BCQM VII Stage–2 Cloth: A Plain-Language Walkthrough From events and threads to bin-coarsened persistence tests (v0.1)

Peter M. Ferguson
Independent Researcher

30 January 2026

How this note relates to the previous one

This note is a continuation of the previous checkpoint write-up. The earlier checkpoint captured the formal result of the bin-coarsening experiment. Here we explain, at a plain-language level, what we did and why, in a way intended for a reader who has not read any BCQM papers.

What problem we are trying to solve

BCQM is a programme that treats *spacetime as emergent*. Rather than assuming a background manifold with coordinates, one starts from a pre-spacetime primitive: **events** (featureless nodes) and **directed connections** (edges) produced by the evolution of **threads**. A thread is a minimal dynamical entity that advances step-by-step, selecting the next event according to a rule. When multiple threads share or re-use events, they create cross-links, and a nontrivial event graph builds up.

Stage–1 (BCQM VI) showed that a short, moving "active slice" of this graph can become connected (*space on*) while coherent "islands" can remain intermittent (*islands fluctuate*). It also showed that the active slice mixes rapidly and can behave like "channels + shortcuts". This means that some standard continuum diagnostics (for example, dimension estimates from random-walk return probabilities) can be structurally unreliable at that scale.

Stage–2 asks: if the active slice is "yarn" (short, local, fast-mixing), can we construct a more persistent object—a **cloth**—from long runs, and then ask metric/dimension-style questions on that cloth?

The primitives in one minute

- **Events:** nodes in a directed graph; no coordinates are assumed.
- **Threads:** each thread advances by selecting a next event; this produces directed edges as it goes.
- **Cross-links (Path A):** a thread can either create a fresh event or re-use an existing event from a current active window. Re-use creates additional adjacency and can drive percolation (connectivity).
- **Glue / lockstep:** when threads become synchronised (in phase and/or cadence), they form a coherent bundle. In Stage–1, this provides an operational "clock quality" observable.

What "persistence" means in Stage–2

A persistent cloth should not be a one-off snapshot; it should be extracted from repeated behaviour. The simplest idea is to treat "persistence" as repeated use of the same structures

over time. However, there is a trap: if we store weights on the events themselves (traffic counts, edge weights), we can quietly drift away from a minimal primitive ontology.

So we explored minimalism-preserving persistence proxies:

- **Lockstep-supported persistence:** define a "core" set of threads (those in the dominant coherent bundle) and focus on what they repeatedly use.
- **Survival-under-perturbation:** do not trust a single run; stability must be demonstrated across seeds and small perturbations.
- **Concurrency:** within a finite time bin, if multiple threads traverse the same directed transition, that indicates short-run channelisation. Concurrency can be logged without mutating primitives.

Bin-based cloth extraction

We chose bin-based processing as the first implementation because "tick" is too fine: it captures microscopic jitter rather than stable structure.

In each run we divide the measurement interval into a fixed number of bins (initially 80). Within each bin we record:

- which **events** were used, and by how many threads;
- which **directed edges** $(u \to v)$ were traversed, and by how many threads;
- a split between **all threads** and **core (lockstep) threads**.

From those per-bin records we derive two distinct signals:

1. **Concurrency (channel activity):** an event/edge is "concurrent" in a bin if its count is at least 2 in that bin.
2. **Bin-hits (route carving):** an event/edge is "persistent" if it appears in at least $h_{\min}$ bins (min_bin_hits). This is a long-run criterion.

A useful mental model

Concurrency is closer to *channel activity* than to *cloth*. The hierarchy that emerged from the experiments is:

- **Edge concurrency** ⇒ "channel is active now" (short-run funneling).
- **Used-by-core bin hits** ⇒ "channel is carved" (routes repeated across bins).
- **Cloth core** ⇒ "carved channels form a stable background" (persistence across longer epochs / stricter thresholds).

At limited sampling, it is unsurprising that concurrency does not immediately yield a stable cloth backbone: cloth is a long-run stabilisation object.

What we tried first, and what it taught us

Concurrent-only edges are too sparse

We initially tried to define an edge cloth using *concurrent edges only* (edges used by at least two threads within the same bin). Empirically, this signal is real but sparse. It tends not to repeat across bins at the current scales, so strict persistence requirements eliminate it.

Refinement: "edges used by the lockstep core"

We therefore refined the edge cloth definition:

- Concurrency is retained as a reinforcement diagnostic.
- The edge cloth core is defined from **edges used by core threads**, counting per-bin presence ("bin hits").

This produces connected edge cloths under permissive persistence (hits1), while preserving the minimalism story (no weights stored on primitives).

How we evaluated "stability"

Two different notions of stability were measured across seeds:

1. **Set stability (Jaccard):** do the exact core sets match across seeds?

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

   We compute this for the core event set and the core edge set.
2. **Metric stability (ball growth):** do the *geometry diagnostics* match across seeds even if the exact edges differ? We compare normalised ball-growth curves $|B(r)|/|C|$ using an L2 distance between curves.

What the long-epoch and bin-coarsening experiments showed

We ran long-epoch ensembles (x5 and x10) and then coarsened binning to see whether strict persistence failures were simply a binning artefact.

Key outcome

- A permissive definition (hits1) yields a **connected cloth** and **extremely stable geometry diagnostics** (ball-growth curves) at high cross-link pressure. Exact edge identity remains seed-sensitive, but the geometry class is stable.
- A strict definition (hits2) yields **tiny recurrent pockets** (motifs) rather than a spanning cloth, even at longer epochs and with bin coarsening. This behaviour is not primarily a binning artefact; it is a property of the strict persistence rule at this scale.

Where this leaves Stage–2

At the current scale, strict cross-bin repetition (hits2) behaves as a *motif detector*, while hits1 behaves as a workable *cloth baseline*. This is not a failure: it tells us that stability emerges first at the level of *geometry diagnostics* rather than at the level of exact microstructure.

Immediate next directions

There are two minimalism-preserving ways forward if a stricter-than-hits1 cloth backbone is required:

1. **Event-filtered edge cores:** define persistent events first, then define edge cloth only between persistent events.
2. **Quantile persistence:** keep the top $q\%$ most-used core edges over an epoch, rather than requiring explicit cross-bin repetition.

Both approaches aim to stabilise edge cloth structure without writing mutable weights into primitives.