閉じても答えを書き終えるチャット

iPhone が通信を切る2つの形と、中身を読めない預かり所：本番の AI アシスタントでの設計と実機計測

Finishing the Answer After the Screen Is Gone: Two Ways iOS Drops a Streaming Reply, and an Encrypted Parcel Store the Server Cannot Read

湯村 信司（Shinji Yumura）　AQUA合同会社　2026年10月26日

27 / 27
iPhone で画面を離れた27回すべてで、全文（最長 18,545 文字）が届いた。本番の答えは1〜2万字で1〜2分かかり、利用者は送って2秒ほどで裏に回っていた

2 通り
iPhone の切り方。「エラーを出して切る」18回、「エラーも終わりも出さずに黙って止める」8回。後者は、見張りを足すまで、開き直さないと何も起きなかった

0.2 〜 1.0 秒
サーバーが書き終えたあとに戻ったとき、戻ってから全文が画面に入るまで（20回）。預かり所は暗号しか持たず、鍵は端末の外に出ない

ABSTRACT

Long streamed replies from an AI assistant rarely survive a phone being put away: iOS suspends the page and drops its connection within seconds, so the reply stops where it was. We added to YUA, a deployed assistant, a mechanism in which the server keeps writing after the client is gone, holds the finished reply for up to one hour in a parcel store that cannot read it (AES-GCM, 256-bit key generated on the device and never stored server-side, one-time pickup, 128-bit random ticket), and the device collects it on return. A five-second acknowledgment window avoids storing anything when the client did receive the whole reply, and an explicit end marker distinguishes a complete stream from one that iOS cut "cleanly". We measured 29 trials on a real iPhone (Safari) against production: all 27 trials in which the user left the screen delivered the full reply (up to 18,545 characters); when the server had already finished, the reply appeared 0.2–1.0 s after return (n = 20). The device logs revealed two distinct ways iOS drops the stream: 18 times with a fetch error ("Load failed") and 8 times silently, with neither an error nor an end of stream, so that ordinary error handling never fires and nothing happens until the page is reloaded. We handle the silent case with two watchdogs that abort the stream after 8 s of silence following a return to the foreground, or 30 s of silence while visible, and verified them on a real WebKit (iOS Simulator) by injecting the stall through a TCP relay, since the simulator itself never drops connections in the background. We report the design, the measurements, and the reproduction method, and we are explicit about what a web page on iOS cannot do: keep streaming in the background, or notify the user when the reply is ready.

要旨

AI アシスタントの長い答えは、iPhone で画面を離れるとまず届かない。iOS は裏に回ったページを止め、数秒で通信を切るので、答えは途中で止まったままになる。本稿は、本番の AI アシスタント YUA に、「端末が離れてもサーバーが最後まで書き、書き終えた答えを中身の読めない預かり所に最長1時間あずけ、端末が戻ったら1回きりで受け取る」仕組みを入れ、実機で測った報告である。鍵（256ビット）と受け取り番号（128ビット）は端末が作り、鍵はサーバーに保存されない。答えは AES-GCM で暗号にしてあずけ、預かり所は中身を読めない。端末が全文を受け取れていれば、5秒の合図でサーバーは何も預けない。「おわり」の印を付けることで、ふつうに終わった流れと、iPhone がきれいな形で切った流れを見分ける。

実機の iPhone（Safari）から本番に対して29回試した。画面を離れた27回すべてで全文（最長18,545文字）が届き、サーバーが書き終えたあとに戻ったときは、戻ってから0.2〜1.0秒で全文が画面に入った（20回）。端末の記録から、iPhone の切り方が2通りあることがわかった。18回は通信エラー（Load failed）を出して切れ、8回はエラーも「おわり」も出さずに黙って止まった。後者はふつうのエラー処理では拾えず、開き直すまで何も起きない。これを、画面に戻って8秒・表にいる間に30秒なにも届かなければ切る2つの見張りで拾い、本物の WebKit（iOS シミュレータ）で確かめた。シミュレータは裏に回しても通信を切らないため、黙って止まる通信は中継で作った。設計・計測・再現のしかたを記し、iPhone のウェブページではできないこと（裏で流し続ける・書き終えたら知らせる）も明記する。

1 はじめに

AI チャットの答えは、サーバーが書きながら少しずつ送る（ストリーミング）。短い答えなら数秒で終わるが、本番の YUA では、検索を使った答えが1〜2万字になり、書き終わるまで1〜2分かかる。利用者は、その間ずっと画面を見ているわけではない。質問を送って、他のアプリを見て、戻ってくる。

iPhone では、これがうまくいかない。Safari も Chrome も、裏に回ったページを止め、通信を数秒で切る。戻ると、答えは切れたところで止まっている。利用者から見れば「閉じたところで一時停止している」ように見える。これは YUA の作りの問題ではなく、iOS の決まりである。ホーム画面の「アプリ」は裏で動き続けられるが、ブラウザで開くページはそうではない。

本稿の仕組みは、この決まりを変えようとはしない。代わりに、**端末が離れてもサーバーが最後まで書き、書き終えた答えをあずかり、端末が戻ったら受け取る**。似た作りは他社にもある（2節）。本稿で記すのは次の3点である。

- **預かり所が中身を読めない設計**（3節）。鍵は端末だけが持ち、答えは暗号のまま最長1時間あずかり、受け取りは1回きり。端末が全文を受け取れていれば何も預けない
- **iPhone が通信を切る2つの形**（4節）。エラーを出して切る形と、エラーも終わりも出さずに黙って止める形。後者は、記録を「出るはずなのに出ていない出来事」の側から読んで初めて見つかった
- **実機での計測と、再現のしかた**（5・6節）。本番に対する29回の記録と、黙って止まる通信を手元で作る中継、シミュレータでできること・できないこと

2 既存の仕組み

切れた流れの再開。Vercel の AI SDK には「Resume streams」があり [1]、サーバーが答えの流れを Redis などに書き写しながら送り、端末が開き直したときに続きから受け取れる。多くのチャットサービスも同様に、答えをサーバー側に平文で置いて再開する形をとる。本稿の仕組みとの違いは、あずける中身を預かり所が読めないこと、端末が全文を受け取れたときは何も預けないこと、そして「途中から再生する」のではなく「書き終えた全文に置きかえる」ことである（3.4節）。

iOS の裏での制限。ページが裏に回ったことは Page Visibility API の `visibilitychange` で分かり [2]、戻る・進むで同じページが戻ったことは `pageshow` で分かる。しかし、裏に回ったページの通信がいつ・どのように切れるかは、仕様に書かれていない。WebKit は裏のページを止め（Page Lifecycle の freeze [3]）、iOS はしばらくすると通信そのものを閉じる。本稿の4節は、その切れ方を実機の記録から整理したものである。

通知。iOS 16.4 以降、ホーム画面に追加したウェブアプリは Web Push で通知を受け取れる [4]。Safari で開いただけのページでは使えない。本稿の仕組みは通知を使わず、戻ったときに受け取る形に限っている（7.4節）。

3 設計

3.1 求めること

- 端末が途中で離れても、答えは最後まで作られ、戻ったときに全文が届く
- 預かり所（サーバーとは別の保管場所）は、答えの中身を読めない。鍵は端末の外に保存されない
- 端末が全文を受け取れたときは、何も預けない（ふつうの会話は、今まで通り何も残らない）
- 受け取りは1回きり。番号を知っても、鍵がなければ開けない。番号も鍵も使い捨て
- ふつうに終わった流れと、途中で切れた流れを、端末が確実に見分けられる

3.2 全体の流れ

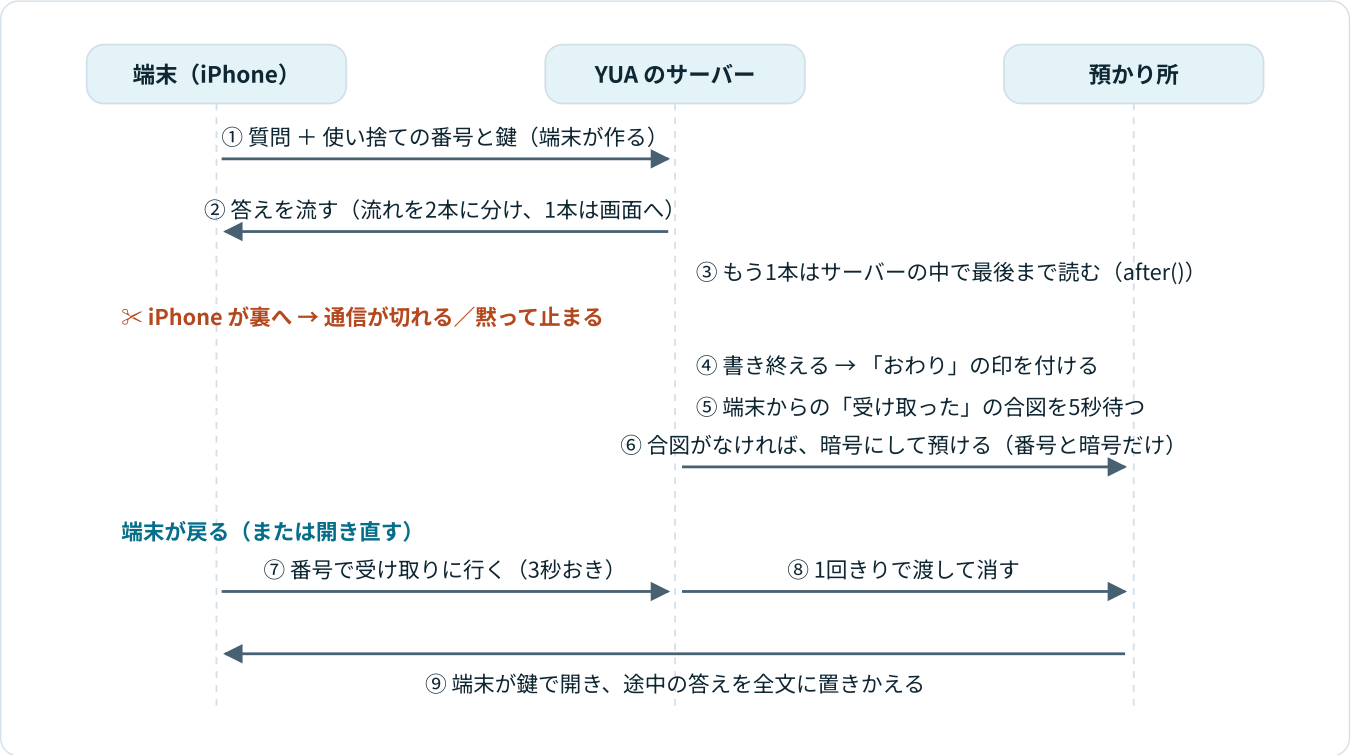


図1　やりとりの流れ。鍵は①で端末からサーバーに渡り、⑥の暗号化にだけ使われて保存されない。預かり所が受け取るのは番号と暗号だけで、⑧の受け取りに鍵は要らない（開けるのは端末だけ）。

3.3 番号と鍵

質問を送るたびに、端末が乱数で受け取り番号（16バイト、128ビット）と鍵（32バイト、256ビット）を作り、質問と一緒にサーバーへ送る。サーバーはこの鍵を暗号化に使うだけで、どこにも保存しない。端末は番号・鍵・どの質問への答えかを、端末の中（localStorage）に控える。控えは最後まで受け取ったら消し、1時間を過ぎても消す。プライベートモードでは番号も鍵も作らず、仕組みは働かない（端末にも何も残さないため）。

3.4 サーバー：流れを2本に分けて最後まで読む

サーバーは答えの流れを2本に分ける（`ReadableStream.tee`）。1本は画面へ流し、もう1本はサーバーの中で最後まで読む。この読み取りは、応答を返したあとも続けられる仕組み（Next.js の `after()` [5]）の中で行うので、端末との通信が切れても続く。書き終えたら、画面へ流す側の末尾に「おわり」の印（`__YUA_END__`）を付ける。

端末は、印が届いたときだけ「最後まで受け取った」とみなし、サーバーへ合図（ack）を送って控えを消す。印が来ないまま流れが終わった場合は、途中で切れたとみなし、途中までの文を保存せず、全文を受け取りに行く。この印が要るのは、iPhone が流れを**エラーではなく、ふつうに終わった形で**切ることがあるためである（4節）。

3.5 預ける前に5秒待つ

サーバーは書き終えたあと、端末の合図を最長5秒待つ（1秒おきに預かり所に「合図は来たか」を聞く）。合図があれば何も預けない。なければ、答えを AES-GCM [6]（12バイトの初期値、16バイトの確かめの印）で暗号にして、番号と一緒に預ける。預かり所は、合図が先に届いていれば預け物を捨て、預けたあとに合図が来れば預け物を消す。どちらの順でも、端末が受け取れた答えは残らない。

3.6 預かり所

預かり所は YUA のサーバーとは別の小さな API（Vercel ＋ Supabase、東京）で、表は番号・状態（預け物／合図済み）・暗号・期限の4列しかない。受け取り（take）は番号だけで行い、渡した瞬間に行を消す（1回きり）。期限は1時間。暗号にすると文字数は約2倍になり、最大580万字（元の答えで300万字）まで受け取る。受け取りの呼び出しは、1人あたり1分に120回・1日3,000回までに抑えている。

3.7 端末：戻ったら受け取る

端末は、ページを開いたとき・画面に戻ったとき（`visibilitychange`・`pageshow`・`online`）・流れが切れたときに、控えにある番号で受け取りに行く。まだ預け物がなければ、画面が見えている間は3秒おきに、送ってから7分まで聞き直す。受け取ったら鍵で開き、質問と同じ会話の、質問のすぐ後ろに全文を保存し、画面の途中の答えを全文に置きかえる。開き直した画面が空の新しい会話なら、待っている答えのある会話を自動で開き、質問の下に「裏で書き続けています」の吹き出しを最初から出す。

送信を始めたあとに切れた控えは、返事が1文字も届いていなくても残す。サーバーが質問を受け取って書いている可能性があるためで、サーバーが断ったとき（回数の上限など）だけ消す。

3.8 通信の見張り

4節で述べる「黙って止まる」通信のために、端末に2つの見張りを置く。画面に戻って8秒のあいだ次の文字が届かなければ、端末から通信を切る（`AbortController`）。表にいる間も、最後の文字から30秒なにも届かなければ切る。切ったあとは、エラーで切れたときと同じ道で全文を受け取りに行く。サーバーは検索などの道具を使う間に10〜20秒黙ることがあるので、30秒はそれより長く取った。間違って切っても、サーバーは最後まで書いて預けるので、答えは少し遅れて全文で届く。

3.9 誰が何を持つか

	端末	YUA のサーバー	預かり所
鍵	作る・控える（1時間）	要求の間だけ使う。保存しない	知らない
番号	控える（1時間）	要求の間だけ。記録に書かない	行の鍵として持つ（1時間）
答えの中身	受け取って保存する	作る側なので平文を扱う。保存しない	暗号のみ。開けない
受け取れたとき	合図を送り、控えを消す	何も預けない	合図済みの印だけ（1時間）
記録に残すもの	なし	「預けた」「受け取りに来た」の1行（番号・中身なし）	なし

表1　鍵・番号・中身を誰が持つか。答えを作る YUA のサーバーが平文を扱うのは当然だが、保存はしない。預かり所は番号と暗号しか持たないため、預かり所が破られても答えは読めない。

3.10 画像・動画・コード

画像は、流れではなく1つの応答なので、サーバーが作り終えた画像（JPEG）をそのまま暗号にして預ける。動画は Google 側で作られ続けるので何も預けず、番号と札を控えて、戻ったときに状態を聞いて受け取る（5秒おき、最長2日）。開発モード（コード）は SSE で流れるため、「おわり」の印を SSE の出来事（`resume-end`）にし、受け取った全文をいつもの再生の仕組みに流し直す。

4　iPhone が通信を切る2つの形

端末に、出来事と時刻だけを書く試し用の記録を一時的に入れ（答えの中身や番号は書かない）、実機で試した記録を読んだ。切り方は2通りあった。

4.1 エラーを出して切る

戻った瞬間に、流れの読み取りが `TypeError: Load failed` で失敗する。ふつうのエラー処理で拾え、そのまま受け取りに行ける。29回のうち18回がこの形だった。

4.2 黙って止まる

戻っても、エラーも「おわり」も来ない。通信はつながったままの形で、ただ何も届かなくなる。端末は「まだ通信中」と思い込み、受け取りにも行かず、画面は途中で止まった答えのまま。利用者が開き直すと、そこで初めて受け取りに行き、全文が届く。8回がこの形だった。次は、その1回の記録である（時刻は日本時間）。

```
21:15:13.9  表  送った（番号 cwgw）
21:15:16.6  裏へ
21:15:16.6  裏  流れが始まった
21:15:19.1  表へ戻った   ← エラーなし。何も起きない
21:15:21.1  裏へ
21:15:55.4  表  開き直した（控え 1件）
21:15:55.4  表  受け取りに行く → 見つかった（送ってから 42 秒）
21:15:55.7  表  全文 4,333 文字が届いた（開き直してから 0.3 秒）
```

この形は、記録に「出ている出来事」を数えているだけでは見つからない。切れたという知らせは一度も出ないからである。「戻ったのに、切れた知らせが出ていない」という、**出るはずなのに出ていない出来事**を探して初めて、8件がまとまって見えた。

4.3 切れないこともある

裏に回って11.9秒で戻った1回は、通信が切れず、そのまま最後まで届いた（送ってから16秒で完了）。一方、2.8秒で戻った回は黙って止まっていた。裏にいた時間だけでは切れ方を予測できず、どちらの形になるかも端末側では選べない。仕組みは、3つのどれが来ても同じ結果になるように作る必要がある。

切り方	回数	端末から見えるもの	必要な手当て
エラーを出して切る	18	読み取りの失敗（Load failed）	ふつうのエラー処理 → 受け取りに行く
黙って止まる	8	何も来ない（エラーも「おわり」も）	見張り（戻って8秒・表で30秒）で切る → 受け取りに行く
切れない	1	そのまま最後まで届く	「おわり」の印を確かめて合図 → 何も預けない

表2　画面を離れた27回のチャットの切り方（画像の1回は流れがないので除く）。黙って止まった8回は、見張りを足す前の作りで起き、すべて開き直しで届いた。

5 実機での計測

5.1 方法

- **端末**：著者の iPhone（Safari）。回線は Wi-Fi と携帯回線の混在で、記録していない
- **相手**：A＝お試しの住所（軽いモデル Gemini 3.5 Flash-Lite、4回）、B＝本番 yua.aquallc.jp（Gemini 3.8 Flash ＋ 検索、見張りを足す前、19回）、C＝本番（見張りを足した最終の作り、6回）。2026年10月25日18時37分〜26日1時55分（日本時間）
- **手順**：質問を送り、すぐ他のアプリに移るか画面を消し、しばらくして戻る。戻る時間・開き直すかどうかは回ごとに変えた。質問は毎回違う
- **記録**：端末の試し用の記録（出来事と時刻。計測後に外した）。時刻は端末の時計。表の「戻ってから」は、最後に表に戻った（または開き直した）時刻から全文が画面に入るまで

5.2 結果

#	相手	種類	答えの文字数	送って裏へ（秒）	最初に戻るまで（秒）	切り方	開き直し	受け取りの回数	戻ってから（秒）	送ってから（秒）
1	A	チャット	3,894	1.8	6.0	エラーで切れた		6	6.6	24
2	A	チャット	2,962	1.7	61	エラーで切れた		1	0.4	63
3	A	チャット	11,683	1.6	131	エラーで切れた		1	0.4	133
4	A	チャット	4,272	1.6	8.1	エラーで切れた		4	0.4	55
5	B	チャット	5,904	1.5	0.3	エラーで切れた		7	0.4	100
6	B	チャット	3,274	—	—	切れなかった		—	—	35
7	B	チャット	4,333	2.8	2.8	**黙って止まった**	要	1	0.3	42
8	B	チャット	14,103	1.5	12	**黙って止まった**	要	1	0.2	81
9	B	チャット	3,776	1.6	12	エラーで切れた		1	0.3	42
10	B	チャット	5,334	2.1	22	エラーで切れた		1	0.4	69
11	B	チャット	18,545	1.7	26	エラーで切れた		6	0.5	111
12	B	チャット	18,535	3.3	2.9	エラーで切れた		1	0.3	378
13	B	チャット	2,368	1.6	12	切れなかった		—	2.8	16
14	B	チャット	17,067	1.8	44	**黙って止まった**	要	4	0.3	96
15	B	チャット	14,128	1.7	41	**黙って止まった**	要	2	0.4	354
16	B	チャット	5,026	3.2	307	**黙って止まった**	要	1	1.0	312
17	B	チャット	14,761	1.7	166	**黙って止まった**	要	1	0.3	168
18	B	チャット	3,758	1.7	62	**黙って止まった**	要	1	0.3	64
19	B	チャット	6,673	1.8	56	**黙って止まった**	要	1	0.3	58
20	B	チャット	4,476	1.3	36	エラーで切れた		1	0.3	38
21	B	チャット	5,566	1.6	86	エラーで切れた		1	0.4	88
22	B	チャット	10,962	1.6	17	エラーで切れた		8	0.4	96
23	B	チャット	3,951	2.3	50	エラーで切れた		1	0.5	52
24	C	チャット	3,721	1.9	98	エラーで切れた		1	0.5	100
25	C	チャット	14,064	1.6	55	エラーで切れた		2	3.8	61
26	C	チャット	4,754	1.5	7.5	エラーで切れた		8	3.5	46
27	C	画像	—	1.9	75	（流れなし）		1	0.6	78
28	C	チャット	5,784	1.6	52	エラーで切れた		1	0.4	54
29	C	チャット	17,502	2.1	124	エラーで切れた		1	0.4	127

表3　実機の29回。「受け取りの回数」は預かり所に聞きに行った回数（1回＝戻った瞬間に預け物があった）。「開き直し」の「要」は、黙って止まった回で、利用者が開き直して初めて届いたもの。#6 は画面を離れなかった回、#13 は離れたが切れなかった回。#12・#15 の「送ってから」が長いのは、利用者が何度も離れて戻ったためで、サーバーの書き終わりではない。

届いたか。画面を離れた27回（チャット26・画像1）すべてで全文が届いた。取りこぼしは0回。利用者が裏に回ったのは送ってから1.3〜3.3秒（中央値1.7秒）で、返事が1文字も届く前に離れる、いちばん厳しい形だった。裏にいた時間は0.3秒から5分までばらついた。

戻ってからの速さ。サーバーがすでに書き終えていた20回では、戻ってから0.2〜1.0秒で全文が画面に入った。17,502文字の答えに2分後に戻った回も0.4秒だった。まだ書いている途中に戻った回は、3秒おきの受け取りで書き終わりを待ち、書き終わってから3〜4秒以内に届いた。

最終の作り（C）。見張りを足した最終の作りでの6回は、6回とも開き直しなしで届いた。この6回はすべて「エラーを出して切る」形だったので、見張りの出番はなかった。見張りそのものの確かめは6節で行った。

5.3 書き終わりまでの時間

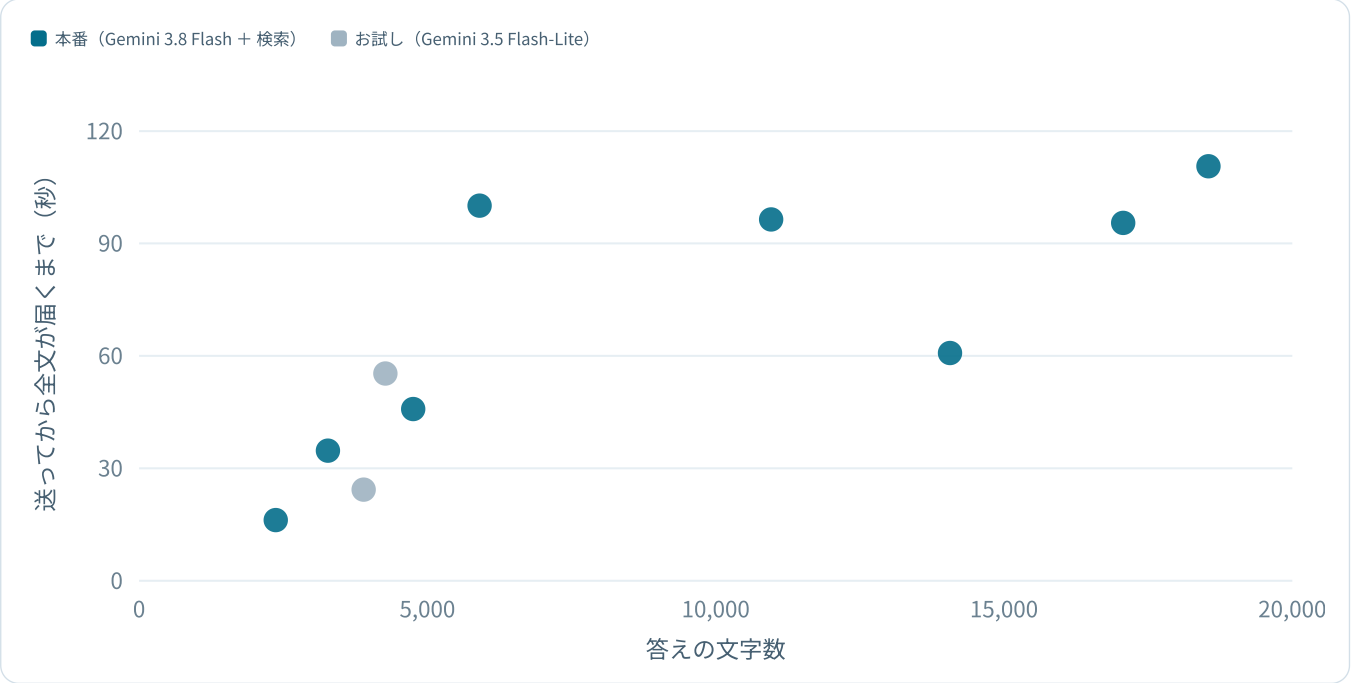


図2　送ってから全文が届くまでの秒数（サーバーの書き終わりが記録から分かる10回）。本番の答えは、1万字を超えると1〜2分かかる。3秒おきの受け取りと5秒の合図待ちを含む。

本番の答えは、2〜6千字で16〜100秒、1万〜2万字で60〜110秒かかった。検索などの道具を使う時間が含まれる。利用者が「閉じたところで止まっている」と感じるのは、この長さの間に裏に回り、戻ったときにまだ書き終わっていない場面である。印の吹き出し（「裏で書き続けています」）は、この待ち時間を「止まっている」ではなく「待てば届く」に変えるためのものである。

6 再現のしかた

6.1 黙って止まる通信を手元で作る

黙って止まる通信は、ブラウザの機能では作れない。Chrome の開発者ツールで「オフライン」にしても、進行中の流れは切れなかった。そこで、開発サーバーの前に小さな TCP の中継を置き、答えの通信だけ、一定のバイト数を流したあとサーバーからの文字を捨てる（つながったまま、端末側は閉じない。サーバーが閉じても端末には伝えない）ようにした。これで、エラーも「おわり」も来ないまま届かなくなる通信ができる。

6.2 シミュレータでできること・できないこと

iOS シミュレータ（iPhone 17、iOS 27.0）の Safari は本物の WebKit なので、画面の出方や戻ったときの出来事は実機に近い。しかし、**裏に回しても通信を切らない**。設定アプリを前に出して Safari を20秒裏に回しても、流れは止まらず、裏にいる間に書き終わりまで届いた。シミュレータは Mac の通信をそのまま使うためで、iOS の切断は再現できない。黙って止まる通信は、6.1の中継で作る必要がある。

試し	環境	作った状況	結果
1	Mac の Chrome	中継で700バイト後に止める → 戻ったことにする	8.1秒で切って印の吹き出し → 全文1,178字
2	シミュレータ Safari	中継で止める → 15秒裏へ → 戻る	戻って8.0秒で切る → 受け取り → 全文3,230字
3	シミュレータ Safari	裏にいるのを5秒だけにし、戻ったあとに止まる（8秒の見張りだけの作り）	**2分たっても何も起きない**（開き直しで届く）
4	シミュレータ Safari	3と同じ（30秒の見張りを足した作り）	戻って8.0秒で切る → 3秒おきに受け取り → 23秒で全文3,743字
5	シミュレータ Safari・本番	送って7秒で Safari を終了 → 30秒後に開き直す	会話が自動で開き、6秒で全文4,944字
6	シミュレータ Safari・本番	ふつうの流れを20秒裏へ	切れずに最後まで届く。見張りは誤って切らない

表4　手元での再現。試し3で、8秒の見張りだけでは「戻ったあとに止まった通信」を拾えないことが分かり、30秒の見張りを足した（試し4）。試し6は、見張りが元気な通信を誤って切らないことの確かめ。

6.3 記録の読み方

実機の記録は、出来事の有無で読む。「戻った」のあとに「切れた」も「終わった」も「受け取りに行った」も無い区間があれば、それが黙って止まった通信である。今回は、届いた件数だけを数えて「全部届いた」と結論しかけ、開き直しに頼っていた8件を見落とした。出来事の記録を取るときは、**出るはずの出来事が出ていない区間**を機械的に探すのがよい。

---

7 考察

7.1 なぜ「途中から再生」ではなく「全文に置きかえる」か

再開の仕組みの多くは、切れた位置から続きを流す。本稿は、途中までの文を捨て、書き終えた全文に置きかえる。理由は2つある。iPhone がきれいな形で切ると、端末には「どこまで届いたか」が正しく分からない（届いたように見えて、実は途中である）。また、答えの末尾には出典や印が付くので、途中の文を保存すると壊れた答えが残る。全文に置きかえる形なら、どの切り方でも結果は同じになる。

7.2 5秒の合図待ちの意味

ふつうの会話（端末が最後まで受け取った）で何かを預けると、預かり所に答えの暗号が毎回残る。5秒待って合図があれば預けないので、預かり所に残るのは、実際に切れた会話の分だけである。本番の記録では、書き終えて合図が届いた回は「預けない」、届かなかった回だけ「預けた」になっていた。預かり所の中身を最小にすることは、守りの設計の一部である。

7.3 見張りの時間の決め方

8秒と30秒は、間違って切ったときの損と、拾えないときの損の釣り合いで決めた。間違って切っても、サーバーは最後まで書いて預け、端末は書き終わりから数秒で全文を受け取る。失うのは「文字が流れてくる見た目」だけである。一方、拾えなければ、開き直すまで何も起きない。本番のサーバーが道具を使う間に黙る時間（10〜20秒）を測ってから、30秒とした。

7.4 ウェブページにできないこと

裏に回ったまま文字を流し続けることと、書き終えたときに通知で知らせることは、Safari で開いたページではできない。前者は iOS の決まりで、後者はホーム画面に追加した形（Web Push）でのみ可能である。本稿の仕組みは、「戻ったときに、待たせずに全文がある」ことを目標にし、それ以上は約束しない。

---

8 限界

- **端末は1台、試したのは著者1人**：iPhone 1台の Safari で29回である。iOS の版・回線・端末の違いで、切り方の割合は変わりうる。Android は測っていない
- **切り方の割合は条件に依る**：18回・8回・1回という数は、「送って2秒で裏へ回る」という著者の試し方での数で、利用者の実際の使い方での割合ではない
- **黙って止まる形の仕組みは不明**：iOS のどの層で通信が止まるのかは分かっていない。本稿は、端末から見える振る舞いと、その手当てだけを記した

- **時刻は端末の時計**：秒数はすべて端末の記録から求めた。サーバーの書き終わりの時刻は、受け取りが見つかった時刻で代用している（最大で3秒＋5秒の遅れを含む）
- **見張りの確かめはシミュレータと中継**：実機の最終の6回は、たまたまエラーで切れる形だったので、見張りが実機で働く場面は記録にない。見張りの動作は本物の WebKit で確かめたが、iOS の切断そのものは中継で作ったものである
- **試し用の記録が入っていた**：計測中は端末に出来事の記録を書く部品が入っていた。計測後に外したが、記録の有無で振る舞いが変わらないことは、外したあとの手元の確認でしか見ていない

---

9 まとめ

iPhone で画面を離れると止まってしまう AI チャットの長い答えを、サーバーが最後まで書き、中身の読めない預かり所に暗号のままあずけ、戻ったときに1回きりで受け取る形にした。実機の29回で、画面を離れた27回すべてに全文が届き、書き終えたあとに戻れば1秒以内に画面に入った。

実機の記録から、iPhone が通信を切る形は2通りあることが分かった。エラーを出して切る形と、エラーも終わりも出さずに黙って止まる形である。後者はふつうのエラー処理では拾えず、戻って8秒・表で30秒の見張りで切って受け取りに行く必要があった。シミュレータは iOS の切断を再現しないので、黙って止まる通信は中継で作って確かめた。

仕組みの新しさは小さい。記す価値があるのは、預かり所が中身を読めない設計と、実機でしか見えなかった切り方、そして「出るはずの出来事が出ていない区間」を探す記録の読み方である。

参考文献

1. Vercel. “Resume streams.” AI SDK UI documentation. ai-sdk.dev/docs/ai-sdk-ui/chatbot-resume-streams
2. W3C. “Page Visibility Level 2.” w3.org/TR/page-visibility-2/
3. WICG. “Page Lifecycle.” wicg.github.io/page-lifecycle/
4. B. Rego. “Web Push for Web Apps on iOS and iPadOS.” WebKit Blog, 2023年2月. webkit.org/blog/13878/
5. Vercel. “after.” Next.js documentation. nextjs.org/docs/app/api-reference/functions/after
6. NIST. “Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.” SP 800-38D, 2007.

データ：実機の29回の記録（出来事と時刻だけ。答えの中身・番号は含まない）と、集計のプログラムは、公開の準備ができ次第、他の報告と同じく GitHub と Zenodo に置く。