PROTOCOL SYMPOSIUM 2026

Blygger: AI-intertwingled diachronic-synchronic social publishing

Venkatesh Rao

24 September 2026

SLIDE 01 OPENING

What is Blygger

- A decentralized, p2p, AI-native social publishing protocol
- A **minimal** update to the blogging and old-Twitter experiences — not a new one
- Degrades gracefully: legacy RSS readers see an ordinary feed and lose nothing
- Some inspiration from Robin Sloan's Spring '83
- Fixes the **synchronic/diachronic temporality tension** in traditional media — loosely, "Twitter time" vs. "GitHub time"
- Supports fine-grained generative AI use at the **block** level

Etymology

- **Blyg** — Swedish for "shy", and nearly homophonic with "blog"
- **Blygger** — a hat-tip to Blogger
- **ygg** — a gesture at Yggdrasil, the Norse cosmic tree of life
- **Why these two** — near-miss spellings of blog/blogger, and both domains were free
- **blygger.org** is the protocol home; **blygger.com** is a future directory

SLIDE 03 OPENING

Intelligence media, in time

- **From destination AI to intelligence media** — intelligence stops being a place you visit and starts circulating as work-in-progress artifacts, metabolized as they move (Protocolized)
- **AI today lacks liveness.** Blygger is an attempt to add it
- **AI is atemporal and superhistorical. Blygger brings it into time** (Superhistory, Not Superintelligence, 2021)

SLIDE 04 OPENING

Demo

SLIDE 05 OPENING

Two kinds of time

- **Diachronic** — time as a *sequence of states*; the changing is the point
  - a git log, a blog series, a book's editions, a changelog
- **Synchronic** — time as a *present moment*; the snapshot is the point
  - a Twitter feed, a group chat, a livestream

SLIDE 06 OPENING

Handling both at once is hard

- **Threads are clumsy** — a synchronic medium doing an impression of a diachronic one
- **Editable tweets break provenance** — if it can change silently, what did you actually say?
- **Nobody reads changelogs** — a diachronic record with no synchronic surface
- **Citation messiness** — diachronic documents make synchronic citation messy
- Each fix for one property damages the other

SLIDE 07 OPENING

Books solve this by being very, very slow

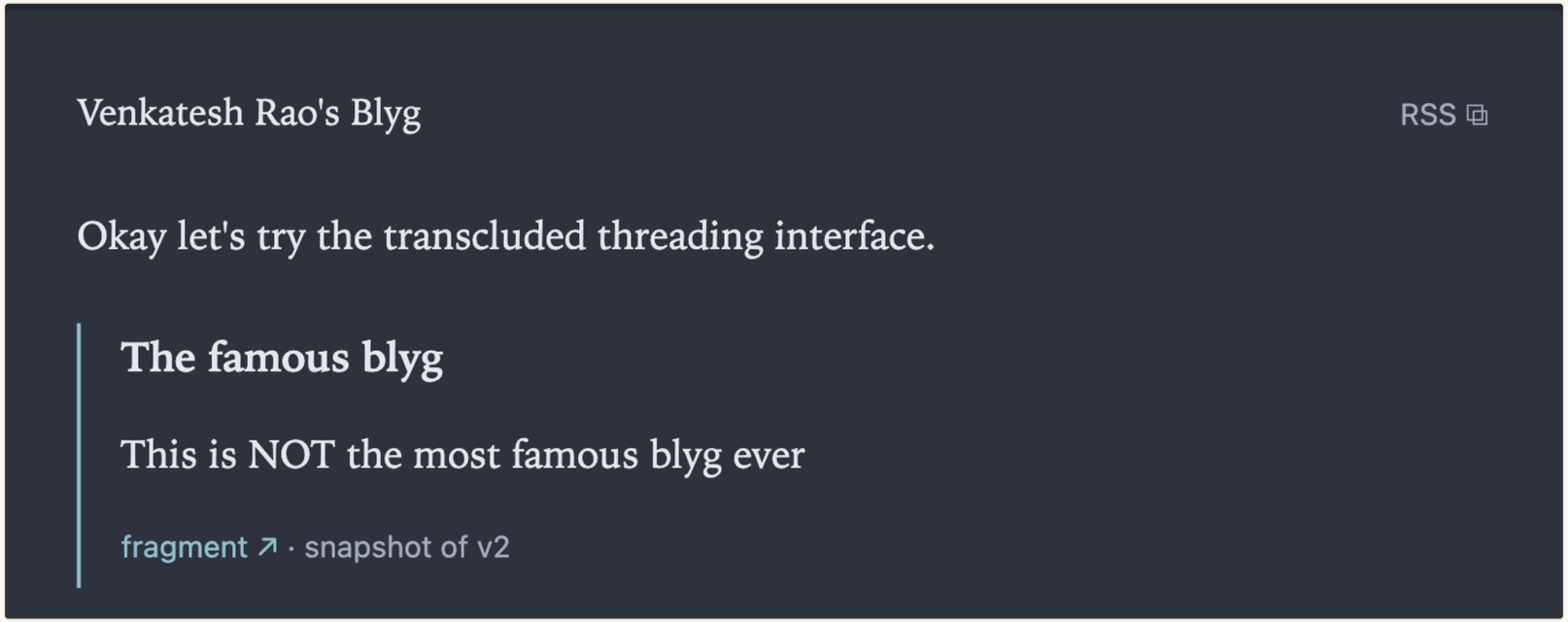
- An **edition** is a diachronic event a synchronic reader can actually see
- It works *because* editions are rare — years apart, a handful per book
- You cannot run a conversation at that tempo
- So: keep the edition, drop the latency

A blyg is fragments and threads



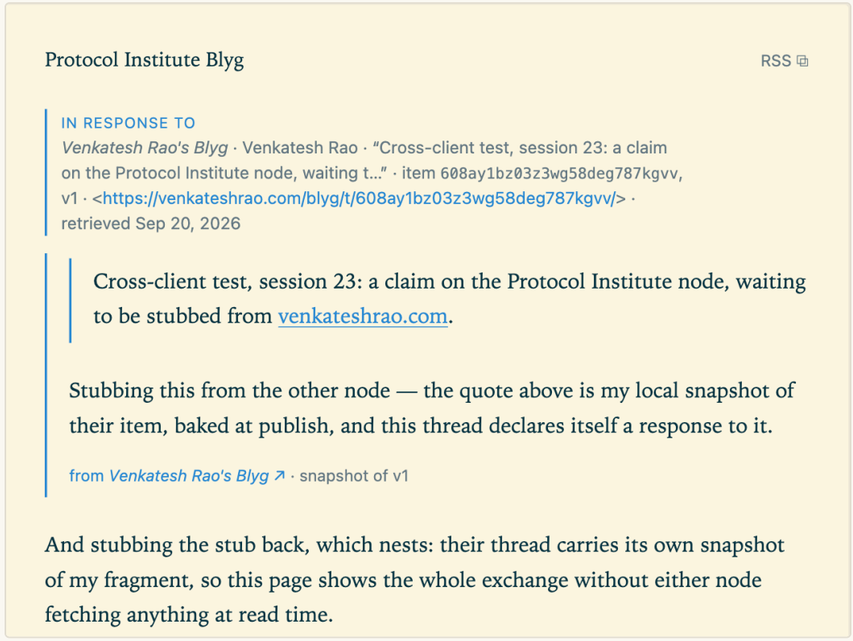
- Two units, not one: a **fragment** (a note) and a **thread** (fragments composed)
- Every item wears its own revision history in public — `v3 · pinned: v1, v3`
- A real page, live at venkateshrao.com/blyg/

Transclusion: quote by reference



- Write `![[id]]` on its own line — you get the item, not a link to it
- Snapshotted at publish: the reader gets the bytes you saw, forever
- Provenance under every quote: which item, which version
- Editing the source later never rewrites the quote

Quoting across origins



Protocol Institute Blyg RSS

IN RESPONSE TO
Venkatesh Rao's Blyg · Venkatesh Rao · "Cross-client test, session 23: a claim on the Protocol Institute node, waiting t…" · item `608ay1bz03z3wg58deg787kgvv`, v1 · <https://venkateshrao.com/blyg/t/608ay1bz03z3wg58deg787kgvv/> · retrieved Sep 20, 2026

> Cross-client test, session 23: a claim on the Protocol Institute node, waiting to be stubbed from venkateshrao.com.
>
> Stubbing this from the other node — the quote above is my local snapshot of their item, baked at publish, and this thread declares itself a response to it.
>
> from *Venkatesh Rao's Blyg* ↗ · snapshot of v1

And stubbing the stub back, which nests: their thread carries its own snapshot of my fragment, so this page shows the whole exchange without either node fetching anything at read time.

- Same `![[id]]`, now naming an item on **someone else's** blyg
- It names an *identity*, not an address — resolved against what you've read
- **The snapshot is local**: publishing never touches the network
- A **stub** declares what it answers; a Webmention tells them, and they check

Quoting what you've already read

- **Ids are global** — 128 random bits, so an id needs no address to be unique
- You can only quote what you have **already imported** from a subscription
- Resolution: your own items first, then your imported copies. Never a fetch
- Same id from two origins → **publish fails**, rather than guessing
- On the wire, one difference: `transclusions[]` gains an `origin`
- **Not a fork** — a fork starts *from* their bytes; this quotes them

A list, never a count



- Verified responses can show on the item they answer — off by default, per item
- A **list**: who, which origin, what relation, when. **No number, anywhere.**
- A row costs a real item on a real origin that passed verification
- Nothing a stranger publishes changes your bytes — chrome, not content

Glossary: structure

- **blyg** — one author's published surface: files, feed, manifest
- **item** — the umbrella; a fragment or a thread, each with one id
- **fragment** — the small unit; one note
- **thread** — fragments composed into one item; the only kind that transcludes
- **id** — 128 random bits; names an *identity*, never an address
- **version** — a bare counter, +1 per publish. No semver, ever
- **pin** — an irrevocable promise to serve one version's bytes forever
- **withdrawal** — the content taken back; the id, history and pins remain

Glossary: acts and network

- **transclusion** — quote by reference; their bytes baked in at publish
- **TK** — an instruction left in the draft for a model to fill
- **stub** — a thread declaring the one thing it responds to
- **fork** — a new draft descending from a pinned version
- **origin** — the base URL a blyg is served from; part of every citation
- **webmention** — a W3C ping saying "I referenced you"; then they verify
- **RSS** — the feed; a legacy reader sees it and loses nothing
- **studio / page** — private tool vs public artifact; only the page is governed

[TK]: leave an instruction in the draft

- `[TK]expand this into a paragraph[/TK]` — an instruction, not yet run
- `[TK]expand this[=]…generated prose…[/TK]` — after you run it
- **This is the intertwingling** — the instruction lives *in* the draft
- Inside a TK scope, `![[id]]` means **source**, not quote — material to work from
- Generation is always an explicit act you review. Publishing never generates.
- The markers are studio-private. They never reach the wire.

Which paragraph was generated?

Protocol Institute Blyg RSS

Running list of links about the origins of life

Cool article about how mitochondria became symbiotic with archaea to create eukaryotes — the merger that gave us complex cells.

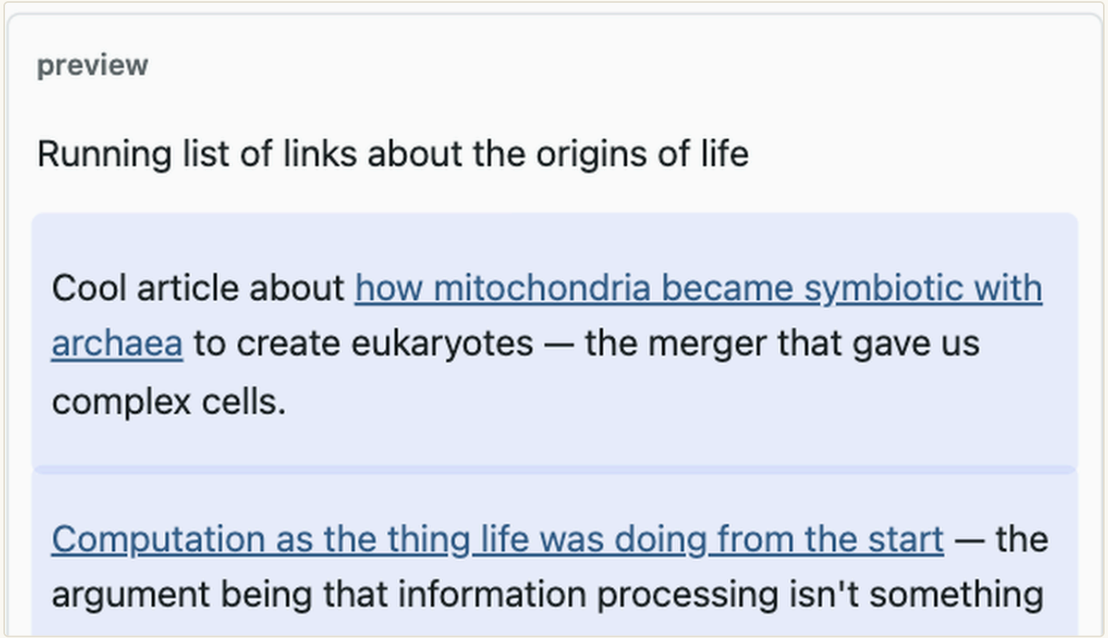
Computation as the thing life was doing from the start — the argument being that information processing isn't something biology invented later, it's what the earliest chemistry was already up to. Two things worth holding onto: first, that the line between "chemistry" and "computation" is mostly one we drew for our own convenience; second, that if we want to understand the beginnings of *new* nature — synthetic, artificial, whatever we end up calling it — we have to get the beginnings of the old one right first. The origin story constrains the sequel.

- A real published page. **How much of this did a model write?**

You can't tell. That is deliberate.

- The item JSON says: `generated[{sources, model, at}]` — two scopes, claude-opus-5
- The HTML carries a `blyg-tk-gen` class you can inspect. It is deliberately unstyled.
- Disclosure is total. It just isn't decorative.
- A visible tint would present a self-asserted claim as a verified badge

Writing it



- The same page you just saw, in my editor — **the same bytes**
- The studio tints what was generated. The published page doesn't.
- I need to see it. A reader can't verify it. Two surfaces, two obligations.
- The protocol governs only the page. Everything else here is my client's choice.

It's just files

- A blyg is a directory: `blyg.json`, `feed.xml`, `items/*.json`, a page per item
- No database in the protocol. Copy the directory to any web host and it still works.
- Root, subdirectory, or subdomain — the protocol never assumes where you put it
- We check this by exporting the whole blyg to flat files and diffing

It's still RSS

- Every blyg feed is a valid RSS feed. Your existing reader works today.
- A reader that's never heard of blygger sees a normal feed and loses nothing
- A blyg can subscribe to any RSS or Atom feed — no cooperation required
- So your blog is already half in: it can be read, it just can't be quoted

You don't get a username

- No accounts, no handles, no `@you@server`. There is no namespace to be in.
- Your domain is your name. DNS is the namespace — it already exists and works.
- Author names are decoration: optional, per-item, the protocol promises nothing
- The only thing the protocol authenticates is the origin the bytes came from
- So: no follower lists, no counts, no follow requests. Following is client-local and invisible.

Nothing is deleted. Pins are forever.

- No hard delete. The single exit is **withdraw** — a permanent, visible endcap.
- A **pin** is an irrevocable promise to host one exact version, forever
- Pins survive withdrawal. That is the point of a pin.
- Unpinned history is unreachable in every representation — no route serves it
- Version numbers are a bare counter. Significance is the pin, not the number.
- **You can fork a pinned version** — anyone's — and your item says so, permanently

The wire, for protocol people

- Two planes: `items/{id}.json` is state, `feed.xml` is only notification
- **The index reconciles, not the feed** — a gap degrades to a diff, not loss
- AP under partition: convergence ~25 min observed, ~45 worst case
- Version watermarks never silently regress — a drop is surfaced, not applied
- IDs: stable random 128-bit, never content-addressed — identity survives edits
- Conformance L0–L3, strict supersets. Unknown constructs ignored.

What is deliberately not enshrined

- **AI.** Generation is studio-side; the wire gets output plus provenance.
- **Identity.** Opaque, client-asserted, never addressable.
- **Editorial convenience.** No reply primitive, ever — transclusion is the primitive.
  - Effectively equivalent to "quote dunking" as the only response
- **Significance markup.** No semver, no edition field. Pins carry weight instead.
- Four separate decisions, months apart, same sentence each time.

SLIDE 25 ROADMAP

Where it is, where it's going

- Shipped: publish, subscribe, generation, and **cross-client threads (v0.3)**
- Running: two live nodes, real pub-sub, legacy RSS — and the stub stack you saw
- Writing now: the normative 0.3 document, from shapes already tested in public
- Then: self-host template, filter plugins, staleness over the DAG, then 1.0
- **Pre-1.0 means no promises** — including the wire. Nothing is stable yet.

If you write

- Nothing to install yet, honestly — the self-host template is v1.0 RC work
- What's useful now: tell me where the fragment/thread split feels wrong
- Especially: does writing with `[TK]` change *what* you write, or just how fast?
- Read the two live blygs; both are real writing, not fixtures

If you publish

- You already have the hard part: a domain nobody can take from you
- Your RSS feed already makes you readable from a blyg — no action needed
- A blyg mounts anywhere: yoursite.com/blyg/, blyg.yoursite.com, or the root
- Ask of you: if you'd never run this, say why — that's the most useful failure report

If you build

- Spec: blygger.org/spec/0.2/ — standalone, complete, no deltas to chase
- The wire is already at 0.3; that document is being written now
- Reference client: MIT, Cloudflare Workers, ~500 tests, documented CSS contract
- The open gap: a **second implementation** — local-first, folder-based, static
- **It is the 1.0 gate. It is undesigned. It could be yours.**

Resources

- Live: venkateshrao.com/blyg/ · blyg.protocol-institute.org
- Directory: blygger.com — submit a feed; it's validated by running the resolver at it
- Spec: blygger.org/spec/0.2/ · Notes: blygger.org/notes/ · Namespace: blygger.org/ns/0.1
- Code: github.com/blygger — MIT (code), CC-BY-4.0 (docs)
- This deck, with cues, is at the URL on screen