se dire administrateur système si on ne se pose même pas la question « y a-t-il d'autres possibilités ? ». Parmi les alternatives sérieuses, pour cette fonction de serveur DNS faisant autorité, le logiciel Knot <https://www.knot-dns.cz/>.

J'ai déjà présenté ici une autre alternative, le logiciel NSD <https://www.bortzmeyer.org/nsd.html>. NSD est extrêmement rapide et fiable mais sa principale faiblesse (qui a été corrigée en version 4) était que l'ajout ou le retrait d'une zone à sa configuration nécessitait le redémarrage du serveur. NSD est donc très bien pour les gérants de TLD ou d'autres grandes zones mais il ne convient pas forcément au cas d'un hébergeur gérant des milliers de zones, avec ajouts et retraits fréquents. Autre manque, NSD ne gère pas les mises à jour dynamiques du RFC 2136 [1]. NSD est conçu pour deux choses, vitesse et fiabilité. Si on veut la richesse fonctionnelle, Knot <https://www.knot-dns.cz/> mérite un coup d'œil.

Les tests ci-dessous ont été faits avec la version 1.4rc2, la 1.4 définitive devrait sortir « bientôt ». Voici la configuration minimale d'un serveur Knot gérant example.com :

```
interfaces {
  my-iface { address 0.0.0.0;}
}

zones {
  storage "/var/knot";
  example.com {
    file "./example.com"; # Relative to "zones / storage"
  }
}
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2136.txt

Indiquer au moins une interface est obligatoire, Knot est sourd, autrement. On peut alors démarrer le serveur (ici, à la main, pour des tests mais, en production, cela serait fait par un script de démarrage) :

```
# knotd -c ultra-simple.conf -v
2014-01-03T14:46:31 Reading configuration '/home/stephane/System/DNS/Knot-tests/ultra-simple.conf' ...
2014-01-03T14:46:31 Knot DNS 1.4.0-rc2 starting.
2014-01-03T14:46:31 Binding to interface 0.0.0.0 port 53.
2014-01-03T14:46:31 Configured 1 interfaces and 1 zones.
2014-01-03T14:46:31 Server started in foreground, PID = 12472
2014-01-03T14:46:31 Server running without PID file.
2014-01-03T14:46:31 Zone 'example.com.' loaded (serial 2013122101)
2014-01-03T14:46:31 Loaded 1 out of 1 zones.
2014-01-03T14:46:31 Starting server...
2014-01-03T14:46:31 Binding remote control interface to /usr/local/var/run/knot/knot.sock
```

Et on peut tester que le serveur répond bien :

```
% dig @192.168.2.7 SOA example.com

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> @verite SOA example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36598
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;example.com. IN SOA

;; ANSWER SECTION:
example.com. 600 IN SOA ns1.example.com. root.example.com. (
2013122101 ; serial
604800     ; refresh (1 week)
86400      ; retry (1 day)
2419200    ; expire (4 weeks)
86400      ; minimum (1 day)
)

...

;; Query time: 1 msec
;; SERVER: 192.168.2.7#53(192.168.2.7)
;; WHEN: Fri Jan  3 15:46:35 2014
;; MSG SIZE  rcvd: 127
```

Knot dispose, comme BIND et contrairement à NSD (avant la version 4) d'un canal de contrôle. Par défaut, ce canal passe par une prise locale, ici `/usr/local/var/run/knot/knot.sock`. Le programme `knotc` permet de parler au serveur (`knotd`) :

```
# knotc  status
OK

# knotc  zonestatus
example.com. type=master | serial=2013122101 |  busy |
```

Contrairement à NSD (avant la version 4), on peut ajouter des zones « en vol ». Ajoutons dans le fichier de configuration :

```
  example.net {
    file "./example.net";
  }
```

Et un simple :

```
# knotc  reload
OK
```

sera suffisant pour charger la nouvelle zone :

```
2014-01-03T14:54:02 Reloading configuration...
2014-01-03T14:54:02 Zone 'example.com.' is up-to-date (serial 2013122101)
2014-01-03T14:54:02 Zone 'example.net.' loaded (serial 2013122101)
2014-01-03T14:54:02 Loaded 2 out of 2 zones.
2014-01-03T14:54:02 Configuration reloaded.
```

Le canal de contrôle peut être mis sur autre chose qu'une prise locale :

```
remotes {
        ctl { address 127.0.0.1;}
}

control {
listen-on { address 127.0.0.1@5533;}
        allow ctl;
}
```

Et le serveur répond alors sur cette prise réseau :

```
% knotc -s localhost -p 5533 status
OK

% knotc -s localhost -p 5533 zonestatus
example.com. type=master | serial=2013122101 |  busy |
```

Autre fonction intéressante, la mise à jour dynamique. Si on configure une zone avec `update-in` :

```
remotes {
        updater { address 127.0.0.1;}
}

zones {
  storage "/var/knot";
  example.com {
    file "./example.com";
   update-in {
  updater;
   }
  }
}
```

Et qu'on utilise, par exemple, ce script (qui utilise le `nsupdate` livré avec BIND) pour les mises à jour :

```
#!/bin/sh

nsupdate  -d <<EOF
  server 127.0.0.1
  zone example.com
  update delete monportable.example.com
  update add monportable.example.com 300 A 192.0.2.42
  send
EOF
```

La mise à jour est bien faite :

```
2014-01-03T15:04:58 UPDATE of 'example.com.' from '127.0.0.1@22672' Started.
2014-01-03T15:04:58 UPDATE of 'example.com.' from '127.0.0.1@22672' Finished.
```

Ce qu'on peut vérifier avec dig :

```
% dig -p 5353 @127.0.0.1 A monportable.example.com
...
;; ANSWER SECTION:
monportable.example.com. 300 IN A 192.0.2.42
```

Maintenant, si je me mets dans la peau de l'hébergeur qui gère plein de zones, est-ce Knot va tenir le coup ? Testons sur une petite machine, un Raspberry Pi <https://www.bortzmeyer.org/raspberry-pi.html> sous Arch Linux avec mille zones (c'est peu, mais c'est vraiment une petite machine). En cinq secondes, les mille zones sont chargées et Knot répond. 50 Mo de RAM sont utilisés dont 20 en résident. Et si on bombarde le serveur de requêtes avec queryperf <https://www.bortzmeyer.org/performances-serveur-dns.html> ? On atteint 3 800 requêtes par seconde, sans aucune perte.

Et si on compare avec BIND ? Un BIND 9.9.4-P1 sur le même Raspberry Pi consomme effectivement deux fois moins de mémoire mais ses performances avec queryperf sont bien moins bonnes, avec 1 000 requêtes par seconde (en débrayant la journalisation des requêtes refusées, qui est faite par défaut et ralentit beaucoup BIND). Rien d'étonnant (BIND a toujours été très lent dans tous les tests de performance, l'accent est mis par ses développeurs sur la richesse fonctionnelle).

Knot gère évidemment les trucs modernes : DNSSEC (si le fichier de zone est signé, il active automatiquement le mode DNSSEC), NSID (RFC 5001) et RRL (limitation du rythme des réponses). Pour NSID, on met :

```
system {
  identity "ns1.example.com"; # For the old CH TXT hostname.bind trick
  nsid "ns1.example.com";
}
```

Et on obtient le résultat attendu :

```
% dig +nsid -p 5353 @127.0.0.1 SOA example.com
...
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; NSID: 6e 73 31 2e 65 78 61 6d 70 6c 65 2e 63 6f 6d  (n) (s) (1) (.) (e) (x) (a) (m) (p) (l) (e) (.) (c) (o) (m
...
```

Knot dispose également évidemment de la fonction RRL ("*Response-Rate Limiting*") qui permet de limiter la contribution du serveur aux attaques par réflexion <https://www.bortzmeyer.org/jres-dos-2013.html>. Par exemple, on configure avec :

```
system {
  ...
  rate-limit 20;
}
```

Et on limite le serveur à 20 réponses par seconde pour la même question depuis la même adresse IP. Sur le graphique, on voit bien la différence : le premier pic correspond à une attaque par réflexion sans RRL (autant de paquets sortants qu'entrants), le second pic au cas où la RRL est activée (le nombre de paquets sortants chute).

Si vous voulez en apprendre plus sur Knot en français, vous pouvez lire l'article de ses auteurs à JRES <https://conf-ng.jres.org/2013/planning.html#article_160>.