FRRouting workshop

Christian Hopps
David *'equinox'* Lamparter

netdevconf 0x18 · San Francisco, US · 2024-07-16

meetup?

there are some FRRouting people around the Bay Area who can't attend netdevconf for whatever reason poor excuses! ;)

we may have an evening meetup this week
(in order of likelyhood: Thursday > today > tomorrow)

Outline

1) release summary
2) mgmtd/YANG
3) debugs

PIC & multistep/recursive FIB got dropped, sorry

Release Summary: 9.1

- accept default routes as recursive resolvers
- VLAN, ECN, DSCP in pbrd

for the sake of brevity, only “general” FRR changes listed here
whole bunch of protocol specific updates omitted

Release Summary: 10.0

- split configs (zebra.conf) gone, frr.conf wins
- `noprefixroute` a source of unending pain
  - NetworkManager on routers, really?
- “L>” local routes
  - currently internally synthesized, not from kernel

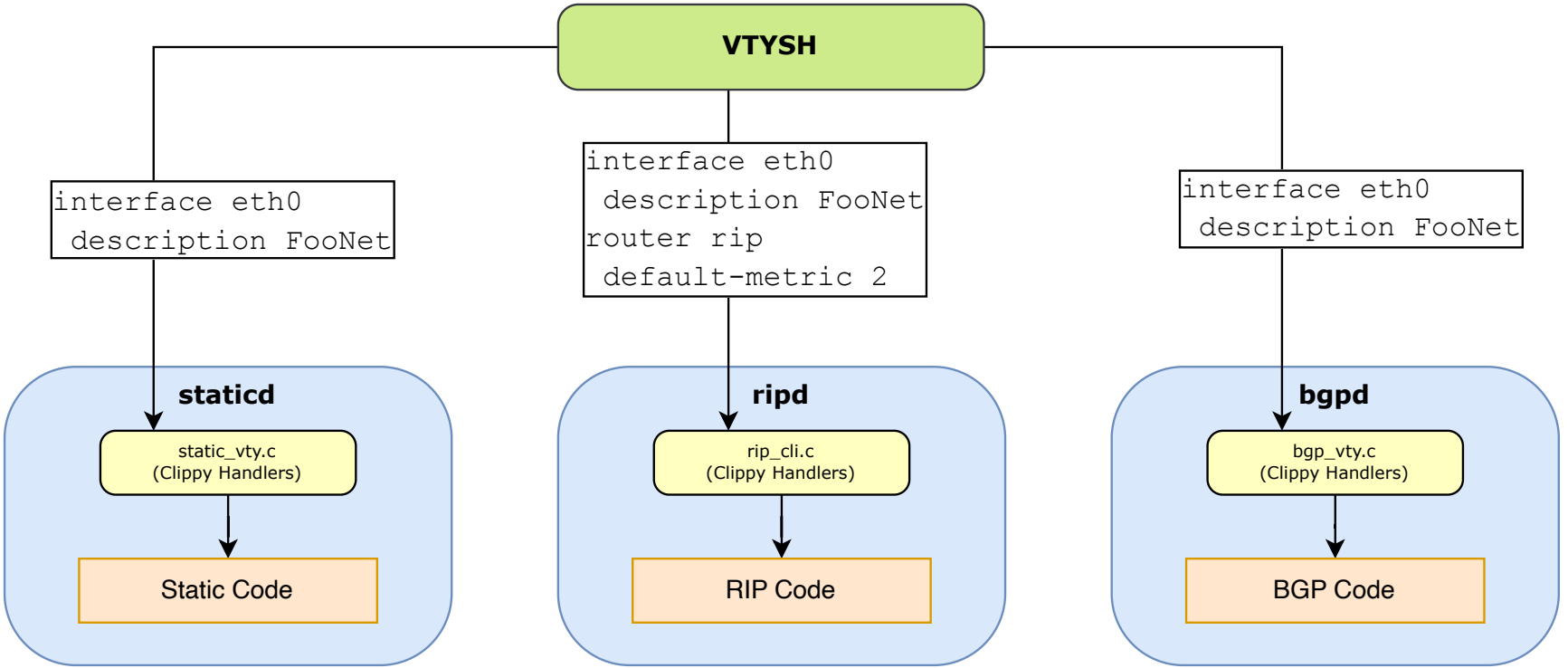
General developments

- IPv6 multicast routing (PIM) maturing, no longer considered “experimental” in FRR
- dataplane backend seeing a lot of ongoing development, particularly for multi-step FIB capable hardware

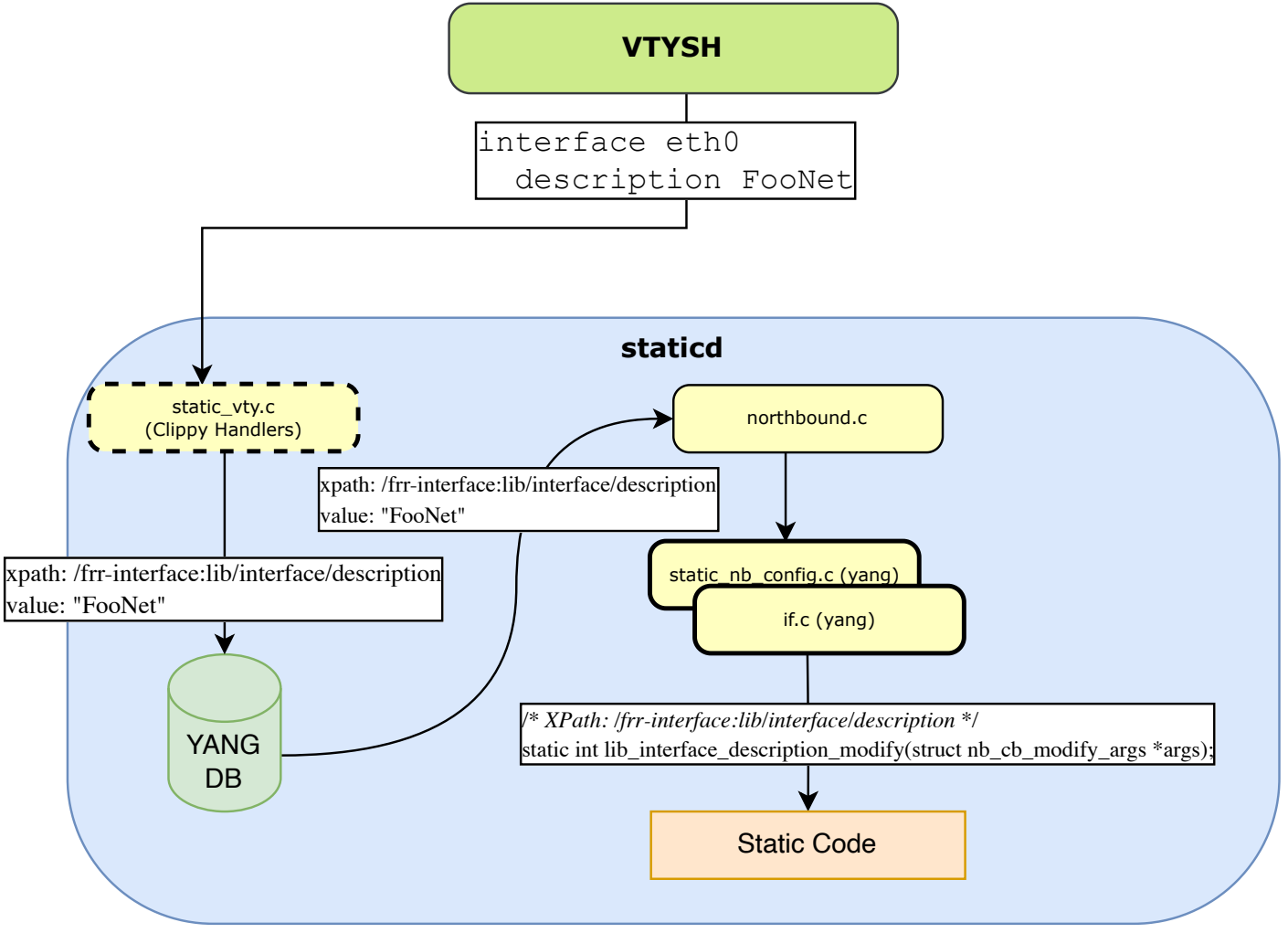
Christian Hopps
LabN Consulting, LLC

FRR New mgmtd (management daemon)

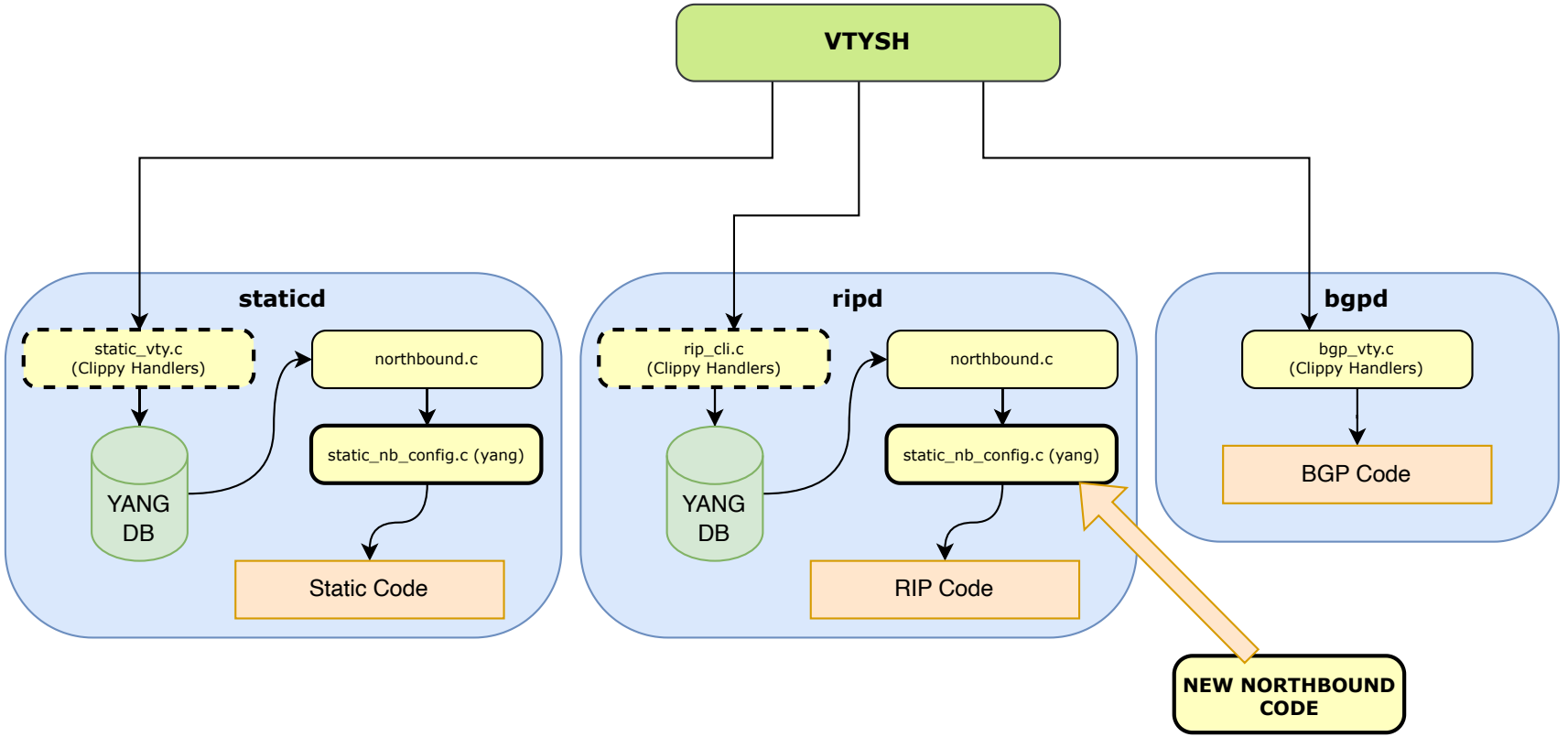
Original CLI Design



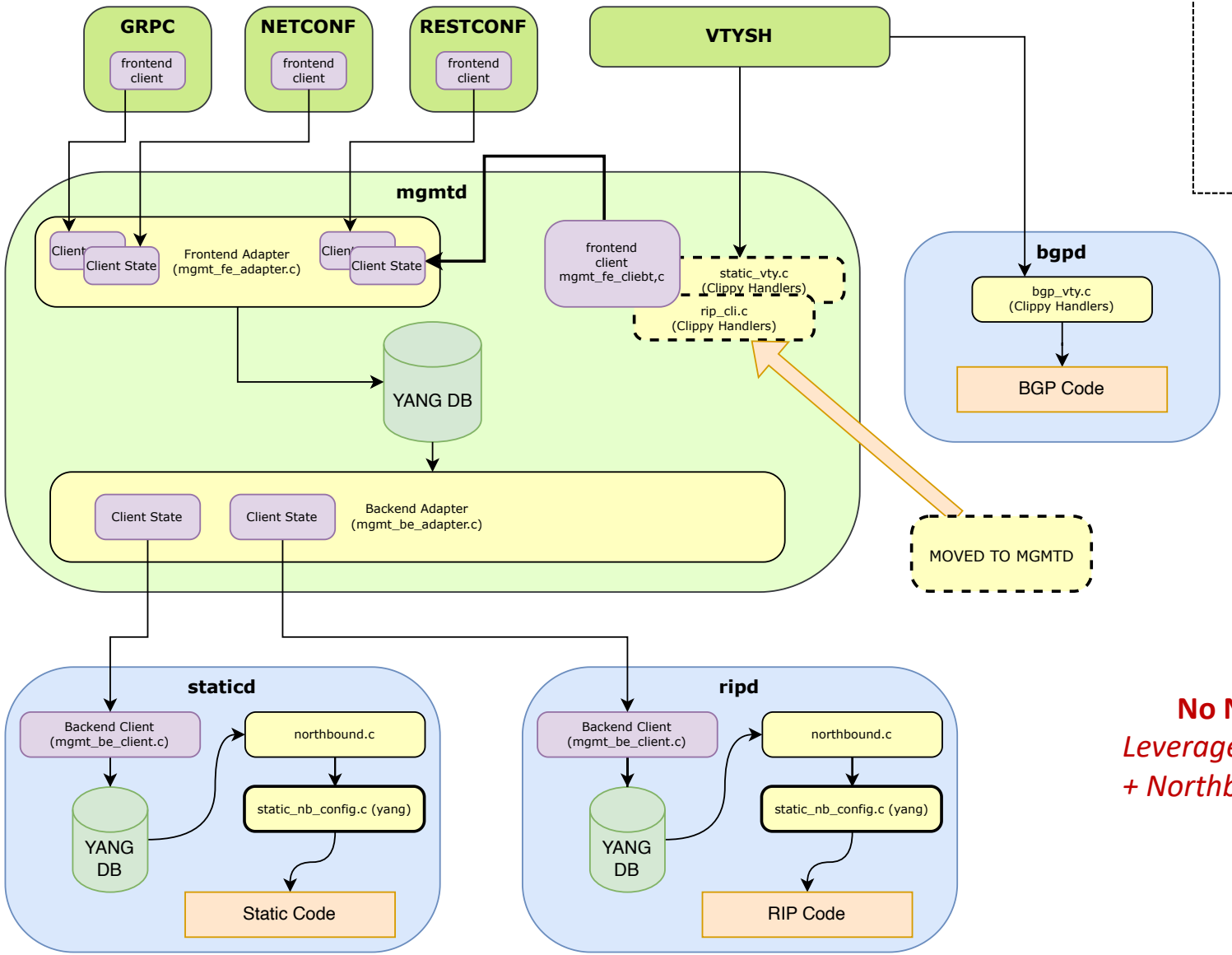
Northbound Conversion



Northbound Conversion

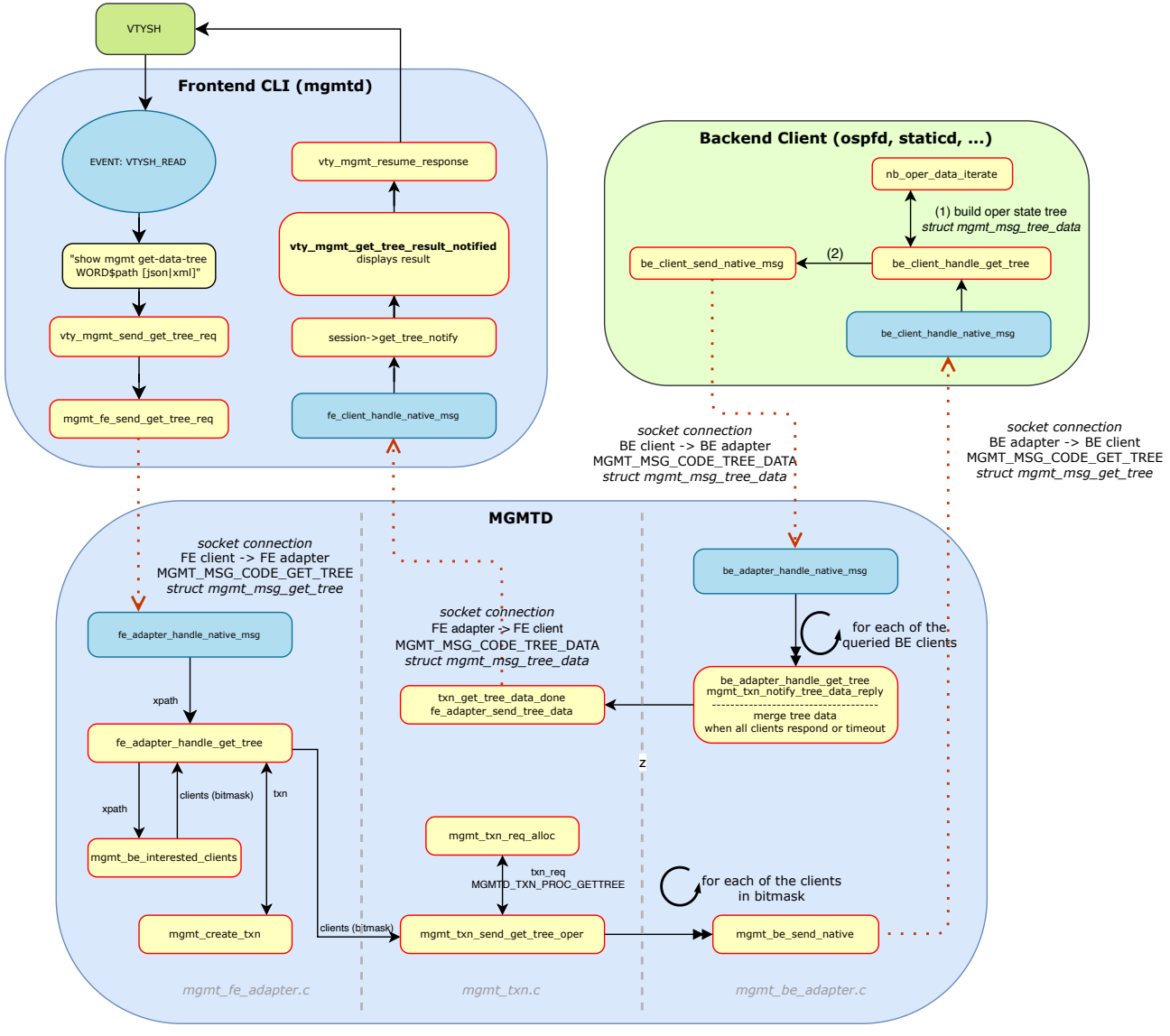


mgmtd



No New Daemon Code

Leverage Existing CLI handlers
+ Northbound YANG Callbacks



Detail of Oper-State Query

XPath used for YANG (config) routing

Zebra Config Routing

```
static const char *const zebra_config_xpaths[] = {
        "/frr-affinity-map:lib",
        "/frr-filter:lib",
        "/frr-interface:lib",
        "/frr-route-map:lib",
        "/frr-vrf:lib",
        "/frr-zebra:zebra",
        NULL,
};
```

RIPd Config Routing

```
static const char *const ripd_config_xpaths[] = {
        "/frr-filter:lib",
        "/frr-interface:lib/interface",
        "/frr-ripd:ripd",
        "/frr-route-map:lib",
        "/frr-vrf:lib",
        "/ietf-key-chain:key-chains",
        NULL,
};
```

XPath used for YANG (oper) routing

Zebra Oper-State Routing

```
static const char *const zebra_oper_xpaths[] = {
        "/frr-interface:lib/interface",
        "/frr-vrf:lib/vrf/frr-zebra:zebra",
        "/frr-zebra:zebra",
        NULL,
};
```

RIPd Oper-State Routing

```
static const char *const ripd_oper_xpaths[] = {
        "/frr-ripd:ripd",
        "/ietf-key-chain:key-chains",
        NULL,
};
```

XPath used for YANG (RPC) routing

Zebra RPC Routing

None

RIPd RPC Routing

```
static const char *const ripd_rpc_xpaths[] = {
        "/frr-ripd",
        NULL,
};
```

Converted to mgmtd

- lib/distribute
- lib/filter
- lib/if_rmap
- lib/keychain
- lib/routemap
- lib/affinitymap
- lib/if
- lib/vrf
- ripd
- ripngd
- staticd
- zebra

Converted to NB

- bfdd
- pathd
- pbrd
- Pimd

- Converted With Issues
  - eigrp
  - isisd

Remaining for Northbound Conversion

- babel
- bgpd
- ldpd
- lib/event
- lib/log_vty
- lib/nexthop_group
- lib/zlog_5424_cli
- nhrpd
- ospfd
- ospf6d
- pceplib
- qdb
- sharpd
- vrrpd

Docs

- Northbound Conversion
  - https://docs.frrouting.org/projects/dev-guide/en/latest/northbound/northbound.html
- mgmtd conversion
  - https://docs.frrouting.org/projects/dev-guide/en/latest/mgmtd-dev.html

reworking debug infrastructure

Too many ~~cooks~~ debugs - existing

let’s tell a cautionary tale…

“How to make your debug logs as messy as possible in 5 easy steps”

Too many ~~cooks~~ debugs - existing

```
if (my_debug)
    zlog_debug(“foo”);
```

used: almost everywhere

https://github.com/FRRouting/frr/blob/66de92184fe52dd8dbe237e6c34b5457bf465fd3/bgpd/bgp_route.c#L497-L499

Too many ~~cooks~~ debugs - existing

```
#define foo_debug( ... ) \
    if (x) zlog_debug(__VA_ARGS__);
```

used: scattershot

https://github.com/FRRouting/frr/blob/66de92184fe52dd8dbe237e6c34b5457bf465fd3/isisd/isis_ldp_sync.h#L13-L17

Too many ~~cooks~~ debugs - existing

```
struct debug flag;
DEBUGD(&flag, “foo”);
```

used: primarily pbrd, vrrpd
bonus: DEBUGI, DEBUGW, DEBUGE exist
remember the ampersand, it’ll come back to bite us
https://github.com/FRRouting/frr/blob/66de92184fe52dd8dbe237e6c34b5457bf465fd3/vrrpd/vrrp_ndisc.c#L163-L166

Too many ~~cooks~~ debugs - existing

```
.log_callback = &log_wrapper,
```

breaks message index (only one “static struct xref”)

used: pcep library, libsnmp

https://github.com/FRRouting/frr/blob/66de92184fe52dd8dbe237e6c34b5457bf465fd3/pathd/path_pcep_lib.c#L580-L586

Too many ~~cooks~~ debugs - existing

```
void my_log(…) { zlog_debug(…) };
```

breaks message index (only one “static struct xref”)

used: ldpd (previously – fixed)

https://github.com/FRRouting/frr/commit/95a737ed1bf7f89cb8e5367ef8e8e66a800084d7#...

PRs

PR #12272

```
DEFINE_DEBUGFLAG(flag, “cli”, “…”);
dbg(flag, “foo”);
```

note the absence of an ampersand…
bulk conversion (coccinelle) split off into separate PR

PR #15646

restructures `struct debug` setup to include CLI commands

adjusts existing users of DEBUGD

PR #15722

```
zlog_debug_cond(flag, “foo”);
```

only some bits in ospf6d changed over

Why does any of this matter?

- Code consistency (duh)
- message indexing

  FRR has had a zlog index (“xrefs”) for a while analogous to `CONFIG_PRINTK_INDEX`
  https://lwn.net/Articles/857148/

```
"G00A3-MV858": [
  {
    "args": "peer->host, size, type == 128 ? \"ROUTE-REFRESH\" : bgp_type
    "binary": "bgpd/bgpd",
    "file": "bgpd/bgp_io.c",
    "fmtstring": "%s bad message length - %d for %s",
    "func": "validate_header",
    "line": 613,
    "priority": 7,
    "type": "logmsg"
  }
],
```


Why does any of this matter?

- `debug unique-id 12345-ABCDE backtrace`

  quite useful since operators don’t want to unnecessarily restart FRR (or risk hanging it by attaching gdb)

Desired Features

- en-/disabling individual debug messages (reduce CPU cost while debugging)
- list in message index which debug flag controls a particular log message

Desired Features

- hit counters for debug messages regardless of enabled status (to extract statistics which code actually gets executed)
  …poor person’s perf counters

General Idea

- call into logging code and do the condition check there (after counting, and with possible additional selectivity)

Naively

```
#define dbg(flag, ...)                          \
    static struct xref … &flag …; \
    zlog_debug(&flag, __VA_ARGS__);
```

https://github.com/FRRouting/frr/pull/12272/commits/8ad7cba4c3e08230d5da67120f61023666feb399#…

… turns out to be a bit costly

- va_start() in zlog_debug turns out to be surprisingly expensive
  - unconditionally copies all registers, including FP/SIMD (a float could be getting printed…)
- flag check is now after the va_start(), hit taken when debug is disabled

Cheaper

```
#define dbg(flag, ... )                    \
    static struct xref … &flag …; \
    if (zlog_test(&flag))                  \
        zlog_debug(&flag, …);
```

- 2 function calls instead of one, but the second one is skipped in the common case (debug flag disabled)

Boss fight

- FRR uses “parametrized” debug flags in a few places

```
if (flag && prefix_match(debug_p, p))
    zlog_debug(“foo”);
```

```
CLI: debug bgp updates prefix A.B.C.D/M
```

https://github.com/FRRouting/frr/blob/66de92184fe52dd8dbe237e6c34b5457bf465fd3/bgpd/bgp_route.c#L2362-L2365

About that ampersand…

- prevents ## token pasting to construct/reference additional symbols
  - can’t call static inline helper function
  - really need it for parametrization support
- makes accessing struct members awkward

Last but not least

- counters need to be MT-safe
- rather not take the cost of atomics or a lock
- rseq?

takeaways

- & shouldn’t be in macro invocation sites
- don’t underestimate the cost of va_args

- cut down to one style before it gets this bad

Questions / general discussion