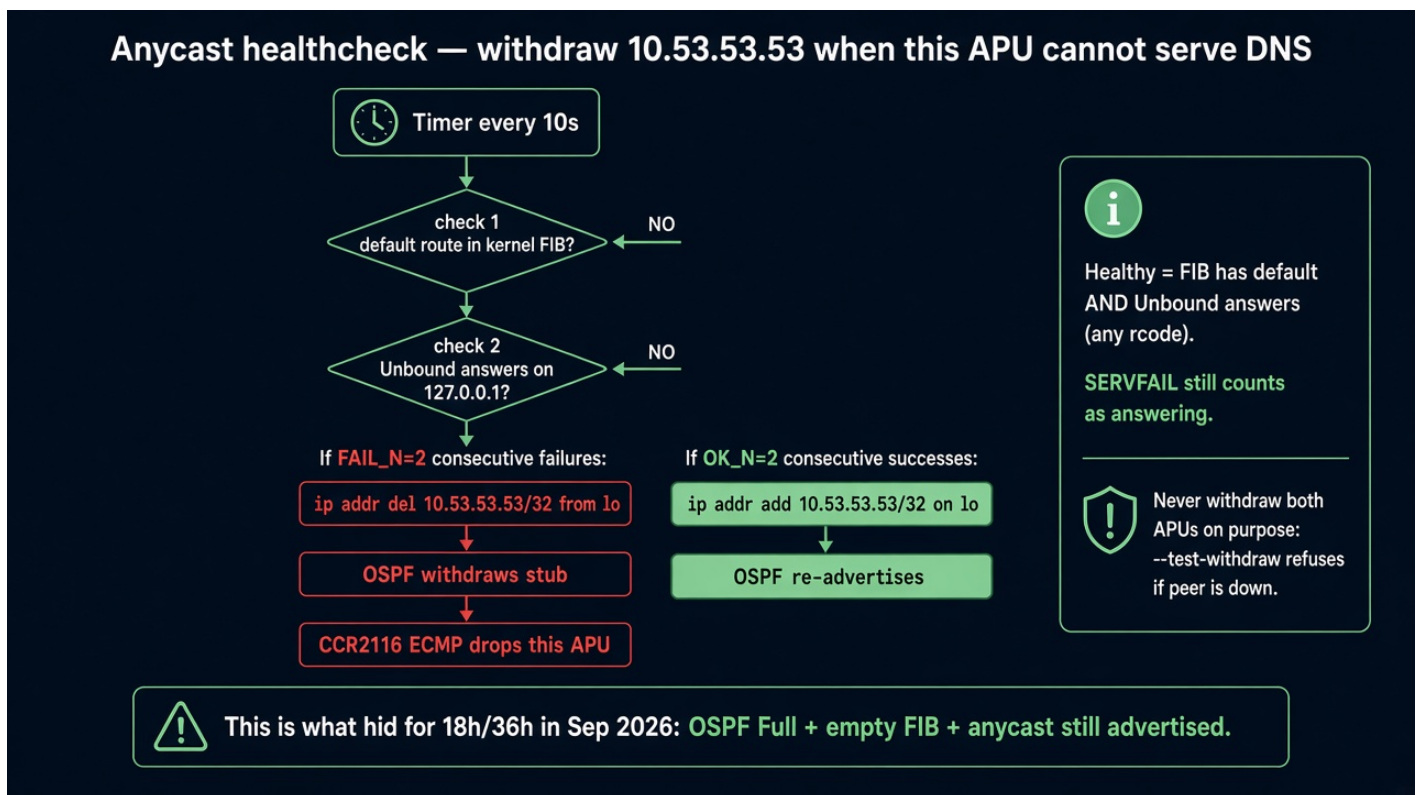


Anycast self-heal

Anycast self-heal

FRR on each APU advertises `10.53.53.53/32` as a stub of `lo`. The core therefore load-shares onto **whatever is currently advertised**, not “whatever has OSPF Full”. If an APU cannot actually serve DNS, it must take that address off `lo` itself. That is `anycast-healthcheck.sh`, run every 10 s by `anycast-healthcheck.timer`.

Without this, OSPF Full + an empty kernel FIB still looks like a live resolver. That is exactly the Sep 2026 blackholes.



What healthy means

Healthy is **both**:

1. A **default route exists in the main kernel FIB** (`ip -4 route show default` matches `^default`). An empty FIB is what an OSPF-Full blackhole looks like from the box.
2. **Unbound answers on 127.0.0.1** — `dig +time=1 +tries=1 @127.0.0.1 example.com A` prints a `status:` line. **Any rcode counts**, including SERVFAIL. The resolver is up; upstream may be down. That condition hits both APUs equally. Withdrawing both would turn a degraded cache into no DNS at all.

Hysteresis: **FAIL_N=2** consecutive failures withdraw; **OK_N=2** consecutive successes restore. The timer fires every 10 s (`OnUnitActiveSec=10s`), so a real fault takes ~20 s to leave the ECMP set, and a flap of a single check is held.

flowchart TD

```
T["Timer every 10s"] --> F{"Default route in kernel FIB?"}
F -->|no| FAIL
F -->|yes| U{"Unbound answers on 127.0.0.1? any rcode"}
U -->|no| FAIL
U -->|yes| OK
FAIL["FAIL_N=2 consecutive"] --> DEL["ip addr del 10.53.53.53/32 dev lo"]
DEL --> W["OSPF withdraws the stub"]
W --> ECMP["CCR2116 ECMP drops this APU"]
OK["OK_N=2 consecutive"] --> ADD["ip addr add 10.53.53.53/32 dev lo"]
ADD --> ADV["OSPF re-advertises"]
```

Why this exists — Sep 2026 blackholes

When	Box	Issue	What happened	How long
2026-09-11	apu02	#497	OSPF Full, kernel FIB empty, anycast still advertised	~18 h
2026-09-12	apu01	#523 / #515	Same shape	~36 h

Trigger: a glibc upgrade made needrestart bounce `systemd-networkd`. `networkd` deleted FRR's kernel nexthop objects; the kernel dropped every route that used them. **FRR 8.4.4 never reinstalled them**. Unbound kept answering locally. The adjacency stayed Full. The core kept both next hops in `10.53.53.53/32`. Every client whose hash landed on the sick APU had **no DNS**, while pings, OSPF, and “unbound up” all looked fine.

Two other fixes sit beside the healthcheck (they do not replace it):

- `no zebra nexthop kernel enable` on both APUs (classic nexthops; the #515 exposure was kernel NHG).
- `needrestart` / `networkd` drop-ins so that bounce does not wipe the FIB.

The healthcheck is the last line: if the FIB is empty anyway, **stop advertising the anycast**.

Known interplay: `netplan` owns `lo`. A `networkd` reconfigure can put `10.53.53.53` back on an unhealthy APU; the next check (≤ 10 s) removes it again.

Safety rails

- **Never withdraw both APUs on purpose.** `--test-withdraw` first asks the **peer** (`ANYCAST_PEER` in `/etc/default/anycast-healthcheck`: `apu01` → `192.168.53.2`, `apu02` → `192.168.53.3`) for a `NOERROR`. If the peer is silent, the test refuses.
- `SERVFAIL` / upstream loss is **not** a withdraw. Only “no default in FIB” or “Unbound does not answer at all”.
- `--test-withdraw` writes a **hold file** so the timer cannot re-add the address mid-test. A hold older than 10 minutes is treated as leftover and removed.
- The script never restarts `FRR` or `Unbound`. It only adds/deletes one `/32` on `lo`.

```
anycast-healthcheck.sh           # one check (what the timer runs)
anycast-healthcheck.sh --status  # print state, change nothing
anycast-healthcheck.sh --test-withdraw [secs]  # default 20s; refuses if peer is down
```

Live proof (install `handover --apply --proof`, measured 2026-09-15 12:45Z on `apu02`): core dropped to one next hop for 11 samples, **0 DNS misses**, then both arms returned.

Where it lives

Piece	Path
Script on the APU	<code>/usr/local/sbin/anycast-healthcheck.sh</code>
Timer / unit	<code>anycast-healthcheck.timer</code> / <code>.service</code>
Peer env	<code>/etc/default/anycast-healthcheck</code> (<code>ANYCAST_PEER=...</code>)
State	<code>/run/anycast-healthcheck.state</code>
Workspace copy	<code>mikrotik-workspace/scratch/anycast-hc/anycast-healthcheck.sh</code>
Install handover	<code>tools/handover/2026-09-13-apu-anycast-healthcheck.sh</code>

The `hp02 /apu` page (REQ-DASH32) is the observer: timer state, `--status`, hold file, health-check journal, core ECMP set, and a per-APU verdict (`healthy` / `withdrawn` / `degraded` / `blackhole` /

unreachable). It does **not** act. Withdrawal is the box's decision; repair is the operator's.

Revision #3

Created 2026-09-16 15:46:31 UTC by Bodo

Updated 2026-09-16 16:20:19 UTC by Bodo