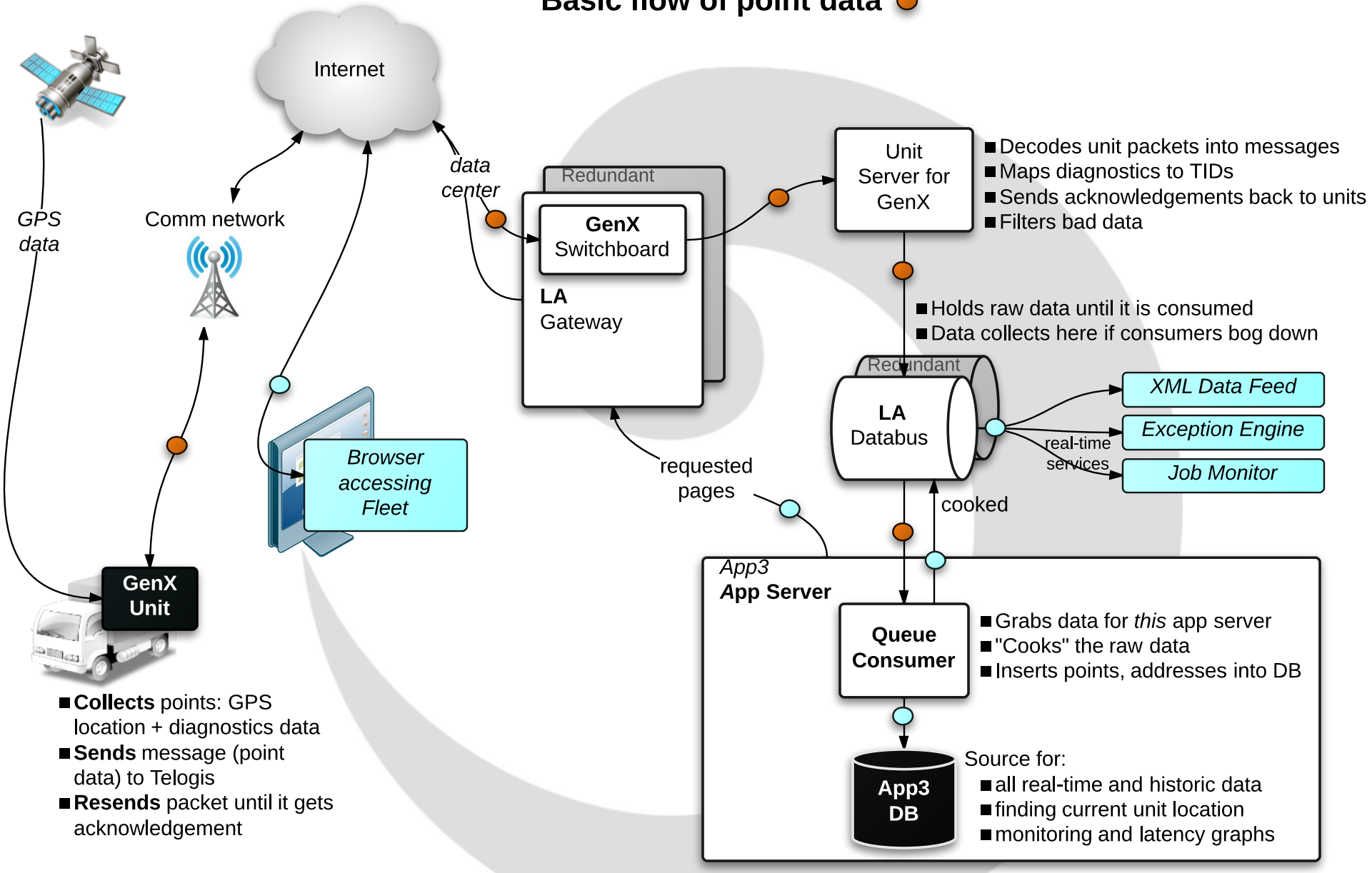
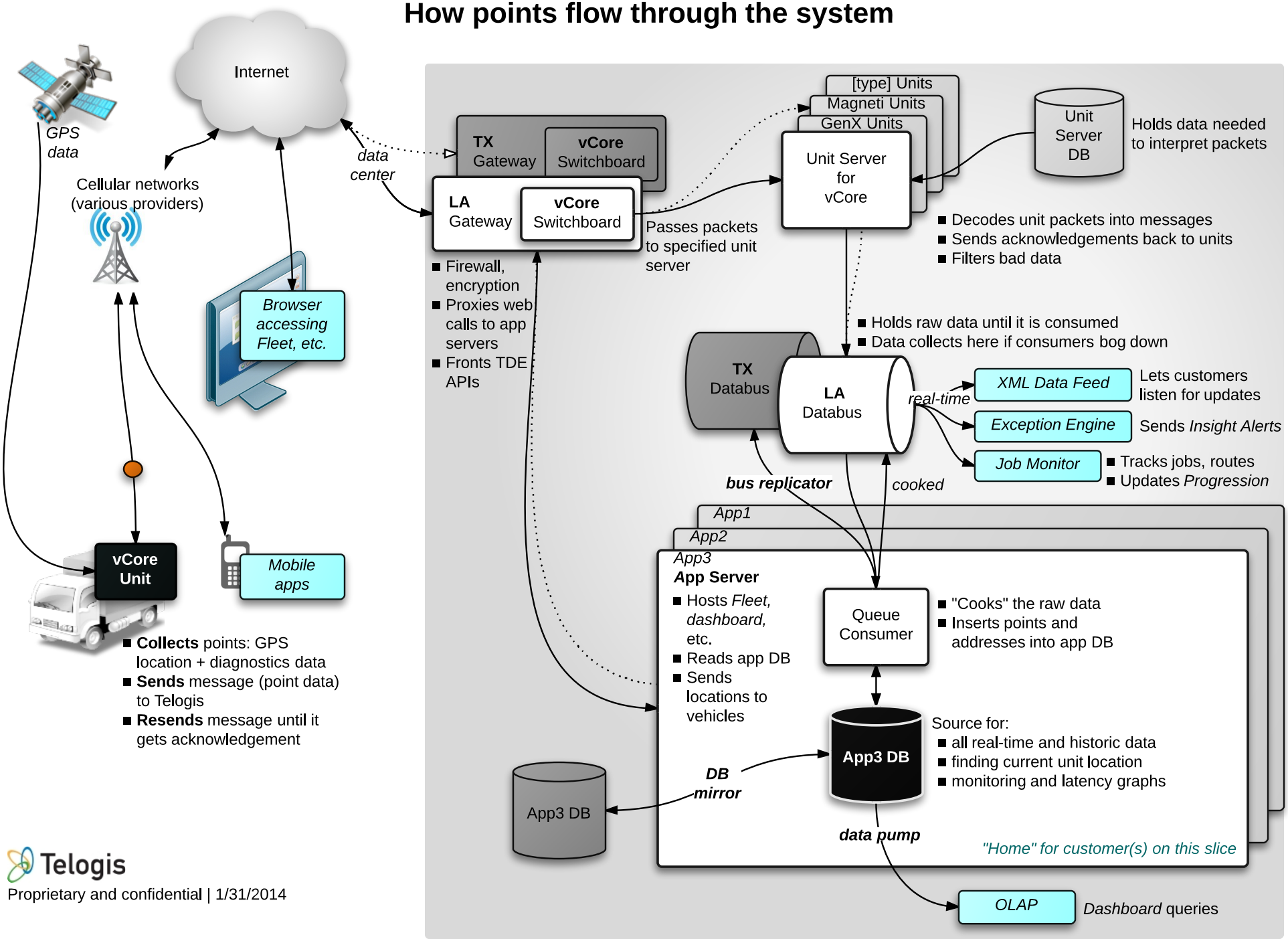
Basic flow of point data

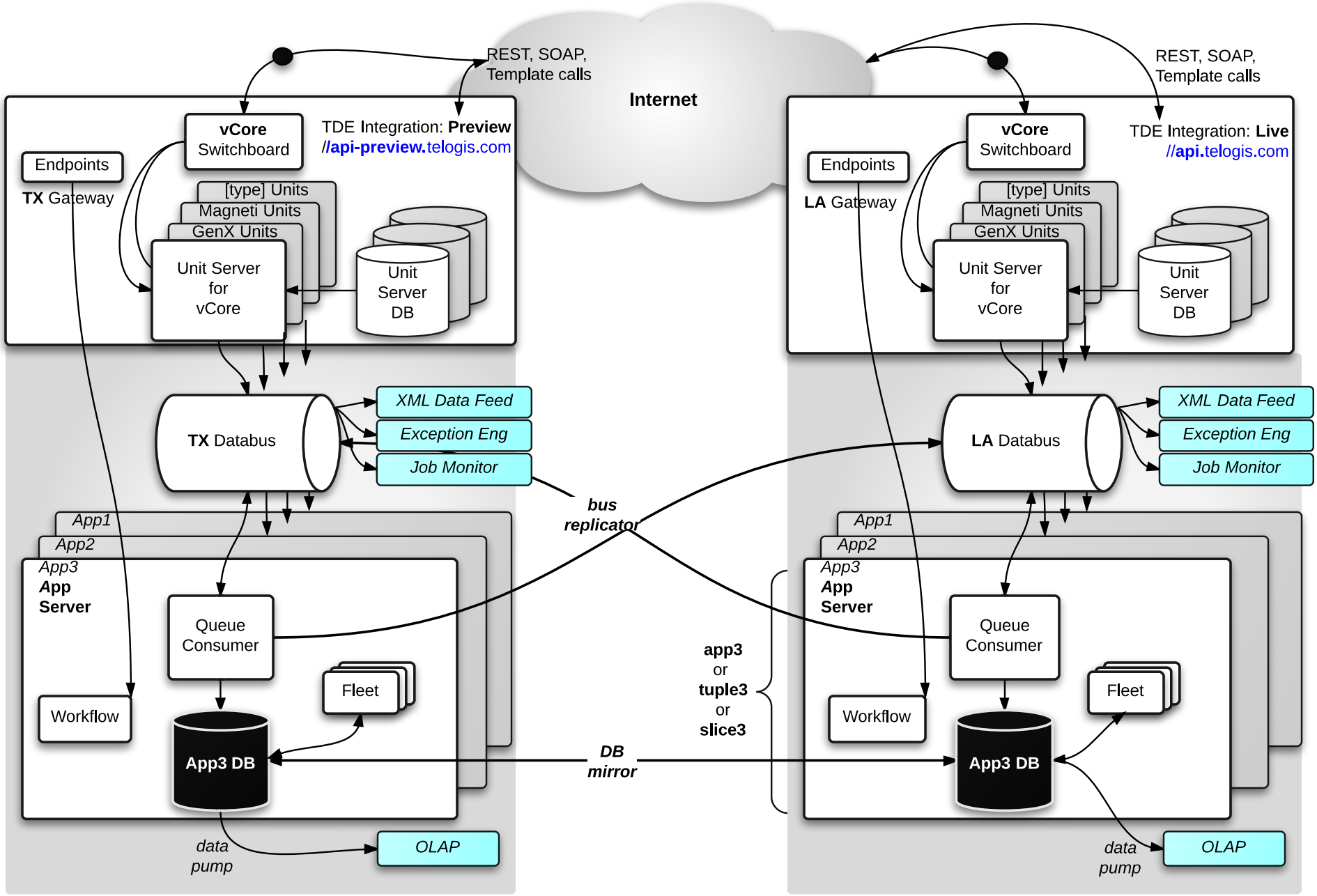


How points flow through the system

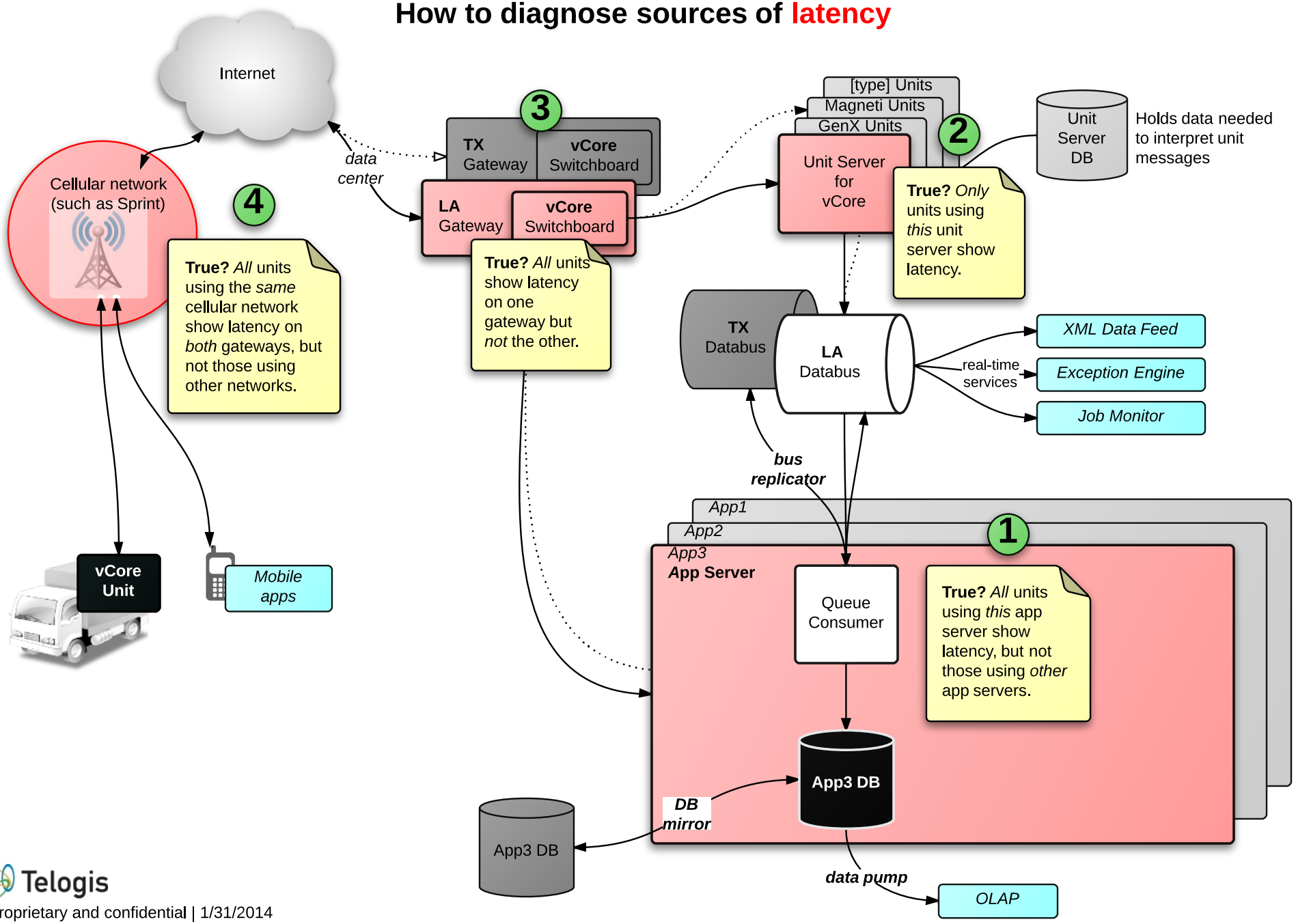


Telogis

View of the cluster



How to diagnose sources of latency



System Components and Terms

Telogis

Cluster
Telogis production cluster is 2 data centers, operating in parallel and mirroring each other: TX and LA

Tuple (slice, app)
… a collection of Linux or Windows app servers mirrored across LA/TX. Designated home for one or more customer "tenants".

Homesiding
Ensures that all traffic for a customer is on the unit server in same data center. If one side goes down, all traffic switches over.

Gateway
Offers firewall and encryption. Handles [1] UDP messages from units (LMS Location Messaging System devices) and [2] HTTP web requests from users, using reverse proxy to app servers. Fronts collections of mirrored app servers in both data centers. Has a state DB, soon to be on a separate server. If overloaded, system backs up.

Integration (TDE) Server
Windows box in each data center that runs all the APIs. No mirroring of URL for API: LA is Live URL, TX is Preview URL. TDE APIs releases occur weekly.

Switchboard
Gateway process that passes each message to the correct unit server (per type of unit, such as GenX).

Application Servers
Runs web-based apps such as Fleet, Report Generator, Dashboard. Each app server has its own dedicated DB on Postgres. Multiple app DBs reside on a DB server farm.

Unit Server
Each type of unit has its own server, which all run in parallel on a gateway and get messages round-robin. It filters out bugs and bad unit data and transforms unit-specific packets from the Switchboard into standardized ones. It keeps state on the unit for transformation and sends transformed messages to the **RabbitMQ** (messaging middleware) bus, for broadcast.

Databus
Holds raw data from all unit servers until it is consumed downstream. Data collects here if consumers cannot process it fast enough. Feeds the Realtime Data Feed.

Queue Consumer
Runs on a specific app server to "cook" its incoming unit messages. It inserts points and address data into the app server's DB and sends cooked data to Job Monitor, Exception Engine, Bus Replicator.

Job Monitor
Runs on a specific app server to track vehicles/drivers during and completing jobs, and job routes. Updates Progression in real time.

Exception Engine
Runs on a specific app server to issue Insight Alerts (e.g., speeding). It monitors unit messages for conditions set by the customer as exceptions (speed > 65 mph).

Bus Replicator
Dumps cooked messages onto bus of the mirrored data center, where that Queue Consumer pushes into the correct app server and DB slice.

DB Mirror
Set of table triggers on the app DBs to copy changes into a pending table. Process runs on the DB server that burns through pending table. Custom routine that does folding. Doesn't mirror points.

Messages
Unit message: a snapshot report from a unit: GPS location + vehicle-specific diagnostic data.
Raw: messages generated by unit server directly from unit reports. Routing key of raw.<location>
Cooked: messages after processing by a Queue Consumer, which might republish them multiple times.
Poll: messages relaying a user request to poll position from a unit poll.<location>.<unittype>. Polls are real-time, not persisted.

Dashboard
Presents OLAP data that is all KPIs set up per company exceptions and hierarchy. It has OLAP cube servers with a data pump taking data from PSQL into OLAP at midnight.

Fleet
ASP.Net and JS App, whose reporting puts high load on the system. Its Report Generator taxes DB servers.

App12
Special app server tuple for AT&T that runs a custom Fleet code base (reunified in Fleet 11). Has its own bus translator to change cooked messages to AT&T format.