Security Audit Report

Optimum Gateway

AUDIT COMPLETED	31 July 2026
REPORT PUBLISHED	18 August 2026

Table of Contents

Report Details

Authored by:	ProbeLab
Publication:	ProbeLab Blog
Protocol audited:	Optimum Gateway - `optimum-gateway`
Repository:	A clean new version of the code was generated at `github.com/getoptimum/optimum-gateway`
Code references:	All suggestions and comments were addressed at `v1.0.2` tag
Audit periods:	March–April 2026 (initial engagement); July 2026 (follow-up review)
Report date:	2026-07-31
Report status:	Final

About ProbeLab. ProbeLab is an independent research and engineering team specialising in the measurement and optimisation of peer-to-peer networking protocols. Our work spans protocol design, performance benchmarking, and security reviews across the Ethereum ecosystem. We maintain active tooling across the Ethereum P2P stack, including Hermes, our instrumented Ethereum node for CL network analysis, and contribute to P2P research on libp2p, including GossipSub.

This audit was conducted as part of our ongoing program of applied security work on infrastructure running in, or alongside the Ethereum consensus layer. This report reflects our findings after the review of the Optimum Gateway and does not constitute a guarantee of the absence of vulnerabilities in code written after that point, nor does it cover the deployment environment or operator-specific configuration.

Executive Summary

This report covers an audit for the Optimum Gateway, split into two phases. All amendments and fixes were addressed by the Optimum team at the `v1.0.2` tag (partner release `getoptimum/gateway`:`v1.0.2`).

Phase 1 (March–April 2026, initial engagement): ProbeLab identified 22 numbered findings, plus 3 aggregator code-review items. The single high-severity finding (Phase 1 F-03, missing mump2p → CL publish step) was patched during the engagement; remaining items were tracked to resolution in `v1.0.2`.

Phase 2 (July 2026, follow-up): ProbeLab re-reviewed the codebase after an initial review of the first raised concerns. Seventeen new findings (F-01–F-17) emerged from Phase 2, none of which critical; all were addressed in `v1.0.2` (F-07 partially resolved; telemetry deferral documented). The follow-up Phase 2 emphasised CL libp2p peering, Ethereum `req/resp` handshake handling, SSZ message processing, attestation subnet alignment, PeerDAS metadata, and observability on the CL ↔ mump2p forwarding path.

The audit confirmed that the gateway runs in parallel to the CL client's native libp2p stack, does not modify validator signing logic or attestation duties, and can be stopped at any time without affecting validator operation.

Engagement	Findings	Crit.	High	Medium	Low	Info	Status
March–April 2026	22 (+3 aggregator)	0	1 resolved (prior F-03)	13 resolved	5 resolved	4 resolved	**All resolved in v1.0.2**
July 2026 follow-up	17 (F-01–F-17)	0	1 resolved (F-01)	9: 8 resolved; 1 partial (F-07)	1 resolved	6 resolved	**All resolved in v1.0.2**

Field	Value
Auditor	ProbeLab
Remediation baseline	`v1.0.2` tag

SECTION 01

What the Software Does

The Optimum Gateway is responsible for receiving, validating, aggregating, and forwarding Ethereum messages (blocks, attestations) between two sides: the Optimum P2P network (mump2p) and the Ethereum CL host. It operates as a sidecar process alongside standard validator infrastructure.

- **Deployment model:** Sidecar daemon alongside validator infrastructure, typically paired with one CL client; does not modify validator signing logic, beacon state, or attestation duties.
- **Interaction surface:** Local CL client via `libp2p/GossipSub`; Optimum bootstrap API for validator-set sync, peer discovery, and trace submission; beacon API for fork-digest verification.
- **Compatible CL clients:** Lighthouse, Prysm, Nimbus, Teku, Lodestar, Grandine; validated by the Optimum team on Hoodi with the gateway at tag `v1.0.2`.
- **Peering:** The CL client must be configured with the gateway libp2p multiaddr (boot-node / trusted peer / equivalent). That is the required step for the CL to reach the gateway. Gateway-side `direct_cl_peers` is not required for the initial connection: when the CL dials in, the gateway registers the peer dynamically in the GossipSub direct-peer set at runtime. `direct_cl_peers` remains recommended: it enables gateway-initiated re-dial after a gateway restart (Lighthouse and Nimbus do not reliably re-dial on their own), and when non-empty it enforces a connect-time peer-ID allowlist (see F-05).
- **Operator profile:** Professional node operators running Ethereum validators.

SECTION 02

Audit Scope

2.1 In scope

Component	Description	Location
CL Host Configuration	libp2p host setup, GossipSub config, connection management, direct CL peer allowlist	`pkg/service/gossipsub-gateway/setup_libp2p_host.go`
Ethereum Handshake	Status/metadata/ping/goodbye req/resp, fork-digest validation, topic subscription	`pkg/service/gossipsub-gateway/subscribe_nodes.go`
Message Forwarding	CL ↔ mump2p routing, deduplication, block arrival telemetry	`pkg/service/gossipsub-gateway/messages_proxy.go`
SSZ / Consensus	Typed SSZ decode (FastSSZ), metadata and status messages	`pkg/protocol/consensus/`, `pkg/protocol/fastssz_codegen/`
Bootstrap / Telemetry	Block latency traces, fork-digest manager, Prometheus metrics	`pkg/service/bootstrapper/`, `pkg/service/telemetry/`
Attestation Aggregator	Batching and compression logic (code review and unit tests)	`pkg/service/aggregator/`
End-to-end stability	Hermes ↔ Gateway message delivery on Hoodi testnet	Validation environment

Networks tested: Hoodi testnet

Topics tested: `beacon_block`, beacon_attestation_* (all 64 subnets)

2.2 Out of scope

- Off-chain infrastructure and hosting environment
- The mump2p core network protocol
- CL client internals
- Operator-specific firewall, Docker networking, and `direct_cl_peers` configuration

The review combined manual code review across all in-scope areas, static analysis, and end-to-end integration testing on Hoodi testnet using Hermes as a passive CL receiver.

2.3 Methodology

The following setup was used to test the interaction between the Optimum Gateway and Hermes (representing the CL side):

- Hermes was deployed locally on Hoodi with peer discovery disabled, using the Gateway's multiaddress as a direct connection, ensuring the Gateway was the only connected peer.
- The Gateway was deployed locally on the same machine, with Hermes's multiaddress configured as a direct connection.
- The experiment began once both hosts successfully established a mutual connection.
- With no other peers connected, all messages received by Hermes originated exclusively from the Gateway.

```
Hermes <--direct.connection--> Optimum Gateway <--connections--> mump2p Network
```

SECTION 03

Architecture Overview

3.1 System diagram

Documentation: Optimum Gateway (`v1.0.2`)

```
CL_Client --> [Optimum Gateway] <--> [mump2p network]
                    |
            [Bootstrap API]
  (validator sync, fork-digest,
   peer discovery, traces)
```

The gateway is configured for a single network (e.g. mainnet or Hoodi) and is designed to operate with one CL client in the typical deployment.

3.2 Interaction with existing infrastructure

The gateway runs in parallel to the CL client's native libp2p stack. It does not replace GossipSub, does not modify the CL client, and does not interact with validator signing, attestation construction, or block proposal logic.

- **Parallel operation:** The CL client's existing `libp2p/GossipSub` mesh keeps operating independently; the Gateway is simply notified of new messages on that mesh.
- **Fallback behavior:** If the gateway stops, or loses connectivity to mump2p, the CL client continues on its existing libp2p mesh. Validator duties are unaffected (see 3.3).
- **No validator access:** No modification of, or access to, validator key management, beacon state, attestation construction, or block proposal logic. It only operates at the plain p2p networking level.

3.3 Failure mode and reversibility

Stopping the gateway is process termination. The CL client keeps its existing libp2p mesh. Validator duties, attestations, and proposals are unaffected. On stop, the gateway closes CL gossipsub subscriptions, shuts down the CL libp2p host and the mump2p node, and writes nothing to CL storage, validator keystores, or the beacon DB.

SECTION 04

Findings

4.1 First Audit (March–April 2026)

Findings below are from the initial ProbeLab engagement (report dated 2026-05-27 and shared privately with the Optimum team). Wording is preserved from that report. Code paths are updated to the public `pkg/` tree at tag `v1.0.2`. Some of the topics that came up from this first report had a followup comment in the second revisit. Please refer to the table below to find the references between topics in 4.1 and 4.2.

Note: as this report is the merge of both audits, it doesn't include any commit hashes, issue numbers, or pull-request links from the first, Phase 1 audit - these have been worked on in a private repository.

Cross-reference: findings covered in both audits

Prior ID	July 2026 ID	Topic
F-17	F-01	Attestation subnet advertisement vs subscription
F-10, F-12	F-04	Fork-digest validation and topic construction
F-05, F-19	F-07	Peer-level / delivery telemetry
F-15	F-09	Split message handlers
F-20, F-21, F-22	F-08, F-10	SSZ decode and forwarding hot path

4.1.1 Key findings

Prior F-03 · Missing publish step in mump2p → CL path

HIGH RESOLVED

Location: `pkg/service/gossipsub-gateway/messages_proxy.go` (`processMumP2PBeaconBlock`)

Description: The inbound mump2p handler performed message validation and decode steps correctly but did not call `topic.Publish()` to forward the message to the CL-side GossipSub topic. Measurement-mode configuration (`propagation_enabled=false` via Optimum dynamic config) was active in the test environment and initially masked the issue. When propagation was forced on at code level during investigation, blocks still did not reach the CL host, confirming the missing publish step as a distinct and independent issue.

Impact: No messages were forwarded from mump2p to the CL host. End-to-end delivery was non-functional until this was patched during the engagement.

Resolution: The handler was updated to publish to the CL topic after validation (`publishToCLTopic`). End-to-end delivery was confirmed on Hoodi testnet: Hermes received beacon block messages from the gateway following the fix. Dynamic direct peering was also confirmed functional after removing a static direct-peer entry from config.

Measurement / calibration mode: Optimum can remotely disable CL forwarding via bootstrap dynamic config (`propagation_enabled=false`). When disabled, the mump2p → CL publish path is skipped. This is used on Hoodi to measure mump2p vs libp2p baselines independently during calibration. This flag masked F-03 during initial testing (forwarding appeared off for measurement reasons before the missing publish bug was isolated). Not active in normal mainnet partner operation.

Prior F-01 · Seen-message TTL over-retention

MEDIUM RESOLVED

Location: `pkg/service/gossipsub-gateway/messages_proxy.go`

Description: The seen-message cache retained entries longer than needed for deduplication on the forwarding path.

Impact: Unnecessary memory retention under sustained traffic.

Resolution: TTL map window tightened for the deduplication cache used on the CL ↔ mump2p path.

Prior F-02 · Redundant connection manager

MEDIUM RESOLVED

Location: `pkg/service/gossipsub-gateway/setup_libp2p_host.go`

Description: A redundant libp2p connection manager was configured alongside GossipSub's own connection handling.

Impact: Operational complexity; potential for conflicting connection limits.

Resolution: Redundant manager removed; connection policy consolidated.

Prior F-04 · Relay transport enabled unnecessarily

LOW RESOLVED

Location: `pkg/service/gossipsub-gateway/setup_libp2p_host.go`

Description: Circuit relay transport was enabled on the CL-facing libp2p host without an operational requirement.

Impact: Expanded attack surface and unnecessary libp2p capability advertisement.

Resolution: Relay transport disabled for the CL host configuration.

Prior F-08 · Incomplete graceful shutdown: mump2p host not closed

MEDIUM RESOLVED

Location: `pkg/service/gossipsub-gateway/service.go`

Description: Graceful shutdown closed the CL-side libp2p host but did not consistently tear down the mump2p-side host.

Impact: Lingering connections and goroutines on restart; harder clean restarts in orchestrated deployments.

Resolution: Service stop path now closes both CL and mump2p hosts. See also §3.3 Failure mode and reversibility.

Prior F-09 · Unnecessary mutex on libp2p host

LOW RESOLVED

Location: `pkg/service/gossipsub-gateway/setup_libp2p_host.go`

Description: A mutex guarded libp2p host access where the host lifecycle is concurrently safe to use.

Impact: Minor complexity; no direct security impact.

Resolution: Mutex removed.

Prior F-11 · Misleading helper name: `filterEthTopics`

LOW RESOLVED

Location: `pkg/service/gossipsub-gateway/subscribe_nodes.go`

Description: Topic-building helper name did not reflect that it also constructs full Ethereum CL topic paths from descriptors.

Resolution: Renamed to `filterAndBuildEthTopics`.

Prior F-13 · Unnecessary stream handler registered for GossipSub topics

MEDIUM RESOLVED

Location: `pkg/service/gossipsub-gateway/subscribe_nodes.go`

Description: A generic stream handler was registered on GossipSub topic streams where GossipSub already handles message delivery.

Impact: Dead code path; audit noise.

Resolution: Handler removed.

Prior F-14 · Unnecessary mutex in `subscribeToTopic`

LOW RESOLVED

Location: `pkg/service/gossipsub-gateway/subscribe_nodes.go`

Description: Subscription setup used a mutex on a threadsafe operation.

Resolution: Mutex removed.

Prior F-16 · Unnecessary RPC handlers registered

LOW RESOLVED

Location: `pkg/service/gossipsub-gateway/subscribe_nodes.go`

Description: Some RPC handlers remained registered beyond the Ethereum CL handshake `req/resp` set required for interoperability.

Resolution: Paired to supported `status/metadata/ping/goodbye` handlers (see July F-03).

Prior F-06 · `DenyPrivateGater` ambiguous production role

INFO RESOLVED

Location: `pkg/service/gossipsub-gateway/setup_libp2p_host.go`

Description: Role of the private-IP connection gater in production deployments was unclear in documentation.

Resolution: Documented in `SECURITY.md` trust model.

Prior F-18 · `IsBeaconTopic` naming ambiguity

INFO RESOLVED

Location: `pkg/protocol/topics/`

Description: Topic classification helpers conflated beacon-block and attestation topic kinds.

Resolution: Topic metadata moved to `pkg/protocol/topics` with explicit kind resolution (`TopicMetaFor`).

Aggregator · Attestation publish timing

MEDIUM RESOLVED

Location: `pkg/service/aggregator/aggregator.go`

Description: Per-gateway jitter on attestation publish timing was unclear, and the slot gate did not account for early block arrivals.

Resolution: Per-gateway jitter removed. Block-aware gate in `shouldHoldForSlotGate`: if a block is seen in the current slot, collected attestations are released 2s after block receipt; otherwise they are released at 4s from slot start (configurable via `AttestationPublishAfterMs`). Block arrival time is sourced from `LastBlockReceivedMs()` on the gossipsub service.

Aggregator · Compact topic identifier

MEDIUM RESOLVED

Location: `pkg/service/aggregator/`

Description: Aggregated attestation containers on mump2p carried full Ethereum CL topic path strings, increasing payload size on a high-volume path.

Impact: Unnecessary bandwidth and parsing overhead on mump2p attestations.

Resolution: Containers now use a compact topic identifier; the full topic path is resolved at `pack/unpack` time via the topic metadata registry.

Aggregator · Topic type at handler registration

MEDIUM RESOLVED

Location: `pkg/service/gossipsub-gateway/messages_proxy.go`, `pkg/protocol/topics/`

Description: Message type was inferred repeatedly on each message at the forwarding path instead of once when handlers were wired.

Impact: Maintainability concern and minor hot-path overhead on the critical forwarding path.

Resolution: Topic metadata resolved once via `TopicMetaFor` at handler registration; dedicated handlers per message kind (see July F-09).

4.1.2 Informational notes

- **Per-peer `BeaconStatus` storage (prior F-07):** Resolved via shared network beacon status: handshake state is no longer stored per peer ID.
- **Validator sync allocation:** Background sync rebuilds a full validator map on each refresh. Not a performance concern at current scale; superseded by per-gateway validator subset design from API-key auth.
- **Structured logging:** Current logging would benefit from a structured library supporting warning-level output (e.g. `slog`). There is no security implication; this is a recommendation only.

4.2 Second Audit (July 2026)

This report incorporates all findings from both audit engagements. The table below is the July 2026 follow-up at `v1.0.2`.

ID	Title	Severity	Status
F-01	Attestation subnet advertisement vs subscription mismatch	**High**	**Resolved**
F-02	Imprecise metadata exchange (CGC vs data-column subscriptions)	**Medium**	**Resolved**
F-03	Non-spec Status V3 handler / implicit protocol dispatch	**Medium**	**Resolved**
F-04	Fork-digest source of truth unclear	**Medium**	**Resolved**
F-05	Direct CL peer allowlist behavior undocumented	**Medium**	**Resolved**
F-06	Single-node vs multi-node CL pairing ambiguity	**Medium**	**Resolved**
F-07	Peer-level metrics gaps for `CL/mump2p` connections	**Medium**	**Resolved (partial)**
F-08	Hand-rolled SSZ field slicing on hot paths	**Medium**	**Resolved**
F-09	Shared message handler for blocks and attestations	**Medium**	**Resolved**
F-10	Message forwarding hot-path inefficiencies	**Medium**	**Resolved**
F-11	Aggregator behavior unverified during audit window	**Low**	**Resolved**
F-12	Service boundaries (bootstrapper, auth, routes)	**Informational**	**Resolved**
F-13	entities package mixed concerns	**Informational**	**Resolved**
F-14	Protocol package naming and forks layout	**Informational**	**Resolved**
F-15	Global mutable state review	**Informational**	**Resolved**
F-16	Naming and cross-feature cohesion	**Informational**	**Resolved**
F-17	Module layout for open-source contributors	**Informational**	**Resolved**

Severity definitions

- **High** - functional failure or significant risk under realistic operating conditions
- **Medium** - correctness, performance, or operational concern; limited direct exploit surface
- **Low** - best-practice deviation, minor risk
- **Informational** - no direct risk; clarity or improvement recommendation

Status definitions

- **Resolved** - fix merged or validated on Hoodi in `v1.0.2`
- **Resolved (partial)** - core fix shipped; follow-up recommendation documented
- **Open** - not yet addressed in code

4.2.1 Key findings

F-01 · Attestation subnet advertisement vs subscription mismatch

HIGH RESOLVED

Location: pkg/service/gossipsub-gateway/subscribe_nodes.go, pkg/utils/topics.go

Description: The gateway advertised all 64 attestation subnets in libp2p metadata (Attnets = `0xFFFFFFFFFFFFFFFF`) via `newMetadataMessage`, but gossipsub did not subscribe to all corresponding `/eth2/{fork_digest}/beacon_attestation_{0..63}/ssz_snappy` topics. CL clients that use advertised subnets for peering decisions could ban or deprioritise the gateway; validators relying on the gateway for attestation forwarding could miss messages.

Impact: Peering health degradation and potential attestation delivery gaps under realistic CL client behaviour.

Resolution: `UnpackTopics` expands the logical `beacon_attestation` descriptor into all 64 indexed subnet topics (`beacon_attestation_0` … `beacon_attestation_63`). `filterAndBuildEthTopics` builds the full CL-side topic list from those descriptors; `subscribeLocalNode` subscribes to each topic. Metadata Attnets bitfield now matches active subscriptions. Validated on Hoodi with Hermes as direct peer; unit coverage in `topics_test.go`.

F-02 · Imprecise metadata exchange

MEDIUM RESOLVED

Location: pkg/service/gossipsub-gateway/subscribe_nodes.go, pkg/protocol/consensus/rpc.go

Description: Metadata v3 (`metadata/3/ssz_snappy`) advertised `CustodyGroupCount`: 8 (Fulu `VALIDATOR_CUSTODY_REQUIREMENT`) without matching subscriptions to the corresponding `data_column_sidecar_{N}` gossip topics.

Impact: Lighthouse v8+ and other PeerDAS-aware clients could disconnect the gateway after metadata exchange.

Resolution: Dedicated metadata handlers for `v0/v1/v2/v3` with typed `MetaDataV0`, `MetaDataV1`, and `MetaDataV2` structs. CGC=8 is documented inline in `newMetadataMessage`. Advertised custody count aligns with data-column sidecar subscriptions on the CL gossipsub mesh.

F-03 · Ethereum handshake: spec compliance

MEDIUM RESOLVED

Location: pkg/service/gossipsub-gateway/subscribe_nodes.go, pkg/protocol/consensus/rpc.go

Description: Handshake logic previously included paths for non-spec protocol versions and relied on string inspection of protocol IDs to pick handler behaviour.

Impact: Interoperability risk with spec-compliant CL clients; harder-to-audit dispatch logic.

Resolution: Explicit libp2p stream handlers registered only for supported protocols: `status/1`, `status/2`, `metadata/1`, `metadata/2`, `metadata/3`, `ping/1`, `goodbye/1. newStatusMessage` returns typed Status or `StatusV2` based on the stream protocol ID. No Status V3 path. Distinct from the mump2p auth handshake in `handshake.go`.

F-04 · Fork-digest handling

MEDIUM RESOLVED

Location: pkg/service/gossipsub-gateway/subscribe_nodes.go, pkg/protocol/forks/service.go

Description: Fork-digest sourcing and validation flow was unclear: bootstrap API, handshake peer input, and topic construction were not documented as a single pipeline.

Impact: Risk of subscribing to wrong-fork topics if bootstrap data and peer handshake diverge.

Resolution: Bootstrap API documented as canonical fork-digest source (https://`/bootstrap.getoptimum.io/api/v1/fork-digest`?`chain_id`=<network>). Handshake path validates incoming fork digests via `CheckForkSupported` before accepting a CL peer. Genesis override removed; fork migration covered by periodic topic refresh in the gateway service.

F-05 · Direct CL peer allowlist

MEDIUM RESOLVED

Location: `pkg/service/gossipsub-gateway/setup_libp2p_host.go`, `SECURITY.md`

Description: When `direct_cl_peers` is non-empty, `onPeerConnected` disconnects any libp2p peer whose ID is not in the allowlist. This is intentional connect-time filtering, but was not documented for operators.

Impact: Unexpected disconnects if operators misconfigure peer IDs; security reviewers could misread as bug.

Resolution: Threat model in `SECURITY.md` documents the allowlist as a trust anchor. Configuration docs describe `direct_cl_peers` for CL auto-reconnect (especially clients that do not re-dial after a gateway restart) and note that operators must firewall port 33212 when the list is empty.

F-06 · CL pairing model

MEDIUM RESOLVED

Location: Gateway config and service wiring

Description: Code paths suggested multi-CL pairing while documentation and typical deployment assume one primary CL client per gateway instance.

Impact: Operator confusion; telemetry edge cases on multi-CL setups.

Resolution: Documented single primary CL as the supported deployment model. `direct_cl_peers` allow-list defines the intended CL set. `/api/v1/self_info` and `/health` expose per-side peer counts for the connected CL.

F-07 · Peer-level metrics

MEDIUM RESOLVED (PARTIAL)

Location: `pkg/service/telemetry/peers.go`, `pkg/service/telemetry/gossipsub.go`

Description: Connection and delivery metrics lacked consistent `peer_id` labels on all series, making it difficult to detect a silently disconnected CL peer from Prometheus alone.

Impact: Operational blind spot: the gateway could appear healthy on aggregate counters while the CL link is down.

Resolution: `cl_mesh_peer{topic, peer_id}` and `mump2p_mesh_peer{topic, peer_id}` gauges added. GossipSub RPC counters include `peer_id`. `/api/v1/self_info` returns libp2p and mump2p peer IDs and per-topic peer lists.

Partial deferral: A dedicated `direct_cl_peer_active{peer_id}` silence-detection gauge was deferred. At current scale (typically one CL peer plus a small mump2p mesh per gateway), per-topic mesh gauges and health endpoints provide sufficient visibility. Tracked in Open Items.

F-08 · SSZ decode: FastSSZ

MEDIUM RESOLVED

Location: `pkg/protocol/consensus/block.go`, `pkg/protocol/fastssz_codegen/`

Description: Beacon block slot and proposer fields on the hot path were extracted via hand-rolled byte offsets into SSZ payloads. This is fragile against layout changes and harder to audit than typed decode.

Impact: Correctness risk on malformed or future-format blocks; maintenance burden.

Resolution: Generated FastSSZ types vendored in `pkg/protocol/fastssz_codegen/` (regenerated via make fastssz-generate). `DecodeBeaconBlockHeader` performs a single Snappy decompress, validates the SSZ offset prefix, then `UnmarshalSSZTail` on `BeaconBlockHeader`, returning a typed `SignedBeaconBlockHeader`. `BodyRoot` is intentionally left unset: at that position the gossip payload holds a body offset, not a body root, so the returned header is not valid for hashing. Attestation `slot/index` parsing retains fixed-offset reads in `ParseAttestationSSZTopic` where a full decode is unnecessary. Malformed payloads increment SSZ error telemetry and are dropped. Unit tests cover valid blocks, short payloads, and bad offsets.

F-09 · Split message handlers

MEDIUM RESOLVED

Location: `pkg/service/gossipsub-gateway/messages_proxy.go`

Description: A single shared code path mixed beacon block and attestation forwarding logic, increasing audit surface and regression risk.

Impact: Maintainability and correctness concern on the critical forwarding path.

Resolution: Dedicated handlers: `processCLBeaconBlock`, `processCLAttestation`, `processMumP2PBeaconBlock`, `handleAggregatedMessages`. Topic kind resolved once via topics.`TopicMetaFor`.

F-10 · Message forwarding hot path

MEDIUM RESOLVED

Location: `pkg/service/gossipsub-gateway/messages_proxy.go`, `pkg/service/gossipsub-gateway/beacon_block_measures.go`

Description: The CL ↔ mump2p path performed redundant work: late deduplication, duplicate Snappy handling, and synchronous telemetry on the receive path.

Impact: Latency on block forwarding; unnecessary CPU on high-volume attestations.

Resolution: `isDuplicateMessage` (`xxhash` over payload, short TTL cache) runs before publish. Block arrival timestamps recorded in `processBeaconBlockArrival` before dedup so latency samples are captured even when mump2p delivered first. CL publish uses a single path with raw gossip bytes, with no second full SSZ decode on forward. Bootstrapper block traces enqueued asynchronously.

F-11 · Aggregator verification

LOW RESOLVED

Location: `pkg/service/aggregator/`

Description: Attestation aggregation and mump2p emit paths could not be fully exercised during the initial code-review window.

Impact: Unverified `packing/slot-gate` behaviour under load.

Resolution: Unit tests added for attestation packer and aggregator. Slot gate and block-aware publish timing have been documented. Kurtosis integration tests provide ongoing end-to-end validation.

4.2.2 Informational notes (F-12–F-17)

- **F-12 · Service boundaries:** Bootstrapper, auth token manager, and HTTP routes reorganised with documented responsibilities under `pkg/service/.`
- **F-13 · Entities package:** Ethereum topic helpers moved to `pkg/protocol/topics`; gateway message types remain in `pkg/entities.`
- **F-14 · Protocol naming:** `chain_state` and forks layout documented; genesis timing derived from selected chain config.
- **F-15 · Global state:** Fork manager and topic metadata cache scoped to service lifecycle; reduced hidden global mutation.
- **F-16 · Naming cohesion:** "Handshake" disambiguated between eth2 `req/resp` (`serveHandshake`) and mump2p auth (`handshake.go`).
- **F-17 · Module layout:** Public `pkg/` tree established for open-source contribution; `AGENTS.md` and docs describe module boundaries.

SECTION 05

Open Items

All security findings from this audit are addressed in the codebase validated for `v1.0.2`, with the partial deferral noted below.

Item	Status
`direct_cl_peer_active` per-peer silence metric (F-07 deferral)	Recommendation - future telemetry enhancement
Empirical check of the attestation forwarding logic and the correct aggregation through mump2p's network (F-11)	Recommendation - future test of the correct packing, unpacking, and forwarding of attestation messages to the CL node.

Appendices

A. Glossary

Term	Definition
mump2p	Optimum's RLNC-based p2p gossip protocol for accelerated Ethereum block and attestation propagation
Sidecar	Software component running alongside a validator client without modifying it
GossipSub	The libp2p `pub/sub` protocol used for Ethereum CL p2p message propagation
Fork-digest	A 4-byte identifier derived from a network's genesis validator root and current fork version; used to namespace GossipSub topics per network and fork
SSZ	Simple Serialize: Ethereum's binary serialization format for consensus-layer objects
Snappy	Compression algorithm applied to GossipSub messages in the Ethereum CL network
FastSSZ	Code-generation library for typed SSZ `marshal/unmarshal` used in `v1.0.2` beacon block decode
Hoodi	Ethereum testnet used for CL infrastructure testing
Bootstrap API	Optimum-operated API providing fork-digest lists, validator sets, and peer discovery data to gateway instances
Measurement mode	Gateway operating mode in which message forwarding to the CL host is disabled (`propagation_enabled=false`, set via config or remote dynamic config); used for baseline propagation benchmarking
CGC	Custody group count: PeerDAS metadata field indicating how many custody groups a node participates in