ThreeFold Grid support report — abnormal NAT/network behavior on two nodes

Date: 2026-08-04 **Network:** Tailscale-based mesh (self-hosted Headscale control plane), 8 VMs across 8 ThreeFold Grid nodes, provisioned via the Terraform Grid provider.

Summary

Two of our eight deployed VMs — hosted on **Grid node 8009** and **Grid node 11** — show clearly abnormal network behavior compared to the other six VMs in the same deployment (same account, same Tailscale/Headscale mesh, same Terraform module, provisioned the same way). We've found two distinct symptoms, detailed below.

We understand the Grid's value proposition as: hardware is abstracted away, and a workload should behave identically from a networking standpoint regardless of which physical node it happens to land on — an operator shouldn't need to know or care. That's not what we're observing here: two specific nodes produce measurably different, worse network behavior than the other six, for reasons that (as far as we can tell from inside the guest OS) are not something we caused or can fix ourselves. We're reporting this as a platform-level inconsistency, not asking for help debugging our own application — we've already ruled out our own configuration as the cause for both issues (details below).

Affected nodes

VM A	8009	Issue #1 (symmetric NAT)
VM B	11	Issue #1 (symmetric NAT) **and** Issue #2 (port-specific unreachability from peers)

Comparison (unaffected) nodes, for reference

1	yes
13	yes
10	yes
6968	yes
388	yes
50	yes (2 VMs on this node)

All VMs run the identical base OS image, identical Tailscale client version, and joined the same Headscale coordination server the same way (via preauth key, same ACL policy, same DERP configuration — we run our own embedded DERP relay in addition to using Tailscale's public DERP network).

Issue #1: Symmetric NAT on nodes 8009 and 11 breaks Tailscale connection setup for some external clients

Evidence

`tailscale netcheck`, run directly on each VM, shows a clear split:

Node 8009 (VM A) — abnormal:

IPv4: yes, 213.232.253.53:10405

MappingVariesByDestIP: true

Nearest DERP: Amsterdam

Node 11 (VM B) — abnormal:

IPv4: yes, 185.69.166.2:48015

MappingVariesByDestIP: true

Nearest DERP: (our own self-hosted relay)

Node 50 (unaffected VM) — normal, for comparison:

IPv4: yes, 185.69.167.143:54344

MappingVariesByDestIP: false

Nearest DERP: (our own self-hosted relay)

Node 13 (unaffected VM) — normal, for comparison:

IPv4: yes, 185.69.167.146:38850

MappingVariesByDestIP: false

Nearest DERP: (our own self-hosted relay)

`MappingVariesByDestIP: true` indicates the node's outbound NAT is **symmetric** (the external port assigned varies depending on the destination address) — a NAT type that is a well-documented weak point for WireGuard/Tailscale's STUN-based peer-to-peer connection establishment. Every other node in our deployment reports `false` (a "normal", endpoint-independent NAT).

Real-world impact observed

Two of our own client devices (both fully joined to the same tailnet with an otherwise-healthy connection to every *other* peer) cannot establish a Tailscale peer connection to nodes 8009 or 11 at all:

tailscale status --json (as seen from an affected client, for the node-8009 peer):

Online: true

LastHandshake: 0001-01-01T00:00:00Z (never — zero epoch)

RxBytes: 0

tailscaled log on the same client:

wg: [<peer-key>] - Failed to send handshake initiation: no UDP or DERP addr

We confirmed this is **not** stale local state:

- Restarted `tailscaled` on the affected client (no change).
- Ran `tailscale up --reset` with a fresh Headscale preauth key on **node 8009 itself**, giving it a brand-new WireGuard node key — the handshake still could not be established from the affected clients afterward.
- Ran `tailscale up --reset` with a fresh identity on the affected client too — still no change.

Both nodes 8009/11 individually show they *can* reach our own self-hosted embedded DERP relay fine (`tailscale netcheck` reports ~9–24ms latency to it from every node, including 8009/11), and the affected external clients can also reach that same DERP relay fine — yet a direct peer connection between the client and nodes 8009/11 specifically never completes, even via DERP relay fallback.

By contrast, every VM-to-VM connection *within* the deployment (all 8 nodes talking to each other) is completely healthy — fresh handshakes, normal latency, no issues. The symptom is specific to a peer connection either *originating from* or *destined to* nodes 8009/11 when the other side is an external client outside the Grid's own network.

Issue #2: Node 11 (VM B) — specific TCP ports unreachable from other Grid VMs, despite a healthy Tailscale handshake

This is a **separate** symptom from Issue #1, observed between two Grid VMs that already have a fully healthy, current Tailscale connection to each other (not an external client). We are not certain it shares the same root cause as Issue #1, but flagging it here since it's the same physical node (11).

Evidence

From two other VMs in the deployment (node 8009, and separately node 13 — a normal-NAT node) to VM B (node 11, tailscale IP `100.64.0.11`), which runs two ordinary application containers listening on ports 8080/tcp and 80/tcp respectively:

tailscale ping 100.64.0.11

pong from VM B (100.64.0.11) via 10.10.9.2:41641 in 10ms ✅ succeeds

nc -zv 100.64.0.11 22

Connection to 100.64.0.11 22 port [tcp/ssh] succeeded! ✅ succeeds

nc -zv 100.64.0.11 8080

(no output, silently hangs until timeout) ❌ fails

nc -zv 100.64.0.11 80

(no output, silently hangs until timeout) ❌ fails

curl http://100.64.0.11:8080/... --max-time 5

curl: (28) Connection timed out after 5001 milliseconds ❌ fails

The same two ports, checked **locally on VM B itself** against its own tailscale address, respond correctly (HTTP 200 on both). So the applications themselves are healthy and correctly bound; the failure is specifically in the network path *to* those two ports on node 11, from any other peer.

What we ruled out on our side before writing this up

- **Stale/crashed application containers** — recreated both containers via a full redeploy; no change.
- **Our own firewall (UFW)** — the existing rule already allows all traffic on the `tailscale0` interface; we additionally added an explicit temporary allow rule for the exact source IP and port, and it made no difference (removed afterward).
- **Tailscale's own anti-spoofing iptables rule** (`ts-input` chain, drops any packet claiming a `100.64.0.0/10` source that didn't arrive via the `tailscale0` interface) — checked packet counters directly: traffic is being **accepted** by the correct rule (arriving via `tailscale0` as expected), not matching the drop rule.
- `rp_filter` **(reverse-path filtering)** — disabled (`0`) on every interface, including `tailscale0` and `eth0`.
- **Policy/multi-table routing** — the box has a Tailscale-managed routing table (`ip rule` shows `lookup 52` at a higher priority than `lookup main`). `ip route get` for the peer initiating the connection correctly resolves via `tailscale0` in that table — the response path is not being silently misrouted out the wrong interface.
- **Missing local-delivery route** — `ip route show table local` correctly has `local 100.64.0.11 dev tailscale0 ... scope host`, so the kernel does know this address is its own.
- `raw`**/**`mangle`**/**`filter` **table policies** — all default-ACCEPT for OUTPUT; no DROP/REJECT rule matches this traffic in any table we could find.
- **Packet corruption** — captured the actual SYN packets with `tcpdump -vv` (checksum verification): checksums are correct on every retransmission.
- **fail2ban or similar** — not installed/running on the host.

The most useful finding: a real packet capture (`tcpdump -i tailscale0 -n 'tcp port 8080'`, run directly on node 11) shows the peer's SYN packets **do physically arrive** on the `tailscale0` interface, correctly and repeatedly (retransmissions visible) — but **zero response is ever sent back** (no SYN-ACK, no RST — captured the same interface in both directions, nothing outbound at all). Cross-checked against `conntrack -L`: **no connection-tracking entry is ever created** for this traffic at all, despite `nf_conntrack_max` being nowhere near its limit (262144). A locked-down firewall rule would still normally show *up* in conntrack (as an entry that then gets dropped); the complete absence of any tracked entry, combined with a verified-correct checksum and every routing/local-delivery check passing,

suggests something is discarding this specific traffic **before** it reaches the kernel's normal netfilter/conntrack processing — i.e. below what we can inspect from inside the VM (most likely the hypervisor's own virtual switch/network filtering on this specific compute node).

This is as far as we can take it from inside the guest OS.

What we're asking

1. **Root cause**: what, on nodes 8009 and 11 specifically, produces symmetric (`MappingVariesByDestIP: true`) NAT behavior that no other node in our deployment shows? We'd expect this to be uniform across the Grid regardless of which node a workload lands on, so we're treating this as a platform inconsistency worth fixing at the source, not a per-node quirk to route around.
2. **Root cause, Issue #2**: given the packet-capture evidence above (SYN arrives at the guest's `tailscale0` interface, no response ever generated, no conntrack entry ever created, checksums verified correct) — is there a hypervisor-level virtual switch, security group, or network-filtering layer on node 11 that could selectively drop traffic to certain destination ports while leaving others (SSH, ICMP) completely unaffected? This looks like it's happening below what we can inspect or control from inside the guest OS, so we have no further diagnostic path from our side.

Happy to provide any additional deployment details, exact timestamps, or run further diagnostics on request.