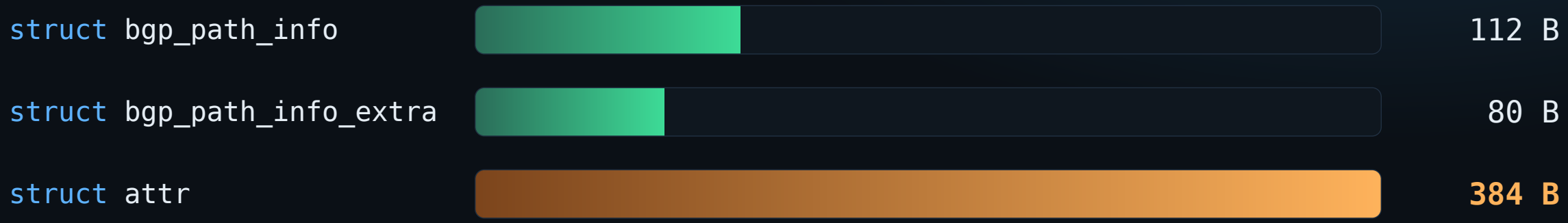
A Dietary Plan for BGP

Donatas Abraitis

NONOG-8 / NIX-2026

9th of September 2026

“hungry” BGP structures



`attr` should hold real BGP attributes. In practice it became a place to stash all sorts of things — and on a full Internet table that's **~360 MB just for attributes**.

Why do we need a diet plan?

Performance

CPUs fetch memory in 64-byte cache lines. Best-path processing walks these structures across millions of paths — fewer cache lines means less pressure and fewer misses.

Smaller memory / CPU footprints

A few dozen bytes is boring — until you multiply it by a full Internet table. Then it's hundreds of megabytes.

Reduce complexity

When EVPN, SRv6 and every other feature live inline, everything becomes coupled to everything else.

Increase maintainability

The structure grows, the dependencies grow, and reasoning about the code gets progressively harder.

Plans

01

Compression

Stop using more storage than the data actually requires.

02

De-commoning

What only a few objects use shouldn't live in a structure everybody allocates.

03

Right home

Locally derived state doesn't belong in an on-the-wire attribute structure.

PLAN 01

Compression

```
struct attr {
-       uint32_t rmap_change_flags;   /* 11 flags — 4 B is generous */
+       uint16_t rmap_change_flags;   /* -2 B */
-       uint16_t encap_tunneltype;
+       uint8_t  encap_tunneltype;     /* -1 B */
-       uint8_t  router_flag;
-       uint8_t  sticky;
-       uint8_t  default_gw;
+       uint8_t  evpn_flags;           /* -2 B, packed */
};
```

−5 bytes

Not exciting on its own. The point is that it accumulates: **a byte here, two bytes there**, across millions of routes.

PLAN 02

De-commoning

```
struct attr {                                   /* offset  size */
-       struct bgp_route_evpn      evpn_overlay;   /*   208    36 */
+       struct bgp_route_evpn *    evpn_overlay;   /*   208     8 */
};
```

−28 bytes

36 bytes inline in **every** attr, EVPN route or not — a full v4/v6 unicast table carries that **millions of times** without ever touching it. Now it's an 8-byte pointer, allocated only when the route really needs it.

PLAN 02 · THE BIG ONE

De-commoning → allocated on demand

```
struct attr_extra {
        long unsigned int          refcnt;              /*     0     8 */
        struct bgp_nhc *           nhc;                 /*     8     8 */
        struct bgp_route_evpn *    evpn_overlay;        /*    16     8 */
        uint64_t                   aigp_metric;         /*    24     8 */
        struct bgp_ls_attr *       ls_attr;             /*    32     8 */
        struct bgp_attr_srv6_vpn * srv6_vpn;            /*    40     8 */
        struct bgp_attr_srv6_l3service * srv6_l3service; /*    48     8 */
        enum pta_type              pmsi_tnl_type;       /*    56     4 */
        struct in6_addr            tunn_id;             /*    60    16 */
        uint32_t                   otc;                 /*    76     4 */
        struct ecommunity *        ipv6_ecommunity;     /*    80     8 */
};
```

−88 bytes from the common case

OTC, next-hop characteristics, AIGP, SRv6, EVPN fields — **none of them relevant to an ordinary unicast path**. Yet every route paid for them.

PLAN 03

Right home

INTERNED — HASHED & SHARED

```
struct attr_extra {
        …
-       uint32_t srte_color;
};
```

PER-PATH STATE — BY DESIGN

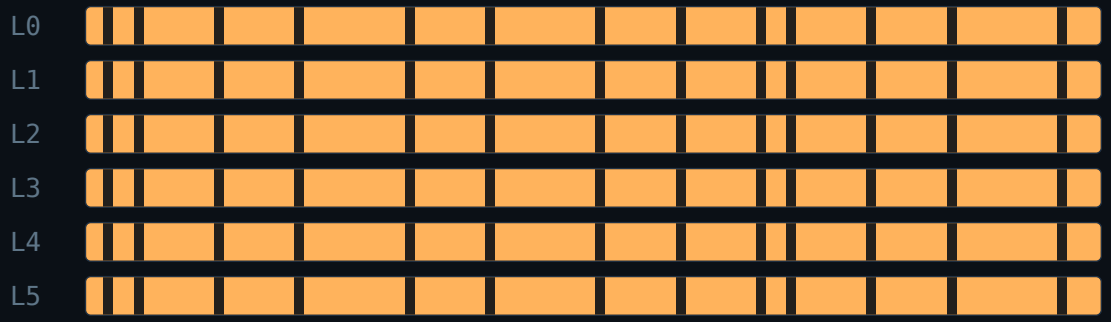
```
struct bgp_path_info_extra {
        …
+       uint32_t srte_color;
};
```

srte_color is not a BGP attribute — it's derived from an extended community. And **attr is interned**: hashed and shared across every path carrying the same attributes. This change saves no memory at all — that's not the point.

Final weight

```
- struct attr     /* size: 384, cachelines: 6 */
+ struct attr     /* size: 248, cachelines: 4 */
```

BEFORE — **384 B / 6 cache lines**



AFTER — **248 B / 4 cache lines**

−136 B per distinct attribute set (384 → 248)

6 → 4 cache lines — two fewer fetches every time best-path touches a path

~130 MB saved on a full Internet table

A plain unicast route should **not** be paying for features it never uses.

Thank you

Donatas Abraitis · ton31337