

Der Grossteil des Request-Traffics ist eine Retry-Schleife, die nie gelingen kann

English version: 2026-07-29-request-response-traffic-analysis.en.md

Datum: 2026-07-29

Kurzfassung

Request/Response-Pakete sind **38,4 % des gesamten Traffics** im EU-Mesh. Wir haben untersucht, woher sie kommen — in der Erwartung, Missbrauch zu finden. Es ist kein Missbrauch. Es ist ein **Fehlermuster, das fast jedes Polling-Script teilt**:

Eine Abfrage, die nie beantwortet wird, flutet das gesamte Netz — für immer, und der Sender kann es nicht bemerken.

Zwischen **88 % und 94 %** der gesamten Request-Last (je nach Zählweise) entfallen auf FLOOD-Requests, die nie eine Antwort bekommen. Einzelne wiederholen sich seit über einer Woche alle 39 Sekunden, Tag und Nacht.

Wer ein Script betreibt, das Repeater abfragt: **dieser Text handelt wahrscheinlich von deinem Script**, und der Fix sind drei Zeilen (§7).

Datenbasis

Kennzahl	Wert
Zeitraum	2026-07-22 – 2026-07-28 (6 Tage)
Aktive Observer	367
Abgedeckte IATA-Zonen	57
Pakete	524 115
Sichtungen	13 413 130
Aktive Nodes	8 939
davon Request/Response	201 448 Pakete (38,4 %)

Messgrösse „Empfangs-Bytes“: Paketgrösse × Anzahl der Observer, die es gehört haben. Das misst die **Verbreitung** eines Pakets im Netz, nicht die Sendezeit. Im Fenster wurden 30 MB einmal gesendet und dabei als 688 MB über alle Empfänger gehört — Faktor 23. Echte Airtime lässt sich daraus nicht ableiten, weil wir Retransmissions der Repeater nicht zählen, sondern nur, was Observer hören.

1. Warum unbeantwortete Abfragen dauerhaft teuer sind

Route	Pakete	Sichtungen	Ø Empfänger pro Paket
FLOOD	48 173	2 302 805	47,8
TRANSPORT_FLOOD	14 663	973 176	66,4
DIRECT	40 400	69 067	1,71

Ein FLOOD-Request erreicht ~28× so viele Funkgeräte wie ein DIRECT-Request.

MeshCore löst das bereits, und der Mechanismus funktioniert: Der erste Request geht als FLOOD raus. **Die Antwort bringt den Rückweg mit.** Ab dann pollt der Client DIRECT, und die Kosten brechen ein. Gemessen an einem realen Paar (e4 → 24):

Tag	Requests	davon FLOOD
07-23	96	55 %
07-24	473	4 %
07-26	224	5 %

Und hier der Fehlerfall: **Der Pfad wird aus der Antwort gelernt.** Keine Antwort, kein Pfad. Kein Pfad, und der nächste Request geht wieder als FLOOD raus. Für immer.

In dieser Schleife gibt es keinen Backoff und keinen Fehlerzustand. Aus Sicht des Clients ist nichts kaputt — er fragt einfach weiter. Aus Sicht des Netzes wird jeder einzelne Versuch an jedes Funkgerät in Reichweite jedes Repeaters gesendet.

2. Die Aufteilung der Request-Last

Kriterium A — pro (Sender→Ziel)-Paar, Paar gilt als bedient ab 20 % Antwortrate:

Kategorie	Paare	Requests	Sichtungen	Anteil	Empfangs-Bytes
FLOOD, unbeantwortet — Verschwendung	4 088	57 184	2 902 094	88,1 %	92 MB
FLOOD, wird beantwortet	579	7 528	313 377	9,5 %	10,0 MB
DIRECT (Pfad gelernt) — günstig	927	37 895	78 816	2,4 %	1,9 MB

Kriterium B — pro individuellem Paket, Antwort muss binnen 10 min eintreffen:

Kategorie	Pakete	Sichtungen	Anteil
FLOOD, keine Antwort	60 695	3 104 090	94,2 %
FLOOD, Antwort binnen 10 min	1 488	121 096	3,7 %
DIRECT	40 424	69 101	2,1 %

Beide Partitionen summieren exakt auf 3 294 287 Sichtungen — die Kontrollsumme aller ausgewerteten Request-Sichtungen. Kriterium A ist das konservativere, weil dort ein Paar oberhalb der Schwelle *alle* seine Requests als beantwortet gutgeschrieben bekommt. Belastbare Aussage: **88–94 %**.

Die DIRECT-Zeile ist die gute Nachricht: sauber konfiguriertes Polling ist praktisch gratis — 37 895 Requests kosten zusammen 2,4 % der Last. Es geht nicht darum, weniger zu pollen.

3. Top-Verursacher (FLOOD, ohne Antwort)

Sender	Ziel	Requests	Antworten	Sichtungen	Req/Tag
27	f1	5 040	0	471 240	840
f4	19	1 596	44	186 252	266

Sender	Ziel	Requests	Antworten	Sichtungen	Req/Tag
bb	30	2 007	0	90 501	335
5d	98	1 361	0	68 536	227
7f	74	699	0	49 879	117
7f	b5	678	0	48 154	113
13	89	825	25	39 419	138
10	3e	1 037	0	32 476	173
69	1f	1 053	0	27 111	176

27 → f1 allein ist **14,4 % der gesamten FLOOD-Request-Last**: 100 % FLOOD an allen 7 Tagen, am 27.07. 2 238 Requests (alle 39 s), auf dieser Strecke keine einzige Antwort.

Der Defekt ist streckenspezifisch, nicht generell — beide Seiten sind aktiv:

- f1 sendet 114 Responses an 19 andere Peers, nur nie an 27.
- 27 empfängt Antworten: 186 von ae, dazu einzelne von 19 weiteren Peers.

Naheliegende Erklärung: ein veralteter Schlüssel oder ein Kontakt, der auf einer Seite entfernt wurde — die MAC-Prüfung schlägt fehl, es kommt keine Antwort. *Das ist eine Vermutung — verifiziert ist nur, dass beide Seiten leben und nur diese eine Strecke tot ist.* Genau so etwas fällt niemandem auf, weil es nirgends gemeldet wird.

4. Das sind Scripts, keine Menschen

Alle 23 Top-Sender sind über alle 24 Stunden aktiv, mit einem Nacht/Tag- Verhältnis von 0,82–1,64 — mehrere senden nachts *mehr* als tagsüber.

Die Intervalle sind Cron-Takte (Median-Abstand pro Paar, Anteil innerhalb ± 10 %):

Sender→Ziel	Takt	Pakete	im Takt
6e → 4 Ziele	3600,0 s (1 h)	je ~130	100 %
c1 → 3c	3600,1 s	102	94 %
2a → 14 Ziele	~2123 s (35 min)	je ~165	89–94 %
52 → 96	318 s (5 min)	1 205	94 %
f6 → c1	1825 s (30 min)	212	92 %
dd → 79	1827 s	198	77 %
af → 3e	621 s (10 min)	645	75 %
e4 → cc	631 s	214	69 %

2a arbeitet **14 Repeater im selben 35-Minuten-Zyklus** ab. Dieser Scanner sendet durchgehend DIRECT und kostet das Netz fast nichts — der Beweis, dass Scannen an sich nicht das Problem ist.

5. Wo die Verursacher stehen

Der `src_hash` im Request ist nur **1 Byte** (Firmware: `MeshCore.h:18` `PATH_HASH_SIZE 1` ; `Mesh.cpp:488-489` `dest.copyHashTo / self_id.copyHashTo` in `createDatagram`). Bei 8 940 bekannten Nodes teilen sich im Schnitt **35 Nodes** einen Hash — **eine eindeutige Identifikation des Senders ist daraus nicht möglich**.

Was *möglich* ist: der erste Path-Hash eines Flood-Pakets ist der erste weiterleitende Repeater, also ein Funknachbar des Senders. Wo dieser Pfad mit 2–3 Byte Hashes läuft, ist er **eindeutig auflösbar**:

Sender	Heimat-Repeater	Standort
bb	DE-NW-Ratingen-1 (3012 Pk.) + DE-NW-Ratingen-3 (2771 Pk.)	51.303, 6.866 — identische Koordinaten, ein Standort
f4	NL-ZH-NIE-T1 (908 Pk.)	51.965, 4.596 (Nieuwerkerk a/d IJssel)
91	NFN-910#59-01(DE.ERL) (730) + MeschBertsRepeater (153)	Erlangen
db	Paul_repeater (387) + RAKpeater (289)	52.073, 5.089 — ein Standort (Utrecht)
13	NL-HEN-RP01-PerksMC.nl (45)	50.878, 6.005

Diese Repeater sind nicht die Ursache. Ein Heimat-Repeater ist die erste Station, die das Paket weitergeleitet hat — der pollende Client steht *in seiner Funkreichweite*. Das ist eine Nachbarschaftsbeziehung, keine Betreiberbeziehung. Die Betreiber sind nützliche Ansprechpartner, weil sie das Gerät vor Ort erkennen können, mehr nicht.

Für 27 (den grössten Verursacher) gibt es nur 1-Byte-Hops: Heimat-Repeater beginnt mit `ae` (99,6 % konsistent — also **ein** fester Repeater). `ae` ist über zwei unabhängige Datenpfade belegt: erster Hop im Hinweg *und* der Peer, der 27 mit Abstand am häufigsten antwortet (186 Responses, mehr als alle anderen zusammen).

6. Kandidaten für die Ziel-Repeater

Der `dest_hash` ist ebenfalls nur 1 Byte, aber die Ziele sind besser greifbar: Repeater senden Adverts, und ein Advert enthält den **vollen** Pubkey. Damit lassen sich Kandidaten filtern — plausibel ist nur, wer (a) Repeater oder Roomserver ist und (b) dessen Advert in einer Zone gehört wird, in der auch die Requests des Senders ankommen (≥ 10 % der Sender-Sichtungen).

Das reduziert 21–38 Namensvetter auf **0–6 echte Kandidaten**:

Paar	Kandidaten	Namen
bb → 30	0	kein passender Repeater in der Region — siehe unten
7f → b5	2	Taucha 🇩🇪 Flugplatz, 🇩🇪 Loehne Horst
5d → 98	3	D-NW-E-TripleZ, Zwickau Repeater, Calberlah
13 → 89	3	DLW_REP01_LS, FF MCR BE Moabit+ 🇩🇪, RockyBeach_P1_Solar
7f → 74	3	HAS Suedwest JO50EB, Heltec Repeater Algermi, nhf-li9urian
69 → 1f	4	DE-NW-ME-VBT10, NL-LTM-RP02, Relais Roxel, BE-PUT-ON1EE-31A
f4 → 19	5	NL-ZH-LID-LaMo-RPT 🇩🇪, NL-AME-SR-r-001 🇩🇪, NL-ZWL-MeshJoeWindeshei, NL-ZH-LID-MAAR-solar, WNS Romeo 2 🇩🇪

Paar	Kandidaten	Namen
27 → f1	6	NL-OV-FIETS, NL-NH-SMT3-Rood (beide AMS+RTM), ipoac.nl zuid/oost, NL6812-16 Yagi SW 240, NL-ZEM-LM, NL-ENS-Stadsveld
10 → 3e	16	zu breit gestreut, nicht aussagekräftig

Zwei Lesehilfen:

- **bb → 30 mit null Kandidaten ist der stärkste Befund der ganzen Tabelle.** Kein Repeater mit Präfix 30 sendet Adverts, die in den Empfangszonen von bb ankommen. Das Ziel ist mit hoher Wahrscheinlichkeit abgeschaltet oder aus der Region verschwunden — und ein Script fragt es seit Tagen 335× täglich per FLOOD ab. Das ist das Fehlermuster in Reinform.
- Bei 27 → f1 sind **NL-OV-FIETS** und **NL-NH-SMT3-Rood** die besten Treffer: nur diese beiden werden in *beiden* Hauptzonen des Senders (AMS und RTM) gehört.

Das sind Kandidatenlisten, keine Zuordnungen. Ein 1-Byte-Hash lässt keine Beweisführung zu; die Liste sagt „einer von diesen“, nicht „dieser“.

Die Live-Ansicht rechnet schärfer als diese Tabelle. Der „Request health“-Tab (Migration 0040) wendet dieselbe Regel **pro Tag** an statt über das ganze 6-Tage-Fenster. Weil die Empfangszonen eines Paares an einem einzelnen Tag enger sind, fallen mehr Kandidaten heraus: 27 → f1 reduziert sich dort auf **einen** Namen (NL-ENS-Stadsveld) statt der sechs unten. Beide Zahlen sind für ihre Methode korrekt — die Tabelle hier ist die konservativere.

7. Was im Script zu ändern ist

Das Problem ist nicht das Pollen. Es ist das **unbegrenzte Wiederholen im vollen Takt gegen ein Ziel, das nie antwortet**.

1. **Fehlschläge pro Ziel zählen.** Antwortet ein Ziel in den letzten N Versuchen nicht (N = 3 reicht), hör auf, es im Normaltakt abzufragen.
2. **Exponentiell zurückfahren, mit Deckel.** 5 min → 10 → 20 → ... → einmal täglich. Ein Ziel, das zurückkommt, wird binnen eines Tages wieder erfasst; ein totes kostet dann fast nichts.
3. **Die tote Strecke protokollieren.** Hätte das Script gemeldet, dass es seit 5 000 Versuchen keine Antwort bekommt, wäre es in der ersten Woche aufgefallen.

Wenn dein Client tagelang FLOOD an dasselbe Ziel schickt, ist genau das das Signal: er hat nie eine Antwort erhalten, denn die Antwort ist es, die den Pfad installiert.

8. Was diese Daten nicht hergeben

- **Wir können nicht feststellen, wer du bist, und haben es nicht versucht.** Das Absenderfeld ist ein 1-Byte-Hash; bei ~8 900 bekannten Nodes teilen sich im Schnitt 35 davon denselben Wert.
- **Inhalte bleiben verschlüsselt.** Request- und Response-Nutzlasten sind zwischen den beiden Endpunkten verschlüsselt (curve25519). Ein passiver Beobachter hat keinen der beiden Schlüssel. Diese Auswertung nutzt ausschliesslich Paket-Header und Zeitstempel — nie Nutzlast.
- **Es geht um kein Fehlverhalten von Repeater-Betreibern.** Wir können oft sagen, welcher Repeater ein Paket zuerst weitergeleitet hat. Das verortet einen Client irgendwo in dessen Funkreichweite — mehr nicht.

9. Methodik und Grenzen

Standortbestimmung stützt sich ausschliesslich auf den ersten Path-Hash über die 83,7 % der Sichtungen mit befülltem Pfad. Zwei Wege scheiden aus:

- *0-Hop-Sichtungen* — `sightings.path` ist zu 16,2 % NULL (Feld nicht geliefert), echte 0-Hop-Sichtungen machen nur 0,1 % aus. Die Datenmenge trägt keine Aussage, und NULL darf nicht als 0 gelesen werden.
- *Signalstärke* — die Observer mit dem stärksten Empfang von `bb` (RSSI -19,3) sehen das Paket bei `min_hops` 4–5. Gemessen wird der letzte Repeater, nicht der Sender.

Der Sender selbst ist auch über die Node-Tabelle nicht auffindbar. Zwei weitere Wege wurden geprüft, beide ergebnislos:

- *Position der Kandidaten.* Kein Kandidat mit passendem Präfix liegt nah genug am zugehörigen Heimat-Repeater — der nächste ist 17,4 km entfernt, alle übrigen 25–60 km, für die Funkreichweite eines Clients unplausibel. Ursache ist die Datenlage: Companions senden nur zu **39,5 %** eine Position (Repeater zu 95,6 %), und pro Sender-Hash gibt es nur 1–4 Companions mit Koordinaten.
- *Gemeinsame Funknachbarschaft.* Sendet ein Kandidat selbst Adverts, müssten diese über denselben ersten Repeater laufen wie seine Requests. Für alle fünf aufgelösten Sender: **null Treffer**. Kontrollprüfung gegen ein Query-Artefakt: 43 Nodes mit Präfix `bb` senden im Fenster 307 Adverts mit befülltem Pfad, über erste Hops wie `0f`, `14`, `4e`, `7777`, `38a1` — aber keiner über Ratingen.

Die wahrscheinlichste Erklärung ist, dass die Poller selbst keine Adverts senden und deshalb in `nodes` gar nicht vorkommen — die Tabelle wird aus Adverts befüllt. *Bewiesen ist das nicht*: dasselbe Ergebnis entstünde auch, wenn ein Sender zwar Adverts sendet, diese aber nie mit befülltem Pfad gehört werden. Praktisch läuft beides auf dasselbe hinaus — der Betreiber des Heimat-Repeaters bleibt der einzige Weg zum Verursacher.

Antwortraten sind eine Untergrenze. Responses laufen meist DIRECT (Ø 1,36 Empfänger) und werden von den Observern schlecht gehört. Bei den 0-Antworten-Paaren stützt sich der Befund deshalb nicht auf die Zählung allein, sondern auf den **100-%-FLOOD-Anteil über alle 7 Tage** — hätte der Sender je eine Antwort erhalten, wäre er auf DIRECT umgestiegen.

IATA-Codes in diesem Dokument (RTM, AMS, BER, BWE, LEJ ...) bezeichnen die Zonen der *empfangenden Observer* — also wo ein Paket gehört wurde. Sie sind nicht die Flood-Scope-Region des Pakets.

Wer dieselbe Auswertung für seine eigene Region fahren möchte: die Methodik teilen wir gern.