

StreamSync MVP v0.1

1. Overview

What StreamSync is, the problem it addresses, and the narrow question v0.1 is designed to test.

DOCUMENT PURPOSE

Define the behavior, test surface, and scope boundary of the first independent StreamSync reference implementation.

MVP QUESTION

Under the same noisy-stream conditions, how does adaptive collective synchronization compare with a conventional reference-stream synchronizer in maintaining and recovering temporal coherence?

What StreamSync is

StreamSync is an executable synchronization-architecture demonstration for multiple time-varying streams. The current API accepts observed phases and optional per-stream availability weights, estimates the population synchronization state, updates a bounded coupling value, and returns a derived correction/next-phase plan. Upstream sensing logic may translate timing, jitter, loss, or delay information into those phase and influence inputs.

The problem

Real streams rarely share a perfect clock or remain equally reliable. Devices drift, observations arrive irregularly, packets disappear, and one modality or receiver may degrade while others remain usable. A useful synchronizer therefore has to do more than correct one measured offset: it must decide how strongly streams should influence one another, avoid runaway reaction to noise, and recover coherently after disruption.

The firefly / Kuramoto insight

StreamSync models each stream as an oscillator with a phase and natural frequency. Like fireflies adjusting their flashes in response to neighbors, the oscillators influence one another through collective coupling. The Kuramoto family supplies the mathematical starting point: coherence can emerge from local phase adjustments even when individual streams are heterogeneous. In v0.1, coupling strength K is bounded and observation availability can reduce or remove a stream's influence.

What v0.1 is testing

- **Acquisition:** initially misaligned streams should move toward higher coherence.
- **Stability:** ordinary jitter and drift should not cause uncontrolled or oscillatory reaction.
- **Resilience:** missing observations and temporary reference loss should degrade gracefully.
- **Relock:** after disturbance, the population should return to a stable coherent relationship.
- **Explainability:** outputs should be traceable to measured phase state, coherence, availability, and bounded coupling.

MVP boundary

Version 0.1 is an architecture demonstration and benchmark instrument, not a production transport, codec, RF stack, satellite system, audio separator, or multimodal-fusion product. It begins with deterministic synthetic tests, exposes a deliberately small REST surface, and makes no production latency, throughput, or downstream domain-performance claim.

StreamSync MVP v0.1

2. MVP behavior + success criteria

Qualitative behavior, benchmark metrics, and the evidence threshold for claiming that the architecture is behaving sensibly.

SUCCESS PRINCIPLE Evidence before interface. The engine and benchmark must demonstrate useful, bounded behavior independently of FastAPI.	LATENCY POSITION Computation time is recorded only as an engineering sanity check. StreamSync v0.1 claims no production latency target.
---	---

Behavior under disturbance

Test	Input / disturbance	MVP success criterion
Initial convergence	Different starting phases	Coherence rises; phase error falls; the population reaches a stable relationship.
Clock drift	Different natural frequencies / slow drift	Drift does not produce persistent divergence; adaptation remains bounded.
Ordinary jitter	Moderate random phase noise	Coherence remains high without chasing every perturbation.
Jitter excursion	Sudden high-noise interval	Coherence may fall temporarily, then recover without thrashing.
Missing observations	Partial loss / availability weights near zero	The system degrades gracefully; unavailable streams contribute less or not at all.
Phase displacement	Sudden phase jump	A measurable disturbance is followed by recovery/relock.
Reference dropout	The baseline's privileged reference disappears	Collective coupling should preserve useful coherence without requiring a permanent master stream.

Metrics retained

- **Coherence r**: Kuramoto order parameter, from 0 (dispersed) toward 1 (highly coherent).
- **RMS phase error**: population spread around the current mean phase.
- **Convergence / relock**: time to sustained coherence threshold in the benchmark harness.
- **Correction magnitude**: RMS and p95 correction size, used to detect overly aggressive behavior.
- **Computation time**: sanity check only, not a product claim.

What the current evidence says

The benchmark does not support a universal superiority claim. A healthy fixed-reference synchronizer is a strong baseline and wins several alignment/convergence metrics. StreamSync's clearest observed advantage appears under complete reference dropout: across five paired seeds, stress mean/min coherence were 0.999/0.999 versus 0.996/0.991 for the baseline, with lower stress RMS phase error (0.041 vs 0.078 rad) and smaller p95 corrections (0.003 vs 0.007 rad). Under reference jitter, StreamSync used much smaller p95 corrections (0.005 vs 0.055 rad) while the baseline retained better internal alignment.

INTERPRETATION The emerging characteristic is reference-tolerant collective coherence with restrained corrective behavior—not “better synchronization everywhere.”
--

StreamSync MVP v0.1

3. POST /v1/sync/session

Create an ephemeral synchronization session and define the oscillator population.

Purpose

Creates in-memory session state for a population of streams. The route initializes phases, natural frequencies, timestep, adaptive controller state, and default availability. It returns the initial synchronization state and current coupling value.

Request fields

Field	Type	Required	Default	Meaning
n_streams	integer	No	8	Number of streams; valid range 2–256.
dt_s	number	No	0.02	Simulation/update timestep in seconds; >0 and ≤10.
initial_phases_rad	array[number]	No	deterministic evenly spaced	One phase per stream; values are wrapped to $[0, 2\pi)$.
natural_frequencies_rad_s	array[number]	No	all 1.0	One natural angular frequency per stream.

Example request

```
{
  "n_streams": 4,
  "dt_s": 0.02,
  "initial_phases_rad": [0.0, 0.7, 1.4, 2.1],
  "natural_frequencies_rad_s": [1.0, 1.01, 0.99, 1.0]
}
```

Example response

```
{
  "session_id": "<uuid>",
  "n_streams": 4,
  "dt_s": 0.02,
  "coherence_r": 0.7185,
  "mean_phase_rad": 1.0500,
  "rms_phase_error_rad": 0.7826,
  "lock_state": "converging",
  "coupling_k": 0.6000
}
```

Notes

- Sessions are process-local and ephemeral. There is no database or production persistence in v0.1.
- If phases or frequencies are supplied, their lengths must match n_streams and values must be finite.
- The deterministic defaults are convenient for reproducible demos, not a recommendation for production initialization.

StreamSync MVP v0.1

4. POST /v1/sync/observe

Submit the latest phase observations, observation availability, and optional timestep update.

Purpose

This is the main observation doorway. It measures the current population state, updates the bounded adaptive coupling K , computes a Kuramoto-coupled next state, compares that state with free-running evolution, and returns the resulting per-stream phase-correction vector.

Request fields

Field	Type	Required	Meaning
session_id	string	Yes	Existing ephemeral session identifier.
observed_phases_rad	array[number]	Yes	One current observed phase per stream; exact session length required.
availability	array[number]	No	Per-stream influence weights in $[0,1]$. 0 means unavailable; intermediate values reduce influence.
dt_s	number	No	Optional update timestep; otherwise session timestep is reused.

Example request

```
{
  "session_id": "<uuid>",
  "observed_phases_rad": [0.1, 0.6, 1.2, 1.8],
  "availability": [1.0, 1.0, 0.5, 1.0]
}
```

Example response

```
{
  "session_id": "<uuid>",
  "coherence_r": 0.8074,
  "mean_phase_rad": 0.9204,
  "rms_phase_error_rad": 0.6379,
  "lock_state": "converging",
  "coupling_k": 0.6025,
  "recommended_corrections_rad": [0.00659, 0.00253, -0.00306, -0.00759]
}
```

Reference loss is representable directly

Example: availability = [0.0, 1.0, 1.0, 1.0] marks stream 0 unavailable while the remaining streams continue to contribute to the coupled update. This is the API-level representation used to expose the architecture's reference-dropout behavior.

BOUNDARY

The endpoint does not itself ingest packets, media frames, RF samples, or raw jitter traces. Upstream domain logic must convert those observations into phase and availability inputs.

StreamSync MVP v0.1

5. GET /v1/sync/plan

Read the current synchronization state and the most recent derived correction / next-phase plan.

Purpose

Returns the current state of a session after the latest observation. The plan is an inspectable synchronization output: it exposes population coherence, individual phase errors, adaptive coupling, the latest correction vector, predicted next phases, and observation availability.

Query

```
GET /v1/sync/plan?session_id=<uuid>
```

Response fields

Field	Meaning
coherence_r	Current Kuramoto order parameter.
mean_phase_rad	Circular population mean phase; null only when the mean is undefined.
phase_errors_rad	Per-stream signed circular error relative to current mean phase.
rms_phase_error_rad	RMS population phase spread.
lock_state	Simple unlocked / converging / locked classification derived from coherence.
coupling_k	Current bounded adaptive coupling value.
recommended_corrections_rad	Most recent correction vector implied by the coupled step versus free running.
predicted_next_phases_rad	Most recent Kuramoto-coupled next-phase prediction.
availability	Most recently supplied per-stream influence weights.

Example response

```
{
  "session_id": "<uuid>",
  "coherence_r": 0.8074,
  "mean_phase_rad": 0.9204,
  "phase_errors_rad": [-0.82043, -0.32043, 0.27957, 0.87957],
  "rms_phase_error_rad": 0.6379,
  "lock_state": "converging",
  "coupling_k": 0.6025,
  "recommended_corrections_rad": [0.00659, 0.00253, -0.00306, -0.00759],
  "predicted_next_phases_rad": [0.12659, 0.62273, 1.21674, 1.81241],
  "availability": [1.0, 1.0, 0.5, 1.0]
}
```

Interpretation

The API exposes a synchronization plan; it does not directly control a playout device or transport. A future integrator would decide how, where, and whether to apply these outputs in a real system.

StreamSync MVP v0.1

6. Metrics / errors / /health

Compact operational reference: what the system measures, how errors surface, and how the service reports health.

Core metrics

Metric	Unit	Meaning
Coherence r	0 to 1	Collective phase organization. Higher is more coherent.
Mean phase	radians	Circular mean of the current oscillator population.
Phase error	radians	Shortest signed angular difference from each stream to the population mean.
RMS phase error	radians	Population spread around the mean phase.
Lock state	categorical	Simple unlocked / converging / locked state from coherence thresholds.
Coupling K	dimensionless	Adaptive collective-coupling strength constrained by controller bounds.
Correction magnitude	radians	Derived coupled-vs-free-running phase correction; benchmark reports RMS and p95.

Operational errors

- **404 — unknown session_id:** returned when /observe or /plan references no active in-memory session.
- **422 — validation error:** wrong vector length, non-finite values, invalid availability outside [0,1], invalid timestep, or forbidden extra request fields.

GET /health

```
{
  "status": "ok",
  "service": "streamsync",
  "version": "0.1.0"
}
```

Current verification

9 / 9 tests pass in the current clean v0.1 package, covering the Kuramoto core, bounded controller, conventional baseline, API round-trip, validation errors, unknown sessions, health endpoint, and reference-dropout representation at the API boundary.

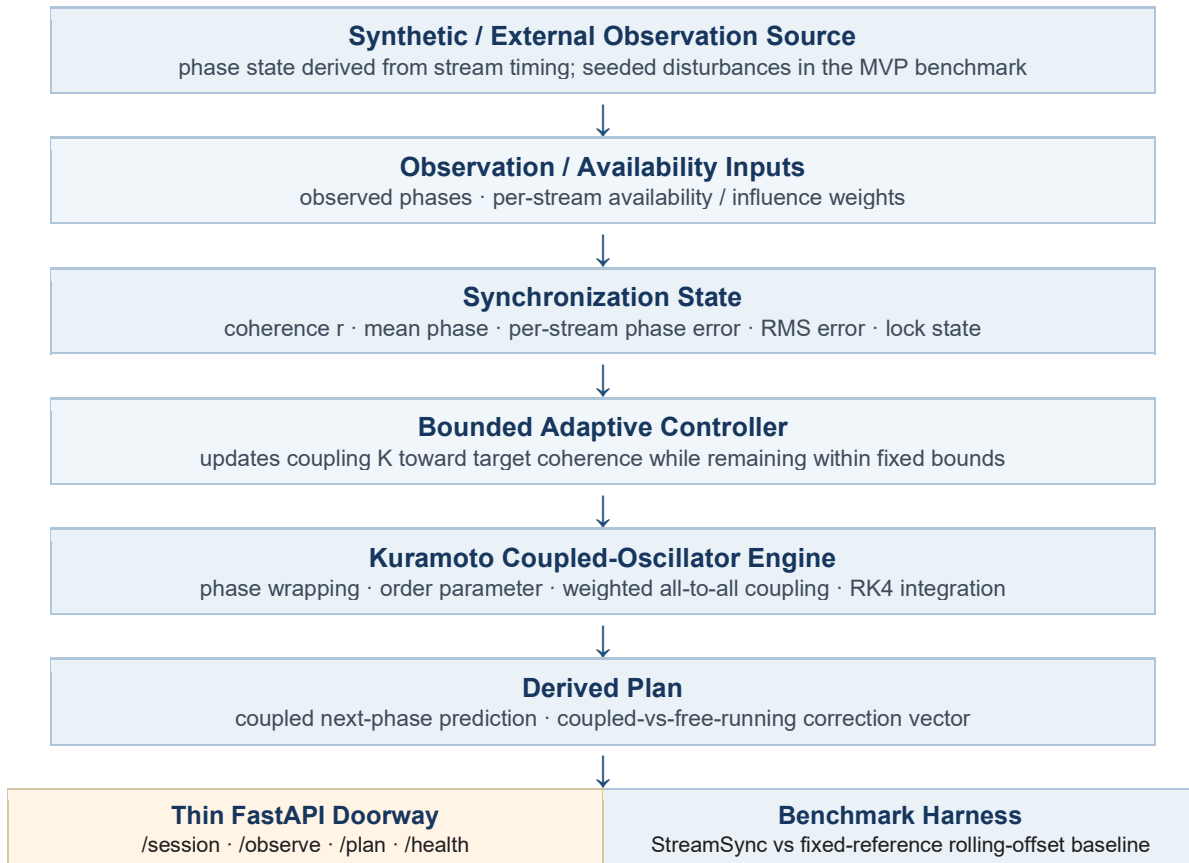
Scope note

No database, authentication system, distributed state, hardware clock discipline, transport layer, RF processing, beamforming, source separation, satellite link, multimodal-fusion model, or production observability stack is included in v0.1.

StreamSync MVP v0.1

7. Architecture diagram

The synchronization engine is independent. FastAPI is a thin doorway, and the benchmark harness remains a separate evidence path.



Design principles

- **Kuramoto first**: collective phase organization is the core mechanism.
- **Evidence before interface**: benchmarks run without FastAPI.
- **Bounded adaptation**: controller K is constrained; missing observations can be down-weighted.
- **Thin boundary**: the API validates JSON, stores ephemeral state, calls the engine, and returns inspectable state/plan data.

IMPORTANT SEPARATION

The current REST API accepts phase observations and availability weights. Timing/jitter/loss sensing is an upstream concern that can be domain-specific without changing the mathematical core.

StreamSync MVP v0.1

8. Roadmap / next steps

Move from synthetic evidence to limited real-stream validation before considering domain-specific integration or optimization.

CURRENT STATUS	GUIDING PRINCIPLE
Engine, controller/state logic, deterministic simulator, baseline, two benchmark families, thin FastAPI wrapper, documentation, and 9-test integration suite are in place.	Future-capable, not future-proof: keep the core modular enough to evolve without pretending to know the exact production system that may eventually use it.

Near-term closeout

Step	Artifact	Purpose
1	Freeze v0.1 package	Keep code, benchmark outputs, README, architecture, and API contract aligned. No new claims beyond current evidence.
2	Private GitHub repository	Publish the clean source tree, tests, reproducible benchmark scripts, and documentation as the technical engine room.
3	StreamSync.com	Use a small public gallery for overview, architecture, API, benchmark, paper, roadmap, and contact—no production or market-performance claims.

Validation path after v0.1

- **Limited real-stream test:** map a small real or recorded stream set into phase/availability observations and check whether the synthetic behaviors carry over.
- **Domain-specific integration only with a real use case:** RF/satellite, multimodal sensor fusion, or audio-array work would require its own sensing front end, metrics, and validation.
- **Optimize only after the use case is known:** lower-level languages, acceleration, deployment topology, persistence, and production networking should respond to actual integration requirements.
- **Preserve the independent engine:** future wrappers or transports should not become the synchronization experiment itself.

Potential future domains — not v0.1 claims

Domain	Possible role of a StreamSync-derived layer	Additional work required
Multimodal AI / sensors	Temporal-coherence layer before downstream fusion.	Domain timestamps, modality-specific confidence, real-data validation.
Satellite / RF	Collective receiver/observation synchronization when references drift, degrade, or disappear.	RF/DSP front end, clock/propagation model, field benchmarks.
Audio arrays	Phase/timing coherence before beamforming or source-separation stages.	Acoustic front end, geometry, beamforming/source-separation algorithms.

Closing statement

StreamSync v0.1 is intentionally modest: a sound architectural stone, not a finished cathedral. Its value is in a working coupled-oscillator core, reproducible evidence, a small inspectable API, and enough modularity to let a future integrator decide what to build around it.