Suivi cyclistes temps réel — Architectures détaillées

Auteur : Dr Hamid MADANI drmdh@msn.com **Date** : 2026-06-05 **Statut** : PROPOSITIONS à discuter — 5 architectures implémentées (variantes A, B, C, D, B+C) **Contexte** : composition `@mostajs/{config,data-plug,orm,net,geo}` + SFU/MCU `studio.amia.fr`

> Document de référence détaillant **chaque** architecture proposée pour le suivi de cyclistes en direct (positions + geofences + vidéo). Chaque variante est implémentée sous `variants/`. Les briques géo reposent toujours sur `@mostajs/geo` ; la persistance, le transport et le temps réel sont **délégués** (DEVRULES §10).

---

0. Briques communes (toutes variantes)

Préoccupation	Module / service	Rôle
Configuration	`@mostajs/config`	env + profils `MOSTA_ENV`
Persistance	`@mostajs/data-plug` → `@mostajs/orm`	`getDialect()`, schémas, DB interchangeable
Transport temps réel	`@mostajs/net`	REST/SSE/WS (fan-out positions)
Géo	`@mostajs/geo`	`distance`, `bearing`, `nearest`, `isWithin`, `geohashEncode`
Vidéo régie	**SFU** `studio.amia.fr/sfu`	faible latence (publisher/viewer)
Vidéo mixée	**MCU** `studio.amia.fr/mcu`	composition 1 flux
Code partagé	`common/course.mjs`, `schema.mjs`	parcours, checkpoints, `computeGeo()`, simulateur

Modèle de données (`schema.mjs`) : `Cyclist`, `Position`, `Checkpoint` (geofence), `Crossing`.

A — Device-direct · `variants/A-device-direct/`

Idée : chaque cycliste publie tout, directement.

```
cycliste : publisher.html —WebRTC→ SFU → viewer (dashboard)
cycliste : GPS → @mostajs/net (SSE) → carte
tracker.mjs → data-plug → orm        geo: nearest / isWithin / distance
```

- **Pour qui** : prototypes, petites courses, démo (**≤ ~20 flux**).
- ➕ : minimal, zéro composant intermédiaire, le plus rapide à monter.
- ➖ : le SFU encaisse *N publishers × M viewers* ; batterie/4G du coureur sollicitée ; pas de découplage producteur/consommateur.
- **Implémenté** : `tracker.mjs` (config+data-plug+geo, simulateur), `server.mjs` (net), `web/` (carte + viewer SFU).

B — Moto-relais · `variants/B-moto-relais/`

Idée : les motos suiveuses captent vidéo **et** agrègent les capteurs GPS des coureurs proches.

```
capteurs coureurs (BLE/ANT+) → passerelle moto (gateway.mjs) —4G→ data-plug → orm
caméra moto —publisher.html→ SFU → viewer
                              geo: distance() pour n'accepter que les coureurs du rayon
```

- **Pour qui** : vraie course sur route, captation TV.
- ➕ : **une** liaison 4G fiable par groupe (pas par coureur), qualité vidéo pro, peu de hardware côté cyclistes ; assignation coureur→moto par `nearest()` / `distance()`.
- ➖ : logistique (motos, appairage capteurs), recouvrement entre motos à arbitrer.
- **Implémenté** : `gateway.mjs` (une instance par moto, `MOTO_ID`).

C — Fan-out massif · `variants/C-fanout-massif/`

Idée : séparer *régie temps réel* et *diffusion publique*, et découpler par un broker.

```
devices —MQTT(cyclists/+/pos)→ ingest-mqtt → geo + data-plug → orm
                                          └→ @mostajs/net (SSE/WS) → N spectateurs (carte)
caméras → SFU (régie)
        └→ MCU (mixage) → egress HLS/CDN → N spectateurs (vidéo)
```

- **Pour qui** : événement public, **milliers** de spectateurs.
- ➕ : MQTT découple/QoS/scale ; SSE/WS = *un flux serveur → N clients* ; HLS/CDN scale la vidéo ; SFU conservé pour la faible latence (régie).
- ➖ : plus de composants (broker, egress, CDN) ; +latence côté HLS public (~2–15 s).
- **Implémenté** : `ingest-mqtt.mjs` (MQTT→geo→DB, `PUBLISH=1` sème un peloton), `egress-notes.md` (SFU/MCU→HLS via ffmpeg), `web/` (spectateur HLS + carte).

D — Edge / offline-first · `variants/D-edge-offline/`

Idée : le cœur pur `@mostajs/geo` tourne **sur l'appareil** ; les données survivent aux coupures.

```
APPAREIL : GPS ─► geo (geofence/nearest/geohash) ─► événements + positions échantillonnées
                                                    └► file locale ─(reconnexion)► data-plug (net)
```

- **Pour qui** : montagne/tunnels, économie batterie/données.
- ➕ : fonctionne **hors-ligne** (geo sans réseau), n'émet que l'essentiel, **store-and-forward** avec retry.
- ➖ : logique embarquée à maintenir ; positions serveur moins denses.
- **Implémenté** : `edge-agent.mjs` (calcul géo local + file + flush/retry, coupures simulées).

B + C — Régie + Public *(2e alternative, non testée)* · `variants/BC-regie-public/`

Idée : captation pro (**B**) + diffusion massive (**C**), géo serveur puis **edge (D)** sur zones blanches.

```
motos: capteurs ─MQTT─► ingest ─► geo+data-plug ─► orm/PostGIS
motos: caméra ─► SFU (régie <0,5 s) ──────────────► régie/commentateurs
                └► MCU (mixage) ─► HLS/CDN ──────► grand public
net (SSE/WS) ◄── positions ──────────────────────► cartes spectateurs
```

- **Pour qui** : couverture événementielle complète.
- ➕ : qualité (B) + échelle (C) + résilience (D) ; une seule brique géo partout.
- ➖ : la plus riche à opérer (orchestration multi-services).
- **Implémenté** : `docker-compose.yml` (mqtt + net + ingest + PostGIS + egress) + README.

Choix orthogonaux (combinables avec A–D)

Axe	Options	Quand préférer
Calcul géo	serveur (net→geo) · **edge** (geo sur device)	edge si bande passante/latence/offline critiques (D)
Topologie vidéo	**SFU** (faible latence) · **MCU** (1 flux mixé) · **HLS egress** (audience massive)	SFU+HLS pour « régie + public » (C, B+C)
Transport positions	SSE (simple) · WS (bidir/commandes) · **MQTT** (IoT, QoS, offline)	MQTT dès qu'il y a des capteurs terrain (C, B+C)
Persistance	SQLite (dév) · PostgreSQL/**PostGIS** · Mongo `2dsphere`	PostGIS pour requêtes spatiales à l'échelle
Analytics	events geo → `@mostajs/notifications` (chute = arrêt anormal, écart peloton) + overlays télémétrie	valeur ajoutée organisateur

Tableau comparatif

Critère	A	B	C	D	B+C
Mise en œuvre	★ très simple	★★	★★★	★★	★★★★
Échelle spectateurs	faible	moyenne	**très haute**	moyenne	**très haute**
Qualité vidéo	variable	**pro**	pro	variable	**pro**
Résilience réseau	faible	moyenne	moyenne	**forte**	forte
Coût/complexité infra	minimal	moyen	élevé	faible	élevé
Latence vidéo	SFU (~0,3 s)	SFU	SFU + **HLS** public	SFU	SFU + HLS

Guide de décision (rapide)

- **Démo / PoC → A**.
- **Course sur route, captation sérieuse → B**.
- **Grand public, large audience → C**.
- **Zones sans réseau fiable → D** (ou option edge sur A/B/C).
- **Événement complet (qualité + échelle + résilience) → B + C** (+ edge D au besoin).

> Invariant : quelle que soit l'architecture, `@mostajs/geo` **reste la seule brique géo** (cœur pur, serveur **ou** device) ; persistance, transport, temps réel et vidéo sont **délégués** aux modules/services dédiés (DEVRULES §10).

Propositions à discuter — Dr Hamid MADANI drmdh@msn.com — 2026-06-05.