# Problem Statement 16
## Modular Internet Measurement Network over NATS

Reference: IETF **IPPM**: framework RFC 2330, OWAMP/TWAMP/STAMP **RFC** 4656/5357/8762, delay/RTT/loss **RFC** 7679/2681/7680, time sync RFC 5905 (NTP), data formats **RFC** 8259/8949 **(JSON/CBOR)**, security RFC 8446 (TLS 1.3), and module auth RFC 7519 (JWT).

**Objective**

Design and implement a **modular Internet Measurement Network** where an **Anchor** connects to a **NATS.io cluster** and dynamically loads independent **measurement modules**. Each module establishes its **own NATS connection** and operates mostly independently, enabling **hot-plug**, **fault isolation**, and **decoupled** measurement vs. management planes.

**Problem**

The previous monolithic design tied management and measurement together, limiting agility and resilience. Build a new system that:

1. Boots an **Anchor** that connects to NATS and **discovers/loads modules at runtime** (no restarts).
2. Runs multiple **modules** (e.g., latency-ping, dns-lookup, http-probe) that **individually connect to NATS**, subscribe/publish on their own subjects, and execute jobs independently.
3. Ensures **decoupling**: if a module fails or is replaced, the **Anchor and other modules continue** unaffected.
4. Provides **management APIs/UI** to dispatch jobs, watch results, and manage module lifecycle—without direct coupling to module processes.

**Required Components**

- **NATS Subjects / Contracts**
    - mmn.jobs.submit, mmn.jobs.result.*, mmn.modules.announce, mmn.modules.health.*
    - A small **schema** for job request/response (JSON or Protobuf) with versioning.

- **Anchor Service**
    - Module discovery (file system, registry, or announcement over mmn.modules.announce)
    - **Hot add/remove** of modules; **health/heartbeat** tracking; **backpressure** (queue depth) awareness.
- **Modules (≥2)**
    - Each module uses its **own NATS client**, subscribes to a job subject, publishes results, emits health.
    - Safe shutdown, retry policy, and **circuit-breaker** for target faults.
- **Management Plane**
    - Web or CLI to submit jobs, view results/live logs, and enable/disable modules.
    - **AJAX/WebSocket** updates for real-time result streaming.

## Non-Functional Requirements

- **Resilience:** A crashing module must not take down the Anchor or other modules.
- **Scalability:** Multiple instances of the same module type should **load-balance** via queue groups.
- **Observability:** Central logs/metrics (per-module latency, error rate, queue depth).
- **Security:** Basic auth for the UI/API; NATS creds/permissions scoped per subject; redact PII.
- **Upgrades:** Support **rolling replacement** of a module version without downtime.

## Deliverables

- **Anchor service** (containerized) + **two or more modules** (e.g., ping & DNS).
- **Contracts/spec** for NATS subjects and message schemas.
- **Management UI/CLI** to submit jobs and watch results in real time.
- **README** with runbook (docker-compose/k8s), example jobs, and fault-injection steps.

## Test Scenarios (must demonstrate)

- **Hot-plug:** Start the Anchor, then add a new module; it announces and starts serving jobs **without restart**.
- **Fault isolation:** Kill one module; other modules and the Anchor **keep working**; health reflects the failure.
- **Scale-out:** Run multiple instances of the same module; show **work sharing** and improved throughput.
- **Backpressure:** Saturate a module; Anchor throttles submissions or reroutes to healthy instances.

## Evaluation Criteria

- **Architecture & Decoupling (30%)** — Clean subject design, versioned schemas, failure boundaries.
- **Reliability & Scale (25%)** — Graceful degradation, queue handling, horizontal scaling.
- **Functionality (25%)** — Working Anchor, modules, and management plane with real measurements.
- **Observability & Security (10%)** — Metrics, logs, health, and guarded subjects/credentials.
- **Docs & UX (10%)** — Clear runbook, diagrams, and an intuitive job/results experience.