The Design and Implementation of OpenBGPd

André Oppermann
<oppermann@networx.ch> <andre@freebsd.org>

Claudio Jeker
<jeker@networx.ch> <claudio@openbsd.org>

SWINOG-9
Berne, 29. September 2004

Why?

Why another BGP Daemon? What is wrong with the existing ones?

Cisco - closed source, arcane config style, bundled with massively overpriced and slow-CPU hardware.

Juniper - closed source, modern config style, bundled with expensive hardware.

Zebra/Quagga - GPL source, arcane config style, very inefficient and ugly implementation, chokes under high load, huge memory requirements.

Others - only basic implementations, not ready for production networks, or no longer maintained.

Design Goals

Very efficient implementation (memory usage and CPU load).

Highly scalable (number of peers and routes).

Easy to extend.

Intuitive and practical configuration language.

Security and safety by design.

BSD licensed.

Little History

André wants to write new BGP daemon since back in middle of 1999 in programming language ADA95.

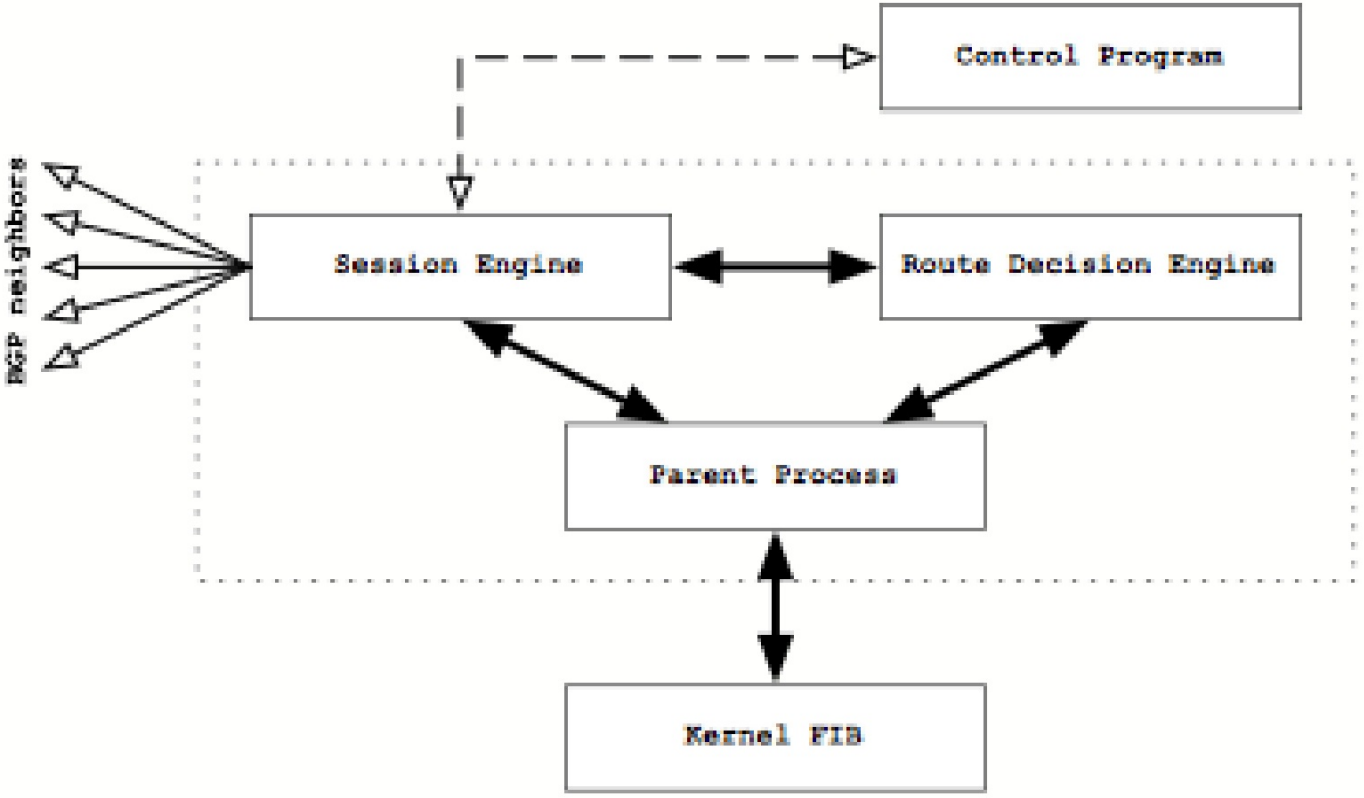
However he only ever gets to think in depth about the optimal design of the daemon but never writes a single line of code.

Henning Brauer comes up with a first code skeleton of the Session Engine in November 2003. He looks for people to help him to develop the code further.

Claudio and André decide to jump onto this and pull all the ideas and system design drafts out of the drawer and start coding.

General Design

OpenBGPd has four parts:



General Design

1. Session Engine handles all BGP sessions to neighbors and all timers (keepalive, hold and idle timer).

2. Route Decision Engine handles the BGP Routing Table (Prefixes and Paths) and generates the Updates.

3. Parent Process handles the configuration and interaction with the kernel routing table (FIB).

4. BGP Control program is the human interface and allows to look at and control of the bgpd processes.

General Design

SE, RDE and Parent Process is one program which forks into three processes at startup. Each process gets its own quantum of CPU time and is scheduled by the Operating System.

BGP Control program connects through a local UNIX domain socket to the Session Engine.

General Design

Advantages

Separation of functional parts of the BGP Daemon into separate processes.

The processes communicate through messages with each other. This is a object oriented approach.

No single part can monopolize the CPU. The Operating System makes sure each process get its quantum.

The Session Engine handles all timers itself. Even if the RDE is busy the SE ensures that Keepalives are generated perfectly in time to keep the BGP sessions up.

Route Decision Engine Design

The RDE makes heavy use of various forms of linked lists to avoid traversal of the entire table entirely (table walker on Cisco).

All paths and prefixes from each neighbor are linked together so when a peer closes we only have to travel the list to recompute the nexthops and to generate the updates in one go. Very fast and efficient.

The same for nexthops which might become unreachable.

Performance and Memory consumption

OpenBGPd uses about 11MB RAM for each full view (146k prefixes) it receives. The first one takes about 20MB.

It takes about 5 seconds to send the entire table to a new neighbor.

It takes about 1 second to purge a dropped neighbor from the table.

Due to the separated design it can handle essentially an unlimited number of flapping neighbors. In this case the RDE may lag a little bit but the SE keeps all other sessions up. Full connectivity is maintained.

In this case Instead of milliseconds it may take a second for normal update to get propagated. This is much faster than Cisco and Zebra/Quagga can handle this.

Performance and Memory consumption

Testserver at TIX in Zürich [194.42.48.111]:

```
# bgpctl show
Neighbor                AS    MsgRcvd    MsgSent  OutQ Up/Down  State/PrefixRcvd
194.42.48.37          4589      96012       2357     0 19:37:56 144914
194.42.48.28          8237      48359       2357     0 19:37:56  67136
194.42.48.39          6772      66275       2357     0 19:38:01 145837
194.42.48.32         15623      80597       2357     0 19:38:02 144864
194.42.48.15          9044     291396       2352     0 11:01:27 145035
194.42.48.14          9177      77786       2358     0 19:38:05 145827
194.42.48.6           8758      83688       2358     0 19:38:06 144844
194.42.48.2           8271      65429       2358     0 19:38:29 144881
194.42.48.47          3257     110024       2360     0 19:39:10 144755
194.42.48.18          1836      69865       2361     0 19:39:51 145675
62.48.4.4            65001      62218       2358     0 19:38:13 144884
64.185.97.128        23265      70855       2358     0 19:38:31 145839
194.42.48.16         13030     393509       2261     0 08:55:26 145139
194.42.48.0/24           0          0          0     0 Never    Active
```

Approx. 130MB RAM used for these 12 ½ full feeds, 1.8million prefixes.

Implemented RFC's

RFC 1771 A Border Gateway Protocol 4
RFC 1997 BGP Communities Attribute
RFC 2385 Protection of BGP Sessions via the TCP MD5 Signature Option
RFC 2858 Multiprotocol Extensions for BGP4
RFC 2918 Route Refresh Capability for BGP4 (broken by design)
RFC 3392 Capabilities Advertisement with BGP4
RFC 3765 NOPEER Community

Cool Features

Atomic configuration updates

When you reload the configuration file all changes will be applied. New neighbors are added and deleted one will be removed. After reload file and running configuration will be in guaranteed in sync. However if filters have changed on an existing session you have to manually clear the affected sessions for the filters to become effective on existing prefixes.

MRT file format table and update dumping

```
dump table "/var/log/bgpd-view-%h%m" 600
dump update in "/var/log/bgpd-updates-%h%m" 3600
```

Cool Features

Wildcard passive listening

You can specify a range of IP addresses from which any other router can connect. The remote AS will be taken from the Open Message. Neighbors are dynamically created from the template.

```
group "TIX" {
        announce none
        neighbor 194.42.48.0/24 {
            descr "TIX peer"
            passive
        }
}
```

Cool Features

BGPd can feed firewall tables (pf only at the moment). Useful for ALTQ queueing etc.

```
set pftable bla
```

Simply default announce policies: self, none, all, default-route

```
neighbor 10.10.10.1 {
    remote-as 65110
    announce self
}
```

Link state detection (for Ethernet). If link gets disconnected you don't have to wait for BGP session to time out. Cisco calls this fast external fallover.

Cool Features

Route collector knob to turn off best path selection. Useful if the machine only collects routes to dump them to MRT files

```
route-collector yes
```

TCPMD5 and IPSEC (static keys and IKE)

```
neighbor 10.10.10.1 {
    tcp md5sig password reallysecret
    ipsec ah out spi 55 sha1 <key> aes-128-cbc
    ipsec ah in spi 56 sha1 <key> aes-128-cbc
    ipsec ah ike
}
```

New Features already written

IPv6 support in SE and RDE. Only IPv6 NLRI over IPv6 sessions is supported. Only the global address will be used for next hops. (BTW: IPv6 is broken by design and it's implementations are damn ugly hacks).

Route Flap Dampening according to RFC 2439.

Transparent AS. Do not prepend local AS.

Prepend Neighbor command. Prepend the neighbors AS number n-times.

Additive Metrics (MED and local pref).

New Features planned

Much better and more extensive filter configuration. Good input from Markus Wild will be incorporated.

Automatic full mesh. This is an extension of the passive wildcard acceptor.

Full support for “soft reconfig” in- and outbound. The nice design of the RDE makes this relatively easy to implement.

Even better and more efficient memory allocator. We have many small fixed and variably sized objects.

SNMP integration for general and individual session information. Via bgpctl socket interface.

New Features planned

Central full route views. Instead of querying all routers you can see everything in one single place.

Firewall table feeding for FreeBSD IPFW.

Better interaction with other IGP routing protocols. (OpenIGP for sure)

Availability

OpenBGPd is maintained in the OpenBSD source tree. All necessary adjustments for other BSD's will be included there.

FreeBSD works except TCP-MD5 and IPSEC because of differences in the PFKEY interface between the operating systems. Work is under way to address that. After FreeBSD 5.3 Release OpenBGPd will in included in the FreeBSD ports collection in the network category.

NetBSD, Darwin (MacOS X), DragonFlyBSD: No status yet but should be the same as with FreeBSD.

Linux port needs some larger adjustments (Netlink API for Kernel routing table). Nobody has yet stepped forward to do it.

Stability

OpenBGPd is stable, reliable and ready for production use.

Practical use is limited smaller sites due to current restrictions in filter language.

It is used by a few dozen smaller AS's as primary BGP speaker already.

Related Projects

OpenISISd

by Claudio, in “I want to do this, where is the RFC?” mode, no code yet.

OpenIGP

André, Fresh IGP Protocol newly designed from ground up to be ultra-fast, ultra-scalable and conceptually non-complex (unlike OSPF and ISIS). No nasty hacks which add huge amounts of complexity! First parts written, about 10% done. Will talk more about this on next SWINOG!

That’s it. Any questions?