# Real-Time Bus Tracking & Historical Playback System

Tianming Yao
Computer Science – Georgia State University
Advisor: Dr. Yingshu Li

# Motivation

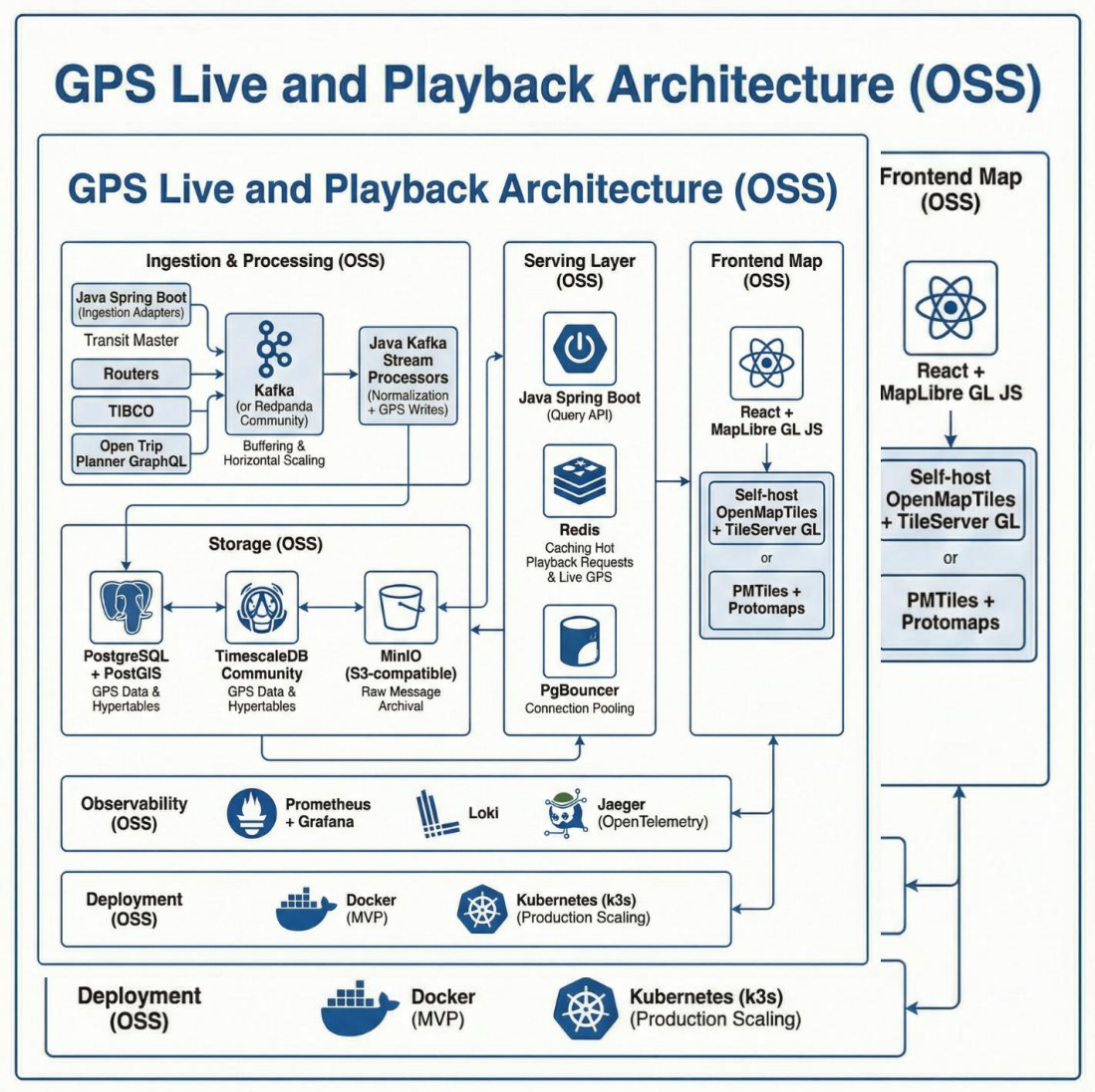
**Why This Project?**

- Transit agencies rely on real-time vehicle telemetry
- Commercial systems are:
  - Expensive
  - Vendor-locked
  - Limited in customization
- Historical playback often restricted

# Problem Statement

**The Need is** a system that is capable of:

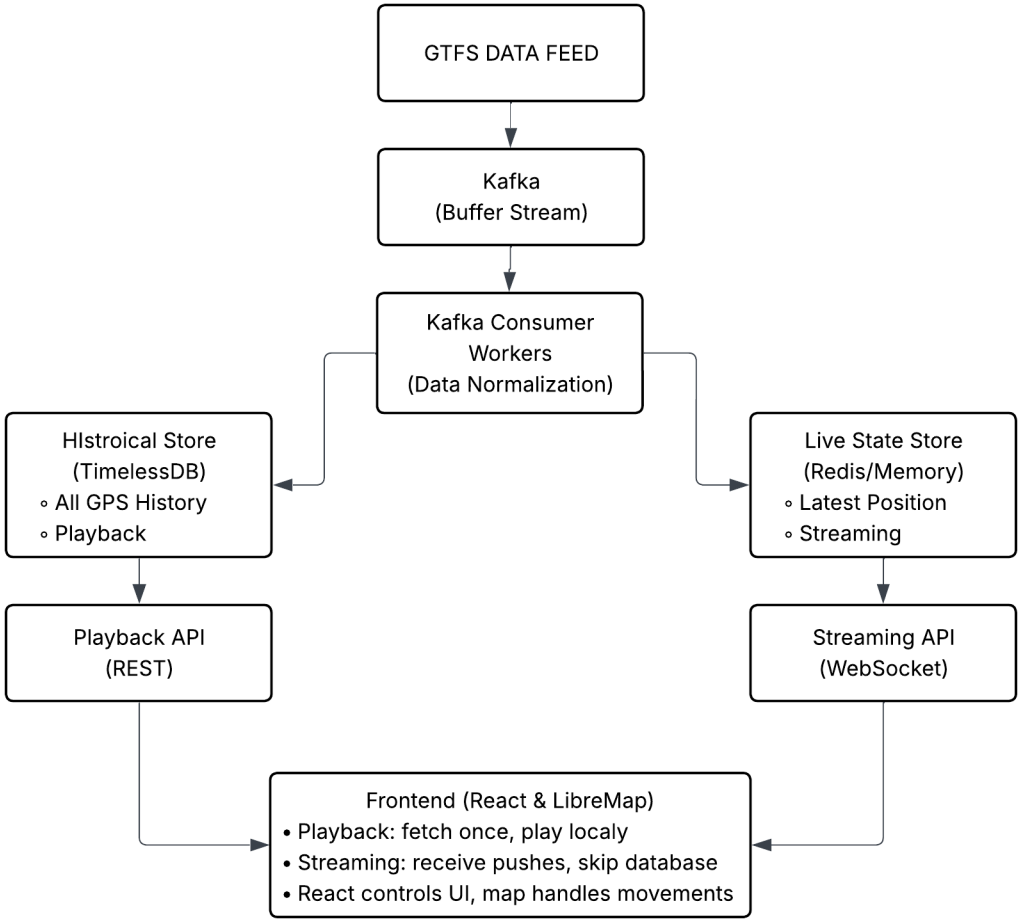
1. Ingesting GTFS-Realtime feeds
2. Storing enriched time-series telemetry
3. Supporting real-time visualization
4. Enabling historical playback
5. Supporting future AI-driven analytics

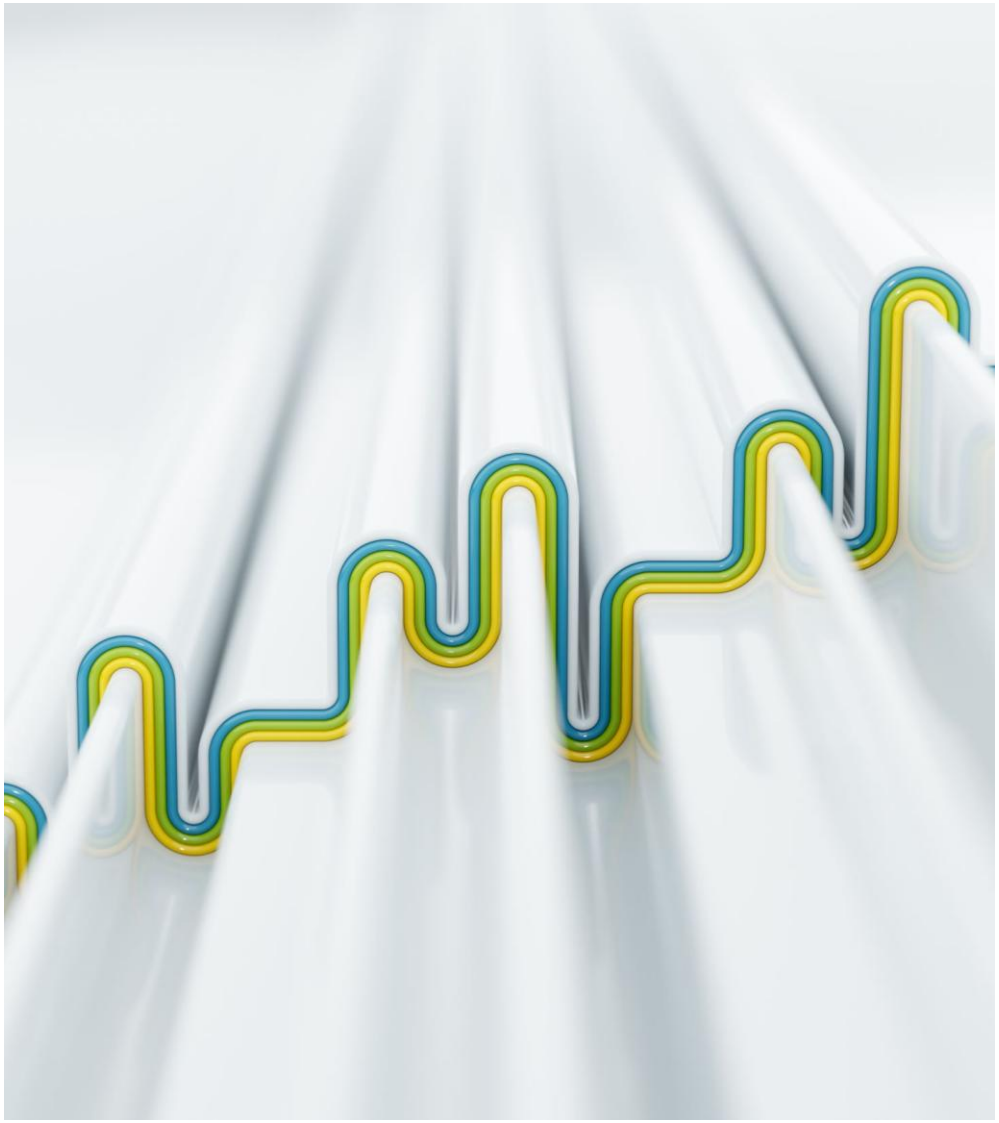



# System Architecture

**High-Level Architecture:**

GTFS-RT Feed
↓
Spring Boot Ingestion
↓
PostgreSQL + TimescaleDB
↓
Materialized View (Enrichment)
↓
REST + WebSocket API
↓
React + MapLibre Frontend


GTFS DATA FEED
Kafka
(Buffer Stream)
Kafka Consumer
Workers
(Data Normalization)
HIstroical Store
(TimelessDB)
◦ All GPS History
◦ Playback
Live State Store
(Redis/Memory)
◦ Latest Position
◦ Streaming
Playback API
(REST)
Streaming API
(WebSocket)
Frontend (React & LibreMap)
• Playback: fetch once, play localy
• Streaming: receive pushes, skip database
• React controls UI, map handles movements



# Database Design & Data Pipeline

**Data Pipeline**

- GTFS-RT → Kafka (stream ingestion)
- Kafka → Backend consumers
- Data normalization
- Stored in TimescaleDB
- Served via REST API

**Time-Series Optimization**

- PostgreSQL + TimescaleDB
- Hypertable on event_time
- Indexed by vehicle_id and timestamp
- Optimized range queries

# Enrichment Layer

**Materialized View**

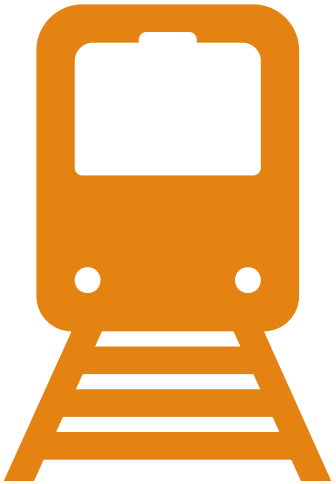
best_vehicle_positions_mv

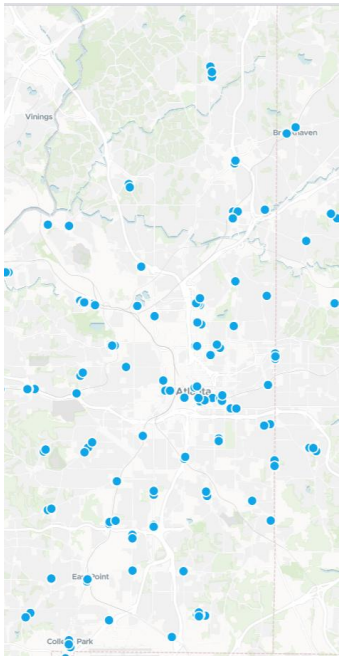
Joins:

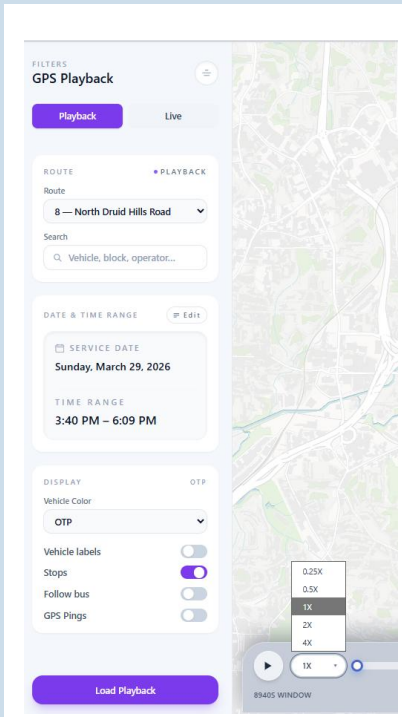
- vehicle_positions
- trips
- routes

Adds:

- route_short_name
- direction_id
- headsign



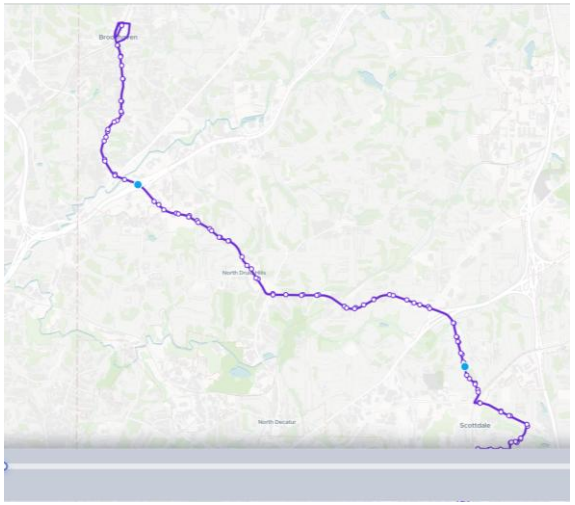


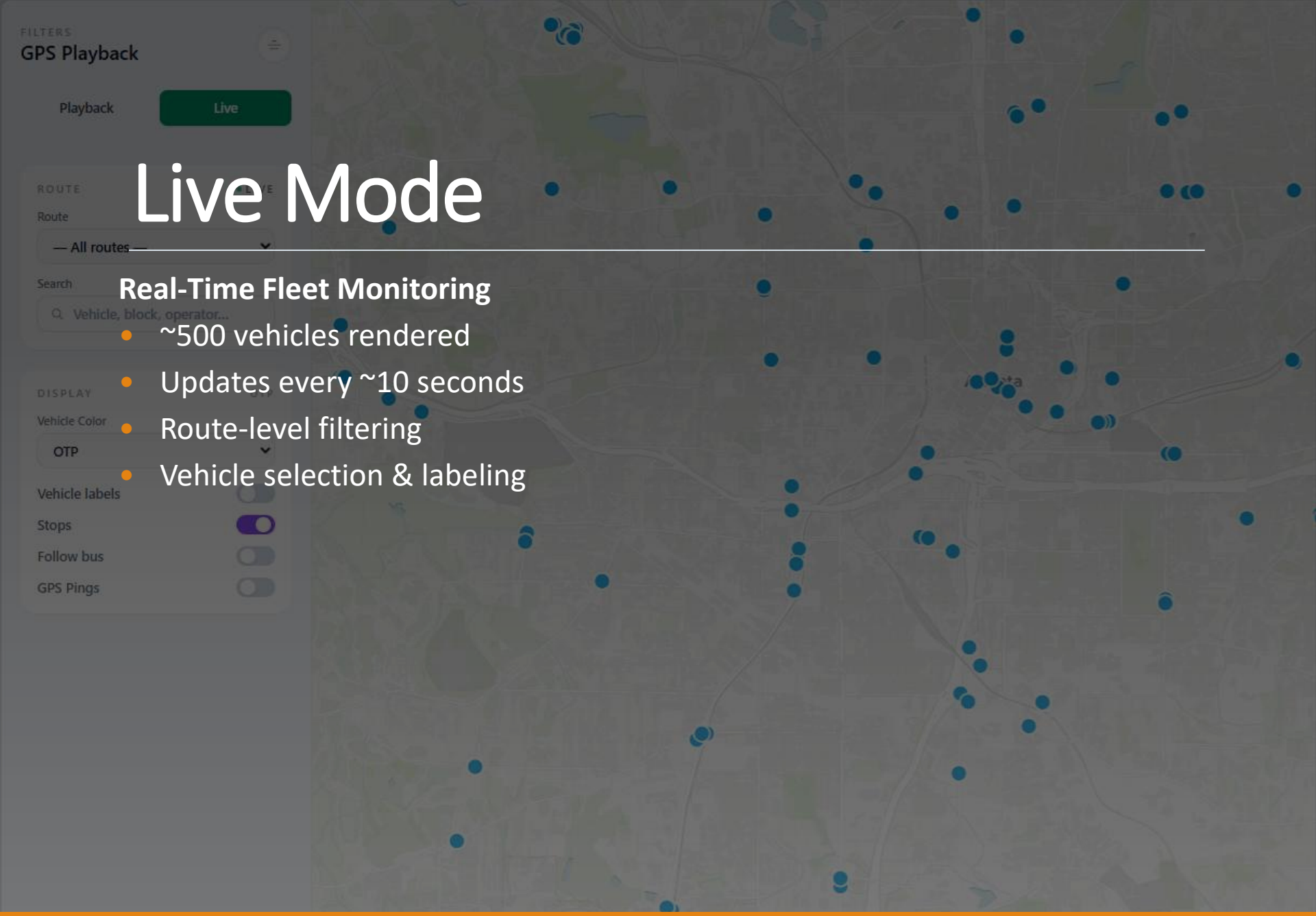




# Frontend Design

**Visualization Stack**

- React
- MapLibre GL
- GeoJSON sources
- WebSocket streaming

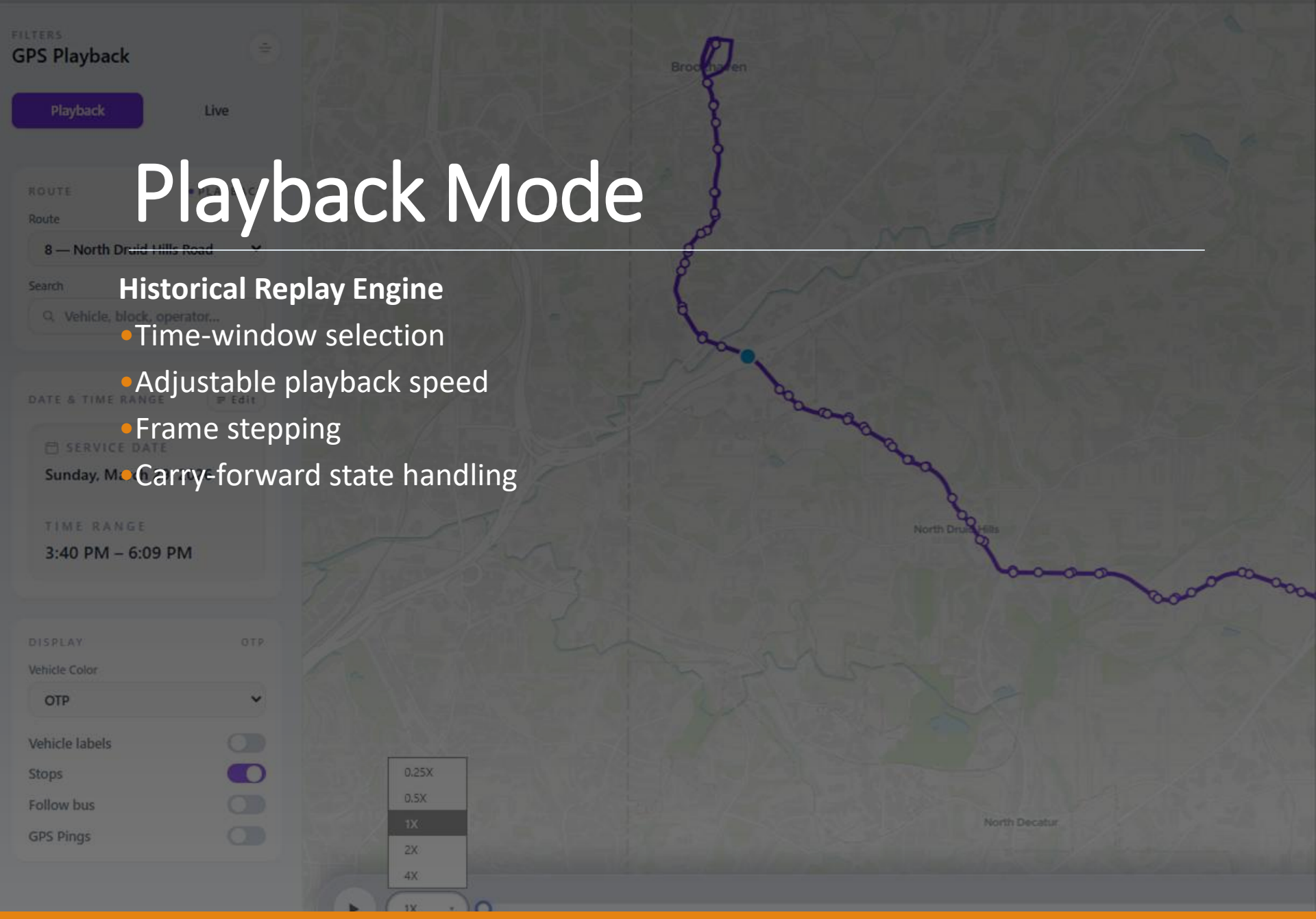





# Live Mode

**Real-Time Fleet Monitoring**

- ~500 vehicles rendered
- Updates every ~10 seconds
- Route-level filtering
- Vehicle selection & labeling




# Playback Mode

**Historical Replay Engine**

- Time-window selection
- Adjustable playback speed
- Frame stepping
- Carry-forward state handling

# Challenge 1:
# State Synchronization & Filtering

**Challenge:**

- Vehicle freezing
- Duplicate keys
- Filtering not reflected
- Stale animation state

**Solution:**

- Deterministic vehicle key mapping
- Cleanup of inactive vehicles
- Snapshot vs incremental update analysis

# Challenge 2:
# Large Fleet Rendering

**Challenge:**

- ~450-550 active vehicles
- Frequent GeoJSON updates
- Risk of fram frops

**Optimizations:**

- useRef state storage
- Single GeoJSON source reuse
- Avoid React re-render loops

# Case Study 1: Operational Scenario

**Route Monitoring & Analysis**

**Scenario:**

Analyze Route 1 during peak hours (6–9 AM)

**System Actions:**

Apply route filtering

Load historical playback

Animate vehicle movement

**Observations:**

Uneven spacing between buses

Delays at intersections

Speed variation across route

**Insight:**

System enables **operational diagnostics**

Helps identify service reliability issues

# Case Study 2: System Limitations & Future Work

**Multi-Source Data Selection**

Data from multiple sources (GTFS / Swiftly / TransitMaster)

Potential inconsistency or delay

**Current Approach:**

Backend normalization

Use **latest timestamp as source of truth**

**Future Improvement:**

Source prioritization logic implementation

Data quality scoring

**Frontend Animation Accuracy**

Data arrives every 10 seconds

Requires estimation between points

**Current Approach:**

Interpolation engine

**Future Improvement:**

Predictive movement modeling

# Future Work

**Planned Extensions**

- AWS cloud migration
- Redis caching layer
- AI-based delay prediction
- Anomaly detection
- Headway variance modeling

# Conclusion

**Key Contributions**

- Open-source fleet intelligence platform
- Time-series optimized backend
- Smooth client-side interpolation engine
- Dual-mode live + playback visualization
- Scalable and extensible architecture