OpenOSPFD

Claudio Jeker <claudio@openbsd.org>

Introduction

- maintain routing table automatically
- choose "best" path
- recover from network failures (reroute)
- default free routing
- divide Internet into autonomous systems (AS)
  - same administrative domain
  - internal vs. external view
  - aggregate prefixes

Introduction

- EGP
  - Exterior Gateway Protocol
  - Exchange prefixes between AS
  - Features
    - Routing policies
    - scalable - 150k routes
  - BGP
- IGP
  - Interior Gateway Protocol
  - Routing table calculation inside an AS
  - Features
    - fast response to network changes
    - neighbor discovery
  - RIP, OSPF, IS-IS

Introduction - Routing Algorithms

- Distance Vector Algorithms
  - exchange of routing tables between neighbors
  - compare tables and choose best routes
  - redistribute again
  - Features
    - easy to implement
    - ability to express routing policies
  - Problems
    - slow propagation of changes
    - count to infinity
      - Path distance vector algorithm does not suffer from this problem
  - Examples
    - RIP, BGP (path distance vector)

Introduction - Routing Algorithms

- Link-State Algorithms
  - every router sends out his link-states
  - all router keep a database of all link-states
  - calculates shortest path
  - Features
    - good convergence properties
    - automatic neighbor discovery
  - Problems
    - complex because the database needs to be in sync
  - Examples
    - IS-IS, OSPF

OSPF - Features

- Most used IGP
- IPv4 only -- OSPFv3 implements IPv6
- Link State Protocol
- Implemented as own IP protocol (not TCP or UDP)
- Router discovery via multicast
- Support for areas to divide network
- IETF designed
  - super complex and badly documented protocol

OSPF - Link-State Database

- 5 different Link-State announcements
  - router LSA
  - network LSA
  - summary LSA for networks
  - summary LSA for AS border routers
  - AS external LSA
- All LS databases in area need to be in sync
- Routing table is generated by a shortest-path-first calculation using router and network LSA.
- remaining LSA types are evaluated and added in a second step

OSPF - Router Discovery

- Hello Packets sent all 10 seconds
- sent via multicast
- bidirectional communication enforced
  - a list of all routers from where a hello was received lately included in hello

- Designated Router (DR)
  - only on broadcast networks
  - reduces the amount of packets sent
  - DR does flooding and retransmission on behalf of all other routers
  - Backup designated router in case DR fails
  - complex and error prone (imprecise RFC)

OSPF - Database Synchronisation

- Initial synchronisation
  - exchange of database description packets in a way like tftp
  - request of LSA entries that are newer
  - receive of requested LSA
  - retransmit LS requests after a time-out (packet loss)

- Flooding
  - flooding keeps all LS DBs in sync
  - every router resends new LS updates
  - every LS update needs to be acknowledged
  - retransmit LS updates after a time-out (packet loss)

OSPF - Areas

- Divide large network into smaller areas
- every area is connected to the backbone area
- if no direct link is available a virtual link is required
- additional duties for area border routers
  - originating summary LSA into connected areas

- network needs to be designed for areas!
- in most cases not needed

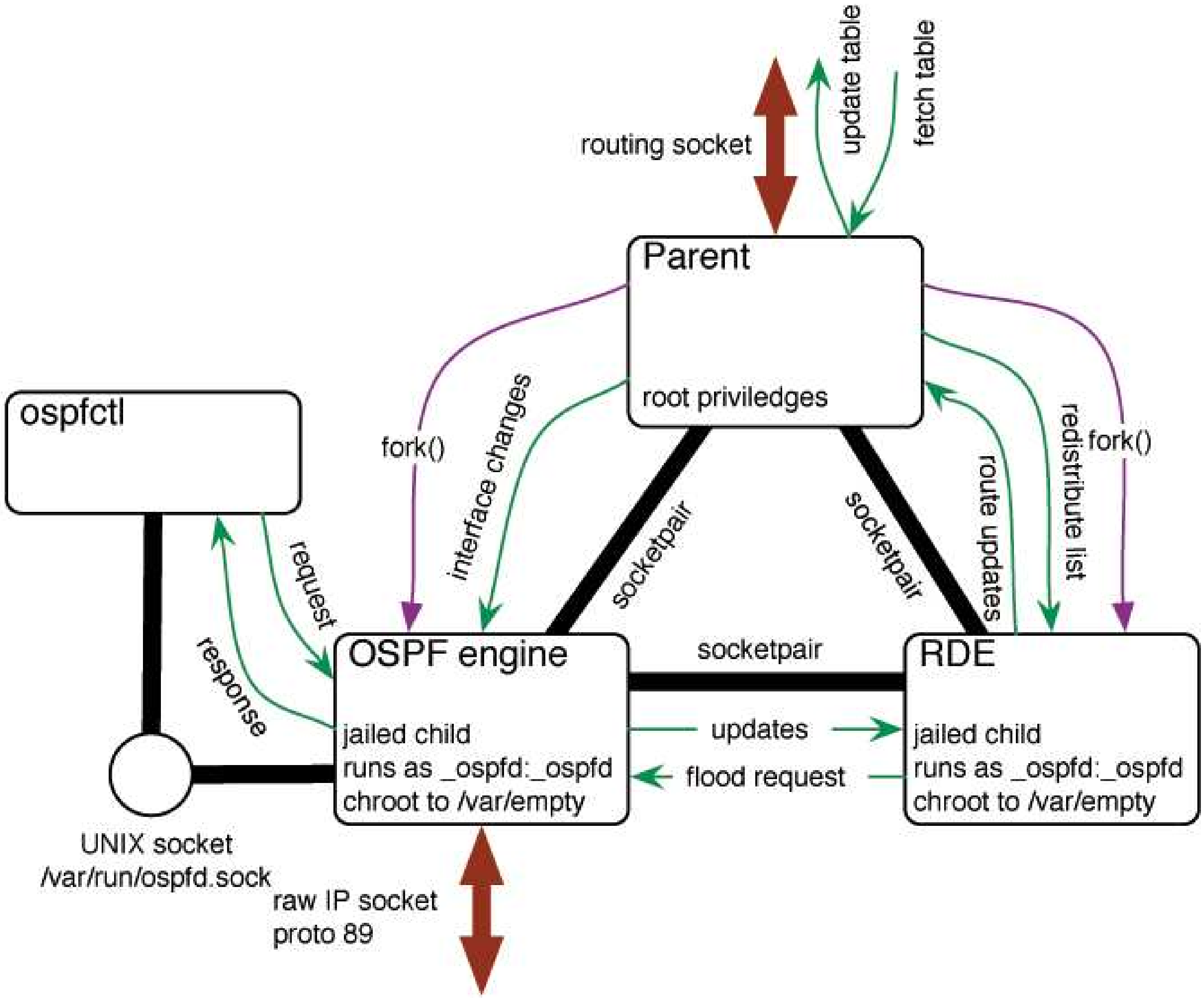
Design - Overview

- Major points: secure, stable, efficient
- steal as much as possible

- "stolen" from OpenBGPD
  - 3 processes
  - privilege separation
  - buffer management
  - imsg framework for internal messaging
  - kroute - routing table management

- differences
  - raw IP packets instead of TCP session
  - more concurrent timers and finite state machines
  - use of libevent instead of poll

Overview



routing socket
update table
fetch table
Parent
root priviledges
fork()
interface changes
socketpair
socketpair
route updates
redistribute list
fork()
ospfctl
request
response
OSPF engine
jailed child
runs as _ospfd:_ospfd
chroot to /var/empty
socketpair
updates
flood request
RDE
jailed child
runs as _ospfd:_ospfd
chroot to /var/empty
UNIX socket
/var/run/ospfd.sock
raw IP socket
proto 89

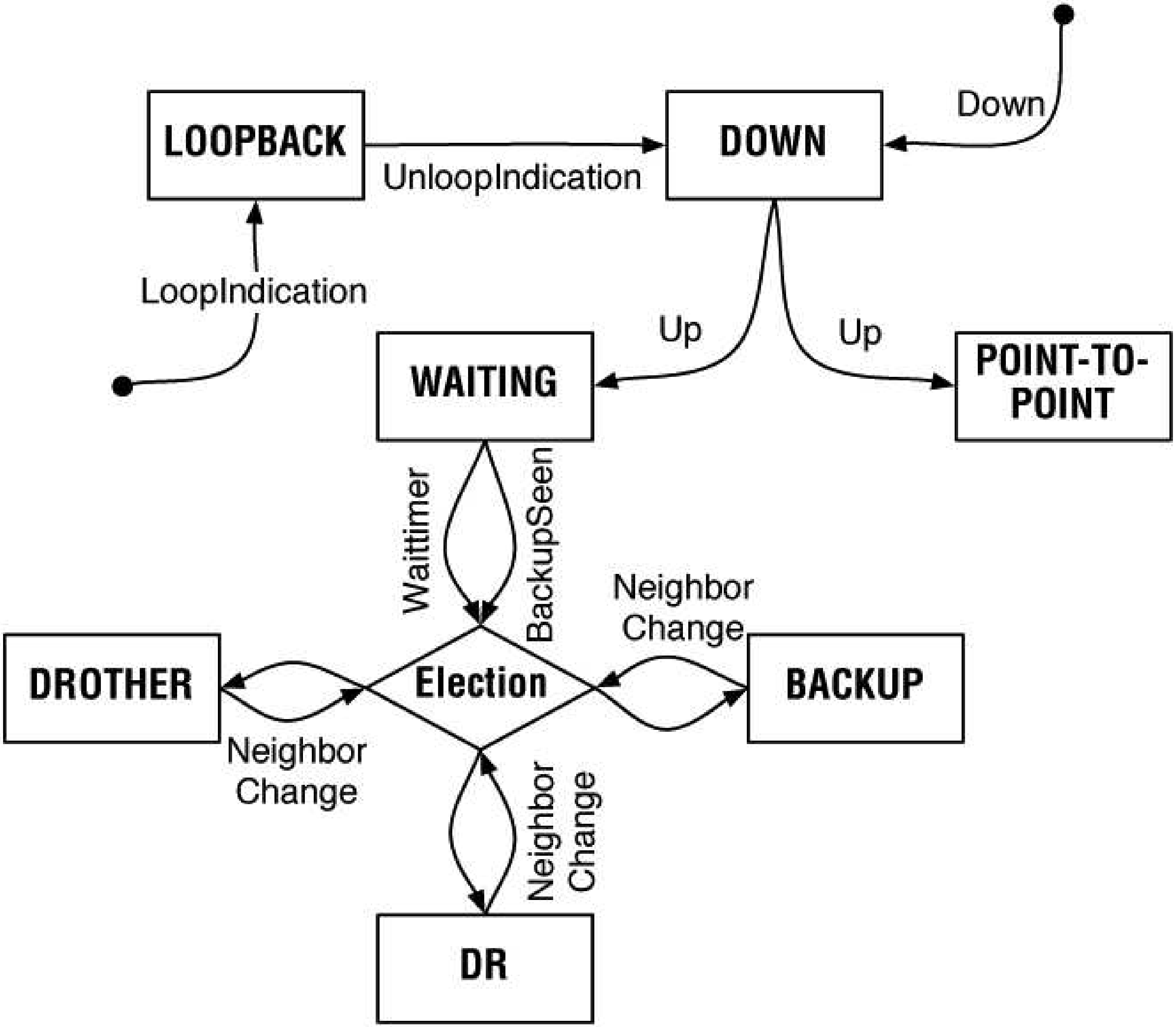
Parent Process

- Responsible for getting the routes into the kernel
- Tracks interface link states
- Maintains its own copy of the kernel routing table
- Fetches the kernel routing table and interface list on startup

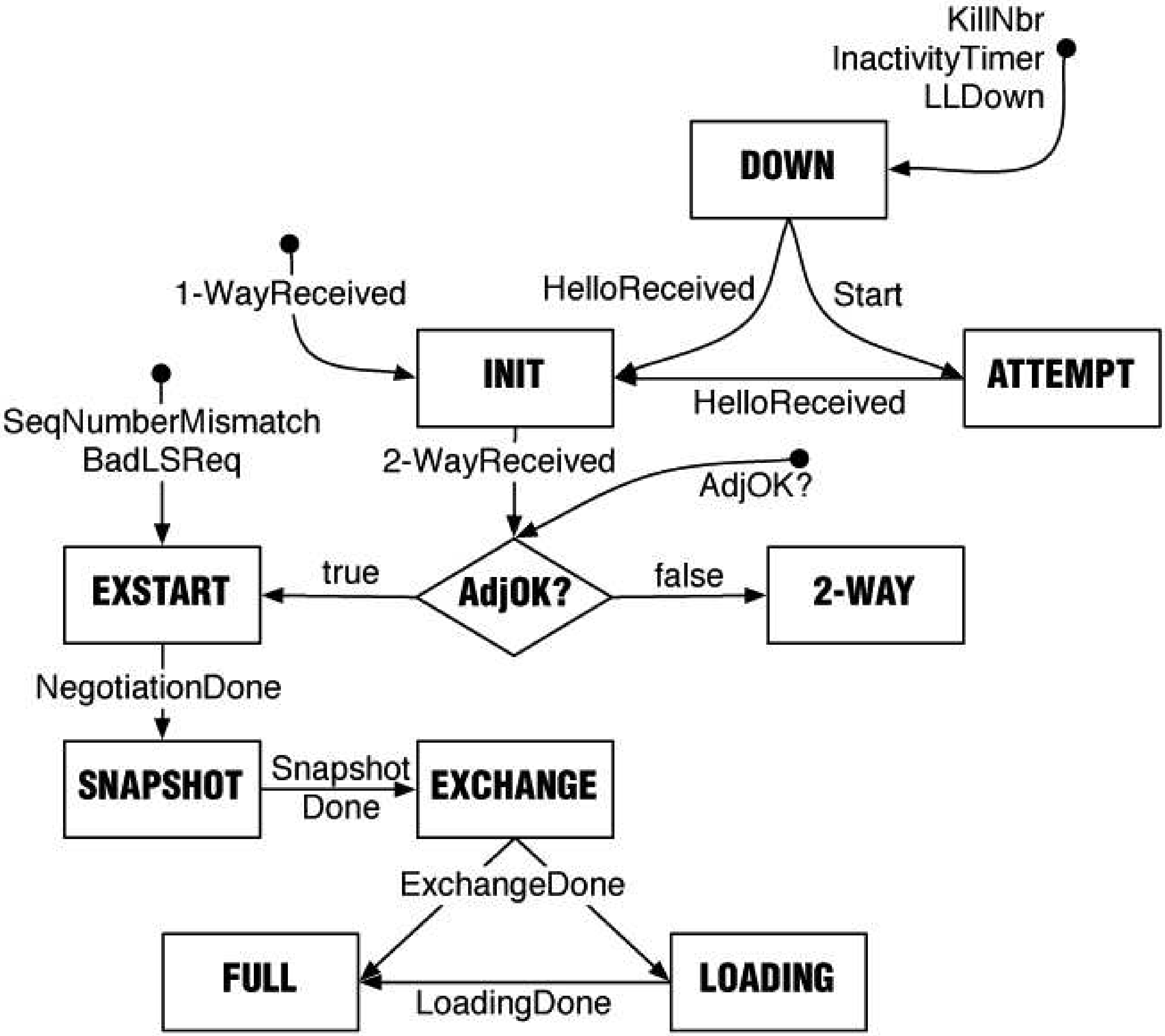
OSPF Engine

- Listens on the raw IP socket
- Verifies and processes the packets
- Interface finite state machine
  - DR / BDR election process
- Neighbor finite state machine
- Initial Database Exchange
- Reliable flooding of LS updates (retransmits)

OSPF Engine - Interface FSM



OSPF Engine - Neighbor FSM



RDE

- stores LS database
- calculates SPF tree
- informs parent process about routing table changes
- redistribution of networks (ASBR)
- summary LSA generation if ABR

ospfctl

- shows current status of ospfd
- Important commands:
- ospfctl show neighbor

```
cjeker@diavolezza:~> ospfctl show neighbor
ID              Pri State             DeadTime  Address         Interface
0.0.0.1         1   INIT/DROTHER      00:00:33  62.48.4.38      fxp0
62.48.4.5       1   FULL/DR           00:00:30  62.48.4.5       fxp0
62.48.4.3       1   FULL/BACKUP       00:00:30  62.48.4.3       fxp0
```

ospfctl

- ospfctl show interface

```
cjeker@diavolezza:~> ospfctl show interface

Interface fxp0 is 2, line protocol is UP
  Internet address 62.48.4.4/24, Area 0.0.0.0
  Router ID 62.48.4.4, network type BROADCAST, cost: 10
  Transmit delay is 1 sec(s), state DROTHER, priority 1
  Designated Router (ID) 62.48.4.5, interface address 62.48.4.5
  Backup Designated Router (ID) 62.48.4.3, interface address 62.48.4.3
  Timer intervals configured, hello 10, dead 40, wait 40, retransmit 5
    Hello timer due in 00:00:04
  Neighbor count is 3, adjacent neighbor count is 2
```

ospfctl

- ospfctl show database

```
cjeker@diavolezza:~> ospfctl show database

                Router Link States (Area 0.0.0.0)

Link ID         Adv Router      Age  Seq#       Checksum
0.0.0.1         0.0.0.1         213  0x80000002 0x7d25
62.48.4.3       62.48.4.3       292  0x80000004 0xadc1
62.48.4.4       62.48.4.4       296  0x80000004 0xabc0
62.48.4.5       62.48.4.5       293  0x80000002 0x2f43

                Net Link States (Area 0.0.0.0)

Link ID         Adv Router      Age  Seq#       Checksum
62.48.4.5       62.48.4.5       217  0x80000004 0x8774
```

ospfctl

- ospfctl show database - detailed output

```
cjeker@diavolezza:~> ospfctl show database router

                Router Link States (Area 0.0.0.0)

LS age: 269
Options: *|*|-|-|-|-|E|*
LS Type: Router
Link State ID: 0.0.0.1
Advertising Router: 0.0.0.1
LS Seq Number: 0x80000002
Checksum: 0x7d25
Length: 48
Flags: *|*|*|*|*|-|-|-
Number of Links: 2

    Link connected to: Stub Network
    Link ID (Network ID): 192.168.5.0
    Link Data (Network Mask): 255.255.255.0
    Metric: 12

    Link connected to: Transit Network
    Link ID (Designated Router address): 62.48.4.5
    Link Data (Router Interface address): 62.48.4.38
    Metric: 20
```

Usage - Configuration

```
# global configuration
router-id 10.28.4.65

# route redistribution
redistribute connected
redistribute static

# areas
area 0.0.0.0 {
        interface lo1

        interface em0 {
                metric          10
                auth-type       crypt
                auth-md-keyid   1
                auth-md         1        "sdf&*di12"
        }

        interface vlan202 {
                metric          50
                auth-type       crypt
                auth-md-keyid   5
                auth-md         5        "Flkjds/8id@"
        }
}
```

Usage - Carp and ospfd

- carp - Common Address Redundancy Protocol
- ospfd - routing daemon using network redundancy for re-routing

  conflicts!
  ... but very powerful if used correctly
- Impossible to run OSPF on a carp interface
- Instead use carp to connect a LAN with servers to an OSPF cloud
  - more than one ospf router
  - default gateway on servers is carped and does not change
- Use a "passive" carp interface and multiple ethernet interfaces to connect router to the OSPF cloud; link-state of carp interface is tracked
  - route in the OSPF cloud will always point to the active carp interface

Usage - interface metric

- "metric does not work"
  - high metric on a interface seems to be ignored
- OSPF calculates path through the network
  - reverse path may have a different cost
- On broadcast networks only the metric into the network is added
  - to control incoming traffic outgoing interfaces need to be adjusted

Future plans

- Config reload
- Even better carp support
  - Making announcements dependent on interface link state
- Interface group support
  - Mostly for dynamic clonable interfaces
  - Makes it possible to configure interfaces that are not present on startup

More future plans

- "redistribute bgp" and especially dependant on route label
- Possibility to add aggregation networks for areas
  - Only needed on ABRs.
  - Telling to add 10.1.128.0/19 instead of 10.1.129.64/28 as soon as an area gets active.
- Conversion table of route labels to AS-ext route ID tags and especially back
- Finally commit all M I have in my trees

Evil future plans

- Make it possible to determine if all routers are in sync
- Make it possible to create a network graph from the LS DB
  - creates nice coloured network graphs for web pages
- Add a way to calculate the rib for any router in the network
  - The LS DB includes all necessary information
  - perfect for monitoring systems
- After all is done I may perhaps start on OSPFv3 aka IPv6 support

Thanks

- Esben Norby <norby@openbsd.org>, who started with the OpenOSPFD project and implemented large parts of it.
- Andre Oppermann <oppermann@networx.ch> and Internet Business Solutions AG for "sponsoring" my work on OpenOSPFD.
- Henning Brauer <henning@openbsd.org> and the rest of the OpenBSD gang for a lot of code to steal from.