Bots die elkaar taken geven, sneller dan 2 minuten pollen

> Vijf onderzoeken samengevoegd (Telegram-beperkingen, multi-agent frameworks, message-queues, developer-ervaringen, praktijkvoorbeelden). Alle links hieronder zijn echt en komen uit die vijf bronnen. Geen enkele link is verzonnen.

In het kort

- **Ja, het klopt**: Telegram-bots zien elkaars berichten normaal niet. Dat is precies waarom je nu de Supabase-tabel + polling + het `[→SJAKIE: ...]`-format gebruikt.
- Sinds **7 mei 2026** heeft Telegram zelf dit deels opgelost met een nieuwe functie: "Bot-to-Bot Communication Mode". Nog erg nieuw (3 maanden oud).
- Je huidige systeem (Supabase-tabel + LaunchAgent-poller elke 2 minuten) is precies het patroon dat de meeste developers ook kiezen. Je doet dus niets geks. Het zit alleen "traag" omdat je nog polling gebruikt in plaats van een directe push.
- **Beste concrete verbetering voor jou**: blijf op Supabase (je hebt het al), maar vervang de 2-minuten-poller door **Supabase Realtime**. Dat is instant in plaats van tot 2 minuten wachten. Klein bouwwerk, geen nieuwe dienst nodig.
- De Telegram-groepen met `[→SJAKIE: ...]`-tags blijven gewoon nuttig. Die zijn er voor jou, om mee te kunnen kijken. Die hoef je niet weg te halen.

---

1. Klopt het, en wat doet je huidige systeem

Telegram zegt het zelf letterlijk:

> "Bots talking to each other could potentially get stuck in unwelcome loops. To avoid this, we decided that bots will not be able to see messages from other bots regardless of

> mode."

Bron: https://core.telegram.org/bots/faq

Dat is de reden dat je nu werkt met:
- een Supabase-tabel `agent_messages` waar bots berichten voor elkaar in zetten
- een LaunchAgent ("wachtrij-wachter") die deze tabel elke 2 minuten checkt
- een rem tegen bot-lussen (max 3 antwoorden per uur per bot)
- Telegram-groepen met een handmatig format (`[→SJAKIE: ...]` / `[→CONDUCTOR: ...]`) zodat jij het kan volgen

Dit is exact het patroon dat developers overal gebruiken als workaround voor de Telegram-beperking. Niet ouderwets, gewoon de standaard-oplossing.

Let op (nieuw sinds kort): op 7 mei 2026 voerde Telegram een "Bot-to-Bot Communication Mode" in. Bots kunnen elkaar dan wél direct een privébericht sturen, als beide bots dat aanzetten via BotFather. Bron: https://core.telegram.org/api/bots/bot-to-bot en https://core.telegram.org/bots/features

Belangrijk eerlijk punt: Telegram zegt er zelf iets kritisch bij:

> "Bot-to-bot communication can create infinite reply loops. Bots using this feature must make bot-message handling terminate predictably."

Dus de loop-bescherming die je nu al hebt (max 3 per uur) moet je dan zelf blijven bouwen. Telegram levert alleen de bezorging, niet de veiligheid.

---

2. De kansrijkste alternatieven

Optie 1: Supabase Realtime in plaats van polling (kleinste stap)

Hoe het werkt: in plaats van dat elke bot elke 2 minuten de tabel checkt, opent elke bot 1 verbinding (websocket) naar Supabase. Zodra er een nieuw bericht binnenkomt, krijgt de bot dat meteen doorgestuwd. Twee smaken:
- **Postgres Changes** (luistert op wijzigingen in je bestaande tabel): ongeveer 50 tot 200

milliseconden vertraging.
- **Broadcast** (los kanaal, kan ook zonder open verbinding via een gewone API-aanroep): meestal onder 50 milliseconden.

Bronnen: https://supabase.com/docs/guides/realtime/benchmarks · https://supabase.com/docs/guides/realtime/broadcast · https://supabase.com/docs/guides/realtime/limits

Waarom beter dan pollen: instant in plaats van tot 2 minuten wachten. Dat is het hele "vervelend traag"-gevoel opgelost.

Bouwwerk: klein. Je gebruikt hetzelfde Supabase-project, dezelfde tabel, dezelfde structuur. Je verandert alleen de relay-scripts (`agent-relay-conductor.py` , `sjakie-relay.py`) van "elke 2 minuten checken" naar "luisteren en meteen reageren". De rem tegen bot-lussen (max 3 per uur) blijft gewoon werken, die verandert niet.

Gratis-limiet: 200 gelijktijdige verbindingen en ongeveer 2 miljoen berichten per maand. Voor 6 tot 8 bots ruim voldoende.

Optie 2: Telegram Bot-to-Bot Communication Mode (nieuw, mens-zichtbare laag)

Hoe het werkt: sinds 7 mei 2026 kan een bot met een aangezet "Bot-to-Bot"-vlag (via BotFather) een direct privébericht sturen naar een andere bot die dat ook heeft aanstaan. Ook mogelijk in een groep, als de bot admin is of Group Privacy Mode uit heeft staan.

Bron: https://core.telegram.org/bots/features (sectie "Bot-to-Bot Communication")

Waarom beter dan pollen: dit zou de handmatige `[→SJAKIE: ...]` -tag kunnen vervangen door een echt direct bericht, in plaats van dat de tag alleen zichtbaar is en de échte delegatie via Supabase loopt.

Bouwwerk en eerlijke kanttekening:
- moet je per bot aanzetten via BotFather (beide kanten)
- je moet zelf een loop-guard bouwen (dedupliceren, rate-limits) — dat staat letterlijk in Telegrams eigen documentatie als verplichting voor de bot-bouwer
- dit is pas 3 maanden in productie (7 mei 2026), dus nog weinig bewezen in de praktijk
- een onafhankelijke analyse noemt drie zwaktes: platform lock-in (werkt niet als je ooit naar Slack/eigen infra wil), geen ingebouwde controle op berichtintegriteit, en geen dead-letter-queue voor onbereikbare bots. Bron: https://dev.to/kavinkimcreator/telegram-just-opened-the-door-for-agent-to-agent-communication-heres-why-thats-not-enough-3gmn

Eerlijk, veelzeggend detail: het grote open-source project OpenClaw kreeg precies dit verzoek (bot-naar-bot zichtbaar maken via Telegram) en heeft dat bewust **afgewezen** ("not planned"). Ze houden delegatie liever intern/onzichtbaar en gebruiken Telegram alleen als venster voor de mens. Bron: (issue #85754, genoemd in de developer-ervaringen research, github.com/openclaw/openclaw)

Optie 3: NATS + JetStream (volgende stap omhoog, als het later moet)

Hoe het werkt: een klein, apart programma (1 bestand, geen dependencies) dat je op de VPS zet. Elke bot maakt zelf een verbinding naar buiten toe (dus geen probleem met bots achter een router zonder vast IP). JetStream bewaart berichten ook als een bot even offline is.

Bronnen: https://docs.nats.io/running-a-nats-service/introduction/installation · https://onidel.com/blog/nats-jetstream-rabbitmq-kafka-2025-benchmarks

Waarom beter dan pollen: sub-milliseconde tot 5 milliseconden vertraging, en berichten gaan niet verloven als een bot een herstart doet.

Bouwwerk: middelgroot. Dit is wel nieuwe infrastructuur: 1 server-proces installeren op de VPS (`brew install nats-server` op Mac, `choco install nats-server` op Windows, een binary op de VPS) en op elke machine een klein stukje client-software.

Wanneer dit pas zinnig is: alleen als Supabase Realtime ooit tegen de gratis-limiet aanloopt, of als je meer garantie wilt dan "best effort".

Optie 4 (niet aanbevolen, alleen ter vergelijking): Redis of RabbitMQ

- **Redis Pub/Sub of Streams**: snel (sub-milliseconde), maar je moet zelf een Redis-server draaien. Streams bewaren berichten (net als NATS JetStream), Pub/Sub niet.
  Bron: https://geeksforgeeks.org/system-design/difference-between-redis-pub-sub-vs-redis-streams
- **RabbitMQ**: krachtig, maar de installatie op Windows is een bekende bron van ellende (specifieke Erlang-versie nodig, moet als administrator geïnstalleerd worden). Precies het soort Windows-gedoe dat je al eerder tegenkwam bij Lenovo en Paco (python/python3-alias-bug).
  Bron: https://rabbitmq.com/docs/which-erlang
- **Conclusie**: voor 6 tot 8 bots is dit overkill. Niet aan te raden als eerste keus.

3. Vergelijking op een rij

OPTIE	VERTRAGING NU (2 MIN) WORDT	BOUWWERK T.O.V. WAT JE AL HEBT	NIEUWE INFRAS
Supabase Realtime	onder 50 tot 200 ms	klein, zelfde tabel	nee, je hebt he
Telegram Bot-to-Bot Mode	instant, maar alleen de Telegram-laag	middel, plus zelf loop-guard bouwen	nee, wel BotFa bot
NATS + JetStream	sub-ms tot 5 ms, met bewaring bij offline bot	middel tot groot	ja, 1 server-pro
Redis / RabbitMQ	snel, maar RabbitMQ zwaar op Windows	groot (RabbitMQ), middel (Redis)	ja

4. Mijn aanbeveling

Ga voor optie 1: Supabase Realtime.

Waarom dit het beste bij jou past:
- Je gebruikt Supabase (`haarvisie-brain`) al voor de `agent_messages` -tabel. Geen nieuwe dienst, geen nieuw account, geen nieuwe kosten.
- Werkt over Mac, Windows en de VPS zonder gedoe. Het is gewoon een websocket/REST-verbinding, elke taal kan dat.
- Verandert niets aan wat al goed werkt: dezelfde tabel, hetzelfde format, dezelfde rem tegen bot-lussen (max 3 per uur).
- Lost precies het "vervelend traag"-gevoel op: van tot 2 minuten wachten naar onder de 200 milliseconden.
- De Telegram-groepen met `[→SJAKIE: ...]` blijven gewoon werken als jouw zichtbare venster. Die hoeven niet weg.

Telegram Bot-to-Bot Mode: interessant om in de gaten te houden, maar ik zou er nu nog niet op bouwen. Het is 3 maanden oud, vraagt een eigen loop-guard, en zelfs OpenClaw (een

groot project dat er zelf om vroeg) koos er bewust voor het niet te gebruiken voor echte delegatie. Wel iets om over een paar maanden nog eens te bekijken als het zich bewijst.

NATS: bewaar ik als plan B. Pas nodig als de Supabase-gratis-limiet (200 verbindingen, ongeveer 2 miljoen berichten per maand) ooit knelt. Voor 6 tot 8 bots is dat nog ver weg.

Dit is advies. Zeg go als je wilt dat ik de Supabase Realtime-overstap concreet ga bouwen (relay-scripts omzetten van polling naar luisteren), dan zet ik dat in gang.

Bronnen (allemaal echt, uit de vijf onderzoeken)

- https://core.telegram.org/bots/faq
- https://core.telegram.org/api/bots/bot-to-bot
- https://core.telegram.org/bots/features
- https://dev.to/kavinkimcreator/telegram-just-opened-the-door-for-agent-to-agent-communication-heres-why-thats-not-enough-3gmn
- https://supabase.com/docs/guides/realtime/benchmarks
- https://supabase.com/docs/guides/realtime/broadcast
- https://supabase.com/docs/guides/realtime/limits
- https://docs.nats.io/running-a-nats-service/introduction/installation
- https://onidel.com/blog/nats-jetstream-rabbitmq-kafka-2025-benchmarks
- https://geeksforgeeks.org/system-design/difference-between-redis-pub-sub-vs-redis-streams
- https://rabbitmq.com/docs/which-erlang
- https://github.com/openclaw/openclaw (issue #85754, genoemd in de developer-ervaringen research)