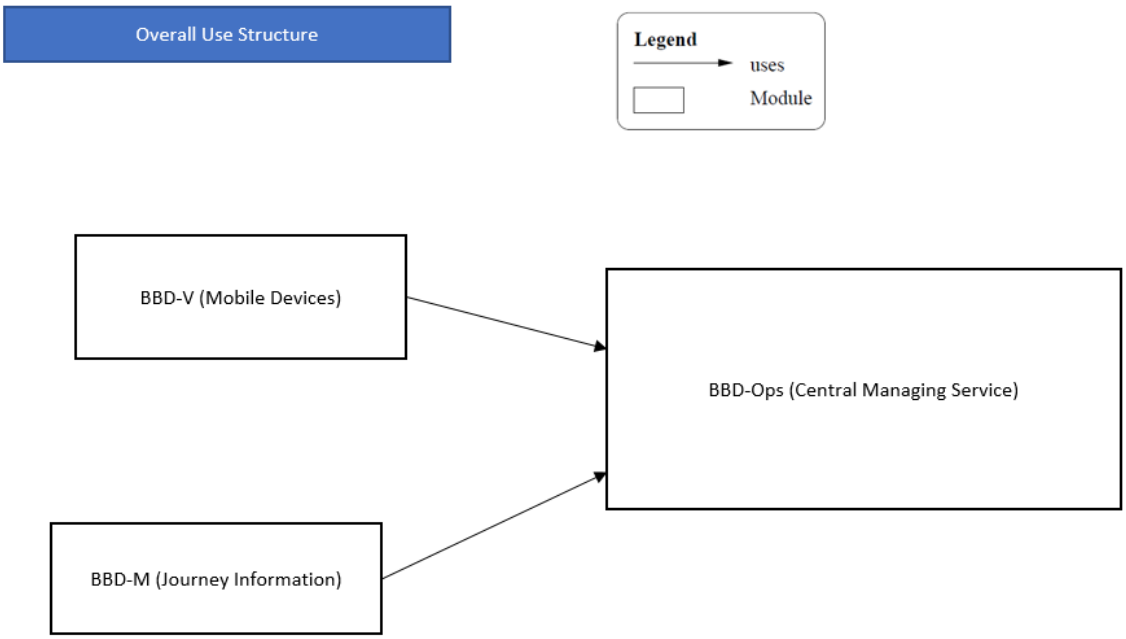
840454023, elee353

Architecture Section

Module structures

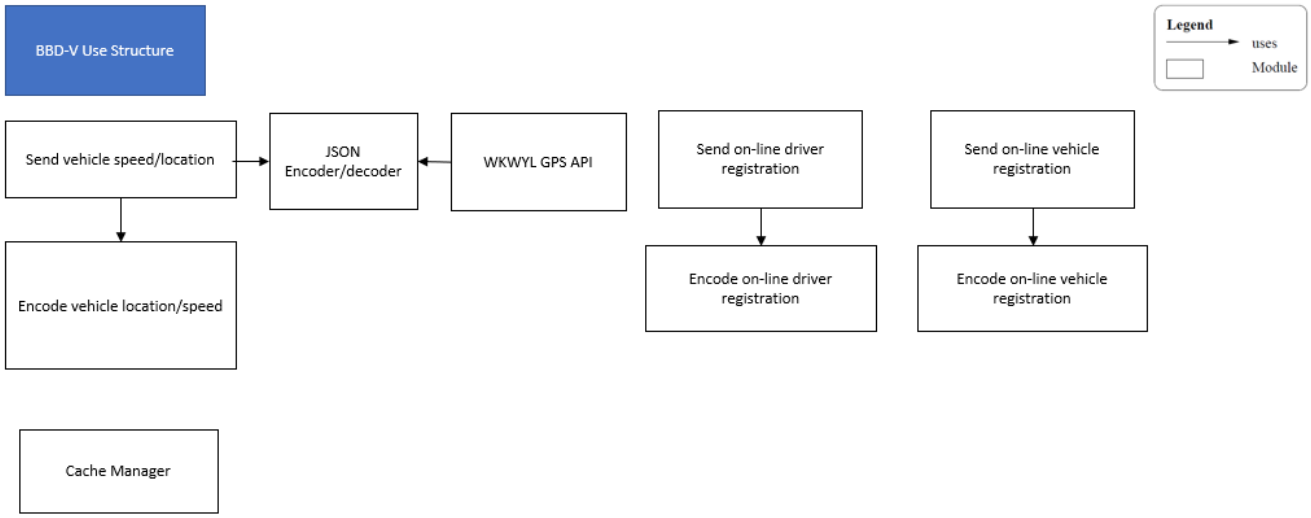
Overall Module Structure



The system consists of three major components: BBD-V, BBD-M, and BBD-Ops.

BBD-V is the mobile app installed on the mobile devices for recording the journeys of vehicles. BBD-M is the app for accessing the journey information by relevant users. BBD-Ops is the central service for managing and distributing the information gathered by the apps.

BBD-V Module Structure



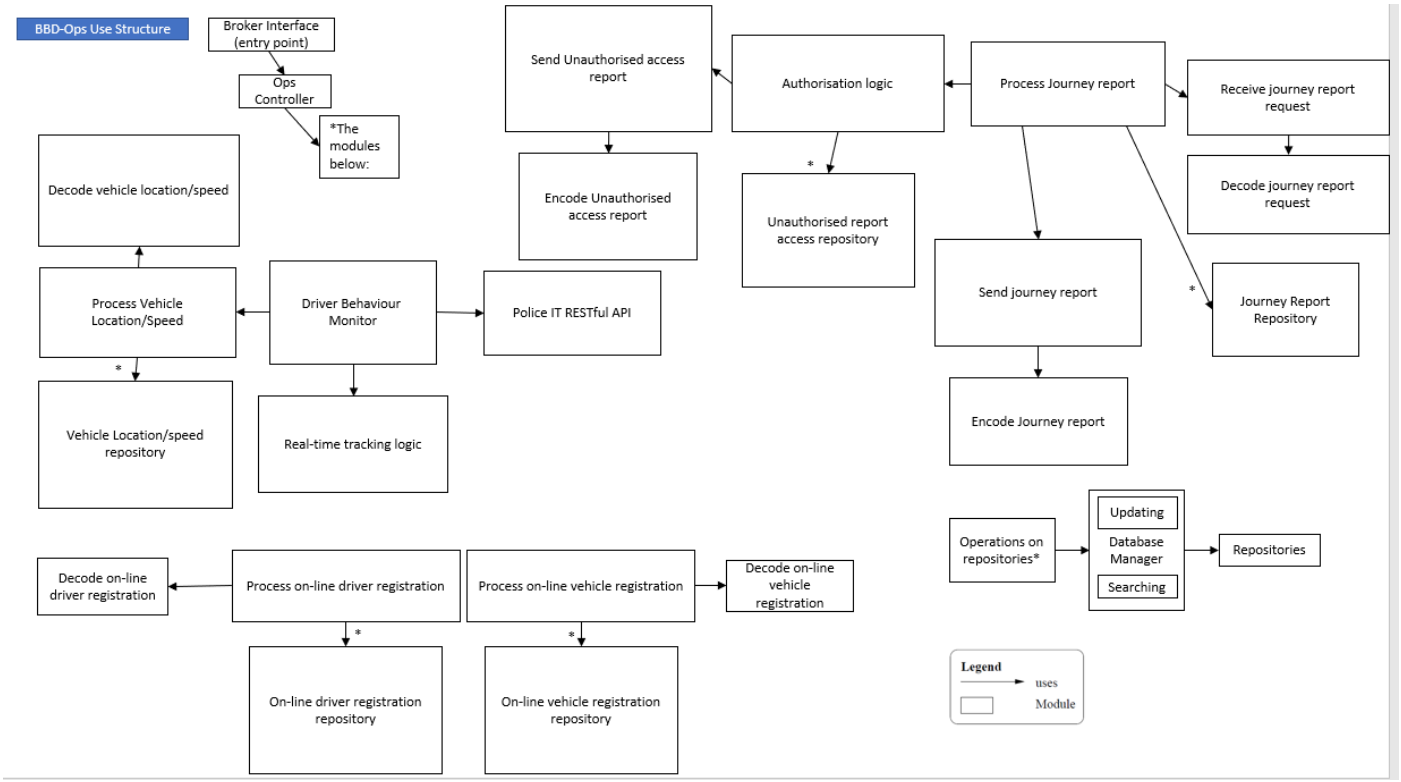
The ‘send vehicle speed/location’ module uses the ‘JSON encoder/decoder’ and the WKWYL GPS API to receive real-time details about the speed limit and road-layout for the specified GPS location. The ‘send vehicle speed/location’ module also uses the ‘encode vehicle location/speed’ module to package the speed/location data before sending. It then uses the network API included in the OS to send the packaged data.

The ‘send on-line vehicle registration’ module is responsible for sending encoded vehicle registration information. The ‘send on-line vehicle registration’ module uses the ‘encode on-line driver registration’ module to package the data before sending it.

Similarly, the ‘send on-line driver registration’ module is responsible for sending encoded vehicle registration information. The ‘send on-line driver registration’ module uses the ‘Encode on-line driver registration’ module to package the information before sending.

The ‘cache manager’ is responsible for caching speed/location information and registration information on the client side.

BBD-Ops Module Structure



The BBD-Ops contains a broker interface to be used by external clients. Requests or data passed in are then filtered by the BBD-Ops controller to select appropriate modules to be used.

The ‘process vehicle location/speed’ module uses the ‘decode vehicle location/speed’ module to unpack the speed/location data received. The ‘process vehicle location/speed’ module then stores the received and calculated location/speed data using the ‘vehicle location/speed repository’ module.

It is worth noting that it is implied that all repository operations are through a database manager. It contains submodules for handling operations such as searching and updating efficiently.

The ‘driver behaviour monitor’ module receives the data from the ‘process vehicle location/speed’ module. The ‘driver behaviour monitor’ module then decides if it should perform real-time tracking or report drivers to the police. The decisions are made based on factors such as current speed limits, unnecessary line changes, and unnecessary line-crossing.

The ‘real-time tracking logic’ module is used by the ‘Driver Behaviour Monitor’ module to allow real-time tracking for the police and parents for targeted drivers.

The ‘police IT RESTful API’ module is used by the ‘Driver Behaviour Monitor’ module to report erratic driving behaviour or excessive speed to the police immediately.

The ‘process on-line driver registration’ module processes the driver information received after unpacking the packet using the ‘decode on-line driver registration’ module. The ‘process on-line driver registration’ module then persists the driver information obtained using the ‘on-line driver registration repository’.

The ‘process on-line vehicle registration’ module processes the vehicle information received after unpacking the packet using the ‘decode on-line vehicle registration’ module. The ‘process on-line vehicle registration’ module then persists the vehicle information obtained using the ‘on-line vehicle registration repository’.

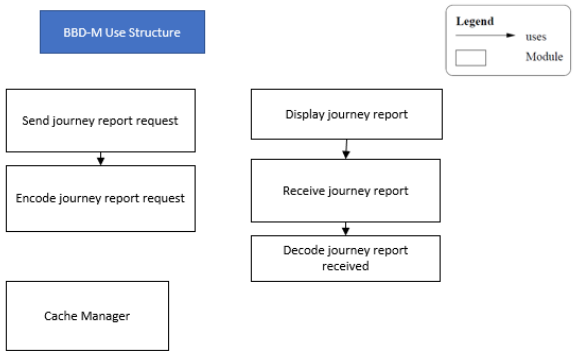
The ‘process journey report’ module uses the ‘journey report repository’ to persist the reports generated. The ‘process journey report’ module also uses the ‘receive journey report request’ module to receive journey report requests. The ‘receive journey report request’ module uses the ‘decode journey report request’ to unpack the request packet received.

The ‘process journey report’ module uses the ‘send journey report’ module and the ‘encode journey report’ module to package the report information and send it as a packet.

The ‘authorisation logic’ module is used by the ‘process journey report’ module to authorise journey report accesses. Unauthorised journey report accesses are persisted using the ‘Unauthorised report access repository’ module.

The ‘Send Unauthorised access report’ and ‘Encode Unauthorised access report’ modules are used to package and send the unauthorised accessor reports.

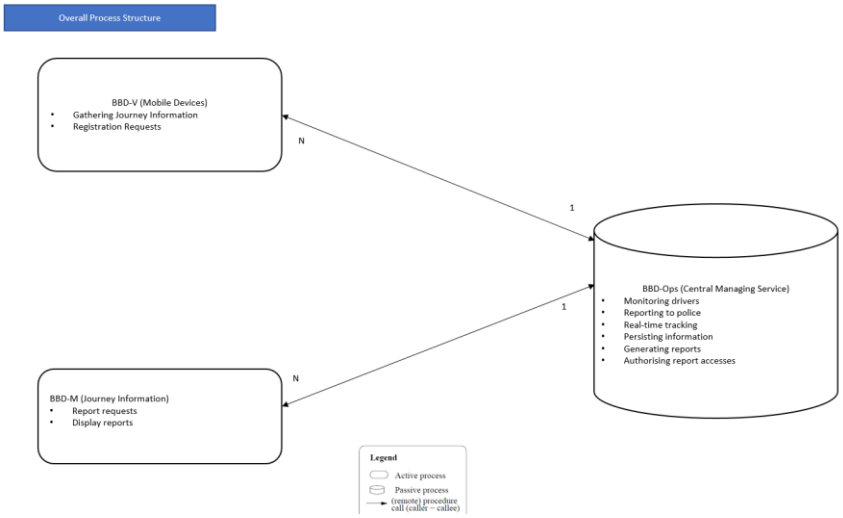
BBD-M Module Structure



The ‘display journey report’ module receives and unpackages the report packets and displays them. It uses the ‘receive journey report’ and the ‘decode journey’ modules to receive and unpackage packets.
The ‘send journey report request’ and ‘Encode journey report request’ modules are used to package and send journey report requests.
The ‘cache manager’ caches the journey reports received on the client side.

Component and Connector structures

Overall Process Structure

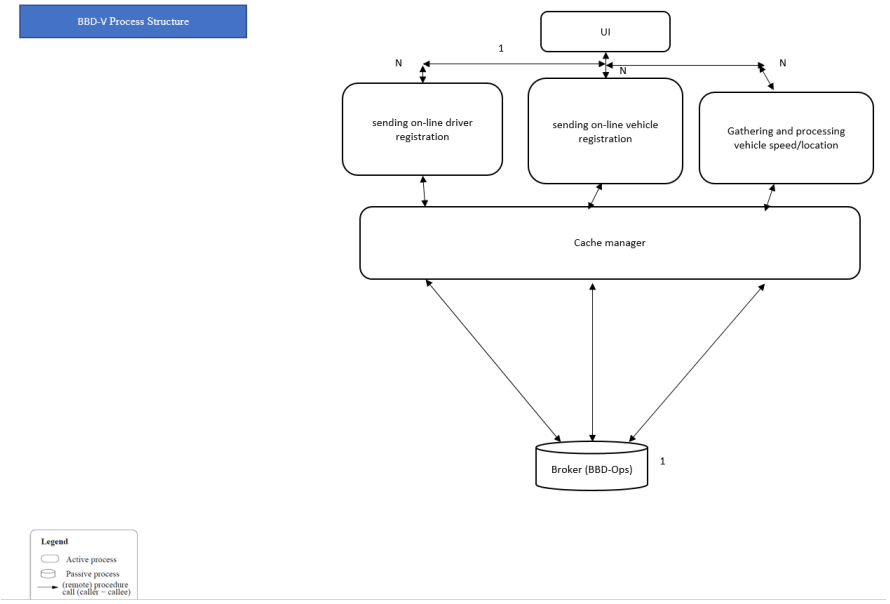


There are two different types of components: “active” processes and “passive” processes.

Active processes run independently from other components and can initiate communication similarly to a thread. On the other hand, passive processes cannot initiate communication and are regarded as “servers”.

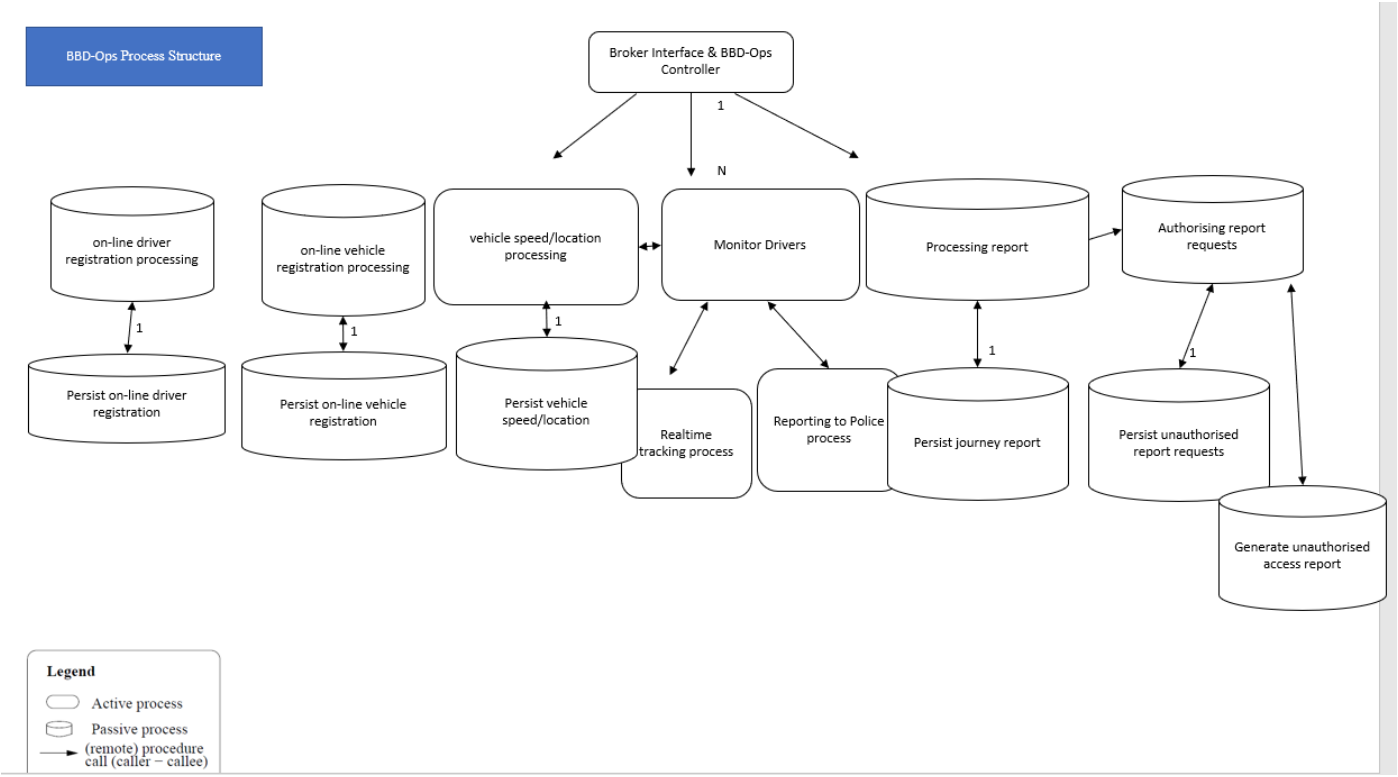
There are multiple BBD-V and BBD-M clients communicating with the BBD-Ops. The BBD-Ops can be duplicated and scaled to handle an increased workload.

BBD-V Process Structure



There is only a single UI process communicating with the following processes: sending on-line driver registration, sending on-line vehicle registration, gathering and processing vehicle speed/location. There are multiple instances of these processes communicating with the UI process. There are multiple caches used to store relevant information such as cookies. The caching operations are handled by the cache manager which control available cache resources on the client side. The registration requests and speed/location information are then sent to the broker, which is the BBS-Ops in this context.

BBD-Ops Process Structure

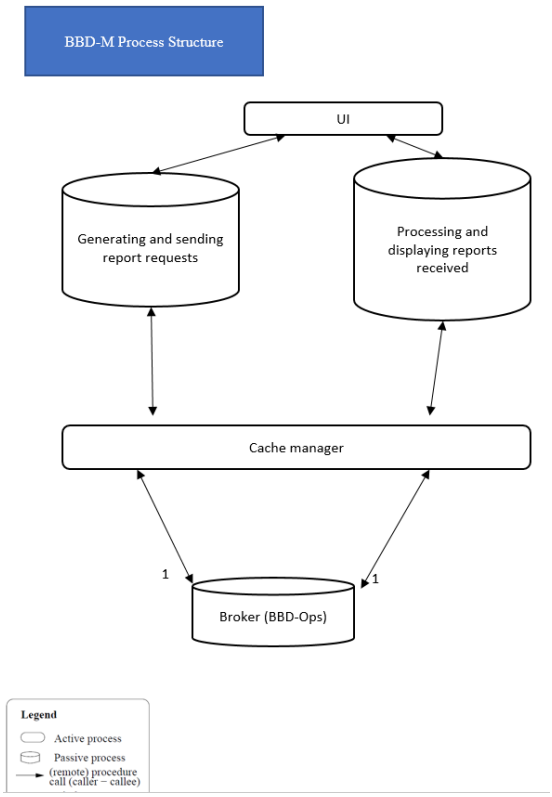


The BBD-Ops is a ‘broker’ and contains a single Broker interface. External clients interact with this broker interface.

There are multiple processes running. These processes include ‘on-line driver registration processing’, ‘on-line vehicle registration processing’, ‘vehicle speed/location processing’, ‘Monitor Drivers’, ‘Processing report’, and ‘Authorising report requests’. There can be multiple processes for the reporting, tracking, and generation of the unauthorised access reports. These processes all communicate with the ‘BBD-Ops controller’ process.

All communication links are synchronous.

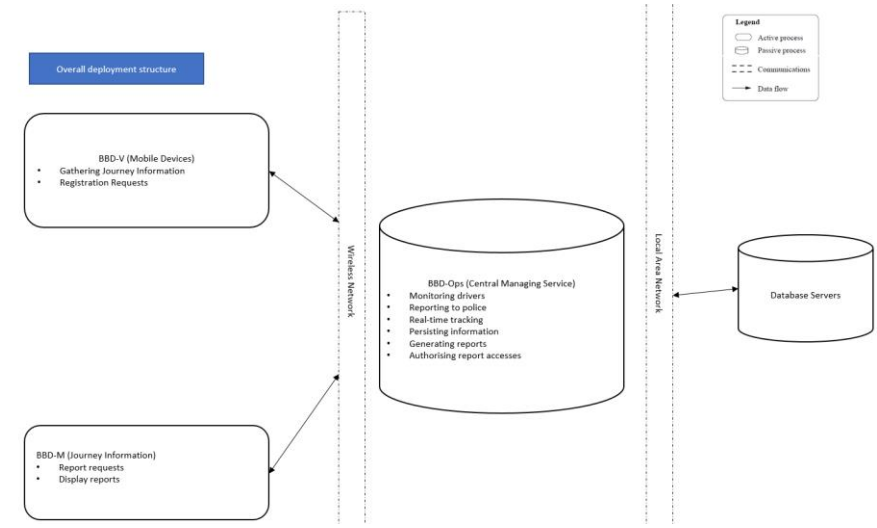
BBD-M Process Structure



BBD-M has a single UI process, 'Generating and sending report requests', and 'Processing and displaying reports received' processes. The UI process communicates with multiple report request and report received processes. The information related to the processes is cached. These processes then communicate with the single broker process to send or receive data.

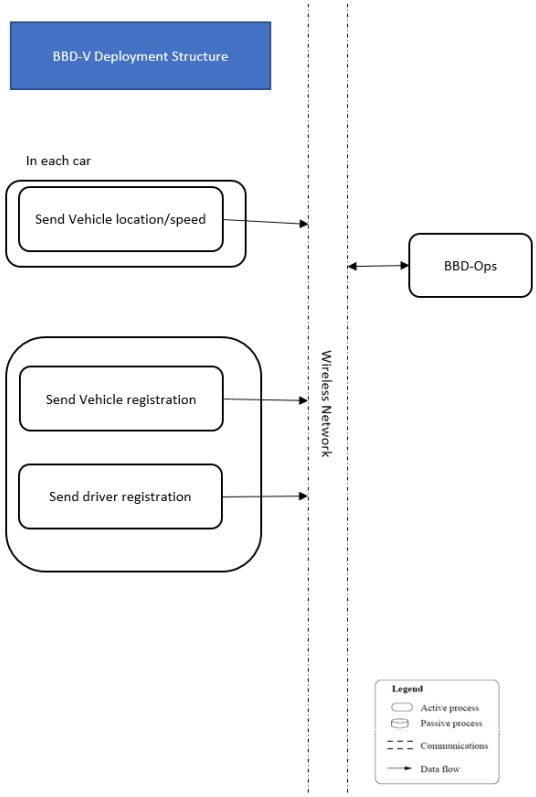
Allocation Structures

Overall Allocation Structure



The BBD-V and BBD-M clients communicate with the BBD-Ops using wireless communication. The BBD-Ops communicate to the database servers using Local Area Networks.

BBD-V Allocation Structure

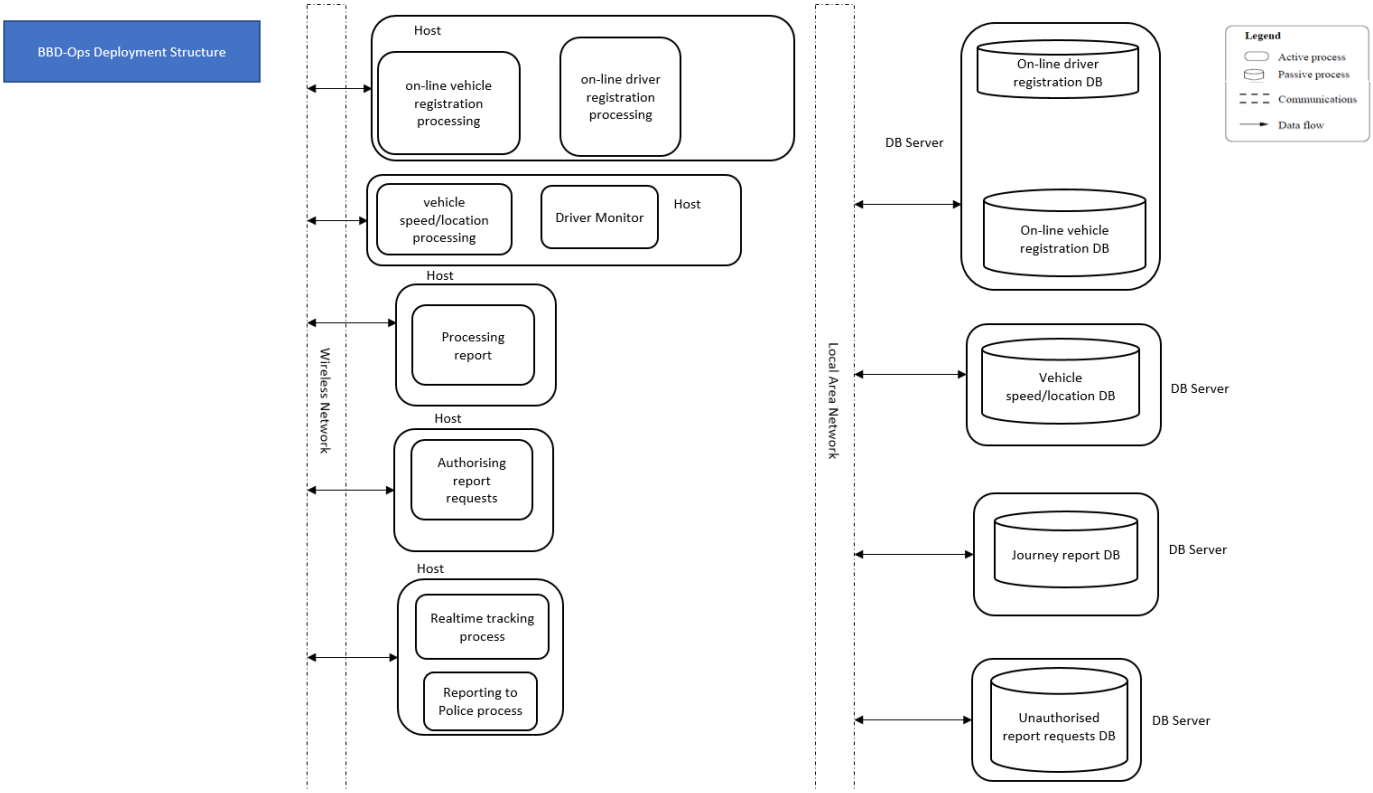


In each car, at least one ‘send speed/location’ process runs on a mobile device.

‘Send driver and vehicle registration’ processes can be run on multiple devices.

These processes communicate to BBD-Ops via a wireless mobile network.

BBD-Ops Allocation Structure

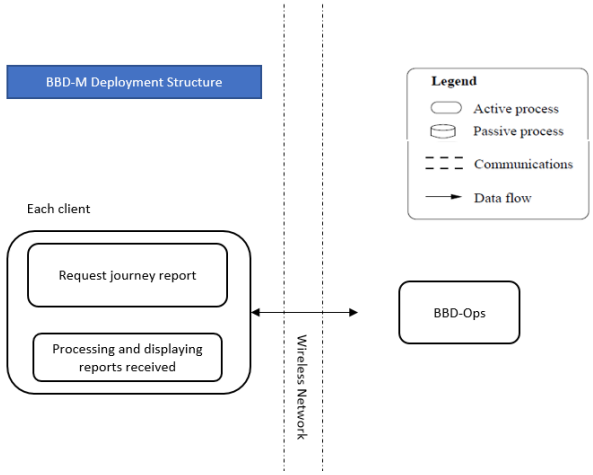


BBD-Ops receives data and requests via the wireless mobile network. The various processes are hosted on separate hosts. The registration request processes are co-located on one host.

The ‘vehicle speed/location processing’ and driver monitor processes are co-located. This helps to improve latency. A similar strategy is applied to the ‘Real-time tracking’ and ‘reporting to police’ processes.

BBD-Ops then use Local Area Network to communicate with the database servers. The database servers include the following: registration database, vehicle speed/location database, journey report database, unauthorised report request database.

BBD-M Allocation Structure



The 'Request journey report' and 'Processing and displaying reports received' processes run on each mobile client. They communicate wirelessly with the BBD-Ops to retrieve journey reports.

Patterns

Client-Server / N-Tier Systems

The overall software architecture of this system is a Client-Server architecture combined with the Layered architecture. The system consists of an application server, a large number of clients, and a database.

The first tier is a presentation tier which deals with the interaction with the user. The presentation tier is also known as the client. There can be a large number of clients accessing the server at the same time. The clients here are thin clients. They only gather speed/location information and send registration requests.

The second tier is an application tier which processes the requests of all clients. It is the actual web application that performs functionalities specific to the BBD application. It does not store the persistent data itself and contacts the database server whenever it needs data instead.

The application tier in this system is designed to be stateless. This allows server duplication for scalability. The third database tier contains the database management system that manages all persistent data.

Broker

The BBD-Ops is a broker. It coordinates communication between requester of service and the server.

Pipe-and-Filter

A lot of the data is transformed serially in the system, for example, the results processed by the 'vehicle speed/location process' is used by the driver monitor process. The pipe-and-filter is applied to carry output from one filter to the input of the next.

Tactics

Performance Tactics

increase computational efficiency

There are several places where the use of an inefficient algorithm may result in the system not meeting the performance requirements, such as the authorisation algorithm in BBD-Ops. This can be improved by selecting efficient algorithms and data organizations.

In addition, if the repositories are indexed by driver identifier or vehicle identifier and use an efficient data structure for searching on the identifier, such as a B+ tree, then updating the repository with the new locations and speeds should exhibit logarithmic (O(log(N))) performance.

The location or map information can be cached and pre-processed beforehand. This off-line pre-processing reduces the computation needed and amount of data required

reduce computational overhead

Co-locating active and passive processes reduces the overhead of the remote communication that would be required if they are not co-located.

manage event rate
There is a choice to be made about the frequency in which the BBD-V sends out its location and speed to BBD-Ops.
For the drivers that need to be tracked in real-time, the event rate is one per second, but for drivers who are decelerating to stop at red lights, the event rate is one per 2 or 5 per second. This prevents oversampling unnecessarily and reduces data usage.

introduce concurrency
Use multiple servers and processes to process the incoming data in parallel. This will allow a lot of journey information from multiple BBD-V clients to be processed and report to be completed for BBD-M simultaneously.

increase available resources
Faster processors, additional processors, and faster networks can reduce latency. BBD-Ops uses local area network when communicating with the database servers. This is significantly faster compared to the wireless network used by the clients. BBD-Ops also has more and faster processors compared to the mobile clients. These enable faster computation and data transmission.

maintain multiple copies of either data or computations
The cache manager is used in BBD-V and BBD-M to cache data received or before it is sent. It reduces lookup time and lowers web server load. It also ensures the security of data loss or corruption.
Multiple copies of the 'vehicle speed/location' and 'driver monitor' processes are used to increase the availability to process more client requests or data received.

Multiple servers can be placed at various physical locations to increase availability.

Modifiability Tactics

Anticipate expected changes
The BBD-Ops is kept stateless and state information is stored in client cookies and databases. If the number of clients increases, servers can be replicated efficiently.

With the stateless servers used, the effects of server failures and recovery are almost unnoticeable. A newly reincarnated server can respond to a self-contained request without difficulty.

Functionalities likely to change are located in a minimum number of modules. The caching manager manages multiple caches on the client side. Locating these cache modules in the cache manager module decreases management concerns when future caching policy and implementation changes.

Adherence to Defined Protocols
Service registries support standards for description and data schemas. For example, JSON is used to communicate with the GPS API. This also increases cohesion because JSON is a widely used standard.

The client-server division of functionality separates client and server functionalities to improve flexibility and reduce coupling.

Hide information

The cache manager handles caching operations and the upper layers using it doesn't necessarily know the caching implementations.

Encapsulation / Use an Intermediary

The broker interface hides internal implementations for the BBD-Ops. This encapsulates the internal implementations and allows future modifications of these implementations.

Split module

The application tier is well split into separate modules which minimises the needs for changing implementations when the hardware changes. Hence, when new devices are released, only the modules specific to the devices need to be changed or added.

Runtime registration

New users and vehicles can be added at runtime, without having to reboot the whole system.

Security Tactics

Authorize users

Authorization is ensuring that an authenticated user has the rights to access and modify either data or services. This is managed by providing access control patterns within the system.

Audit Trail
An audit trail is a copy of each transaction applied to the data in the system together with identifying information. Audit information can be used to trace the actions of an unauthorised journey report accessor. It supports non-repudiation and provides evidence that an unauthorised request was made. It also supports system recovery.

Justification

Scenario 1

For the data to be sent to BBD-Ops, latitude, longitude (GPS coordinates) and vehicle ID will be sent in 4-byte long type. 2-byte int type will be used for speed. Therefore, a total of 14 bytes is used for each message sent.

A checksum of 4 bytes can be appended if necessary. Further encryption can also be added. However, this will increase the overall package size. In the worst-case scenario, we are sending messages to BBD-Ops every second when tracking in real-time. The data packets sent to BBD-Ops are: 14 bytes * 365 journeys per year * 60 seconds * 20 minutes = 6132000 bytes = 6.132 MB

Therefore, this is close to the data limit of 5243000 bytes. By reducing the sampling rate at certain conditions, such as when the vehicle is waiting at a red light, this data usage can be reduced. This should be done in the software implementation. The faster the driver drives, the more the sampling and event rates increase, for example, from 1 sample per 5 seconds when the vehicle is coming to a stop to 1 sample per second when the vehicle is speeding.

By applying the 'manage event rate' tactics, this software architecture can control the data usage to meet the given limit.

Scenario 2

The average 3G network has an upload speed of 10Mbps, the upload time for a packet from BBD-V to BBD-Ops takes 2.0 × 10^-8 seconds.
Assuming the WKWYL API takes a negligible amount of time to respond to requests from the client and the time taken to access the repositories is 0.5 seconds. Assuming it takes 0.1 seconds to communicate between the processes.

The processes include: vehicle speed/location processing, monitor drivers, reporting to police process. Therefore, it takes approximately 5 seconds, which the same as the 5 second restriction.

Scenario 3

Assume it takes 1 second to communicate between the processes. The processes include: generating and sending report requests, processing report, persist journey report. Assume it takes 0.5 seconds to processing and displaying reports received. Therefore, it takes approximately 8 seconds to complete the report, which is below the 10 second restriction.

Scenario 4

As discussed previously, because the architecture has been well split into various modules, only certain modules require changes and most modules can be reused. The BBD-V and BBD-M modules that likely to require changes are 'cache manager', 'sending speed/location', 'send report request', and UI. Assume it takes 8 hours to develop or update each module, it should take 40 hours or likely less to finish the development process.

Scenario 5

The report, the time it was requested, the location it was delivered to, and the identifier of device running BBD-M are persisted in BBD-Ops. These audit records can be later retrieved by the admin to trace unauthorised accessors. By authorising the accessors, we also ensure only the authenticated accessors have the rights to access either data or services.