

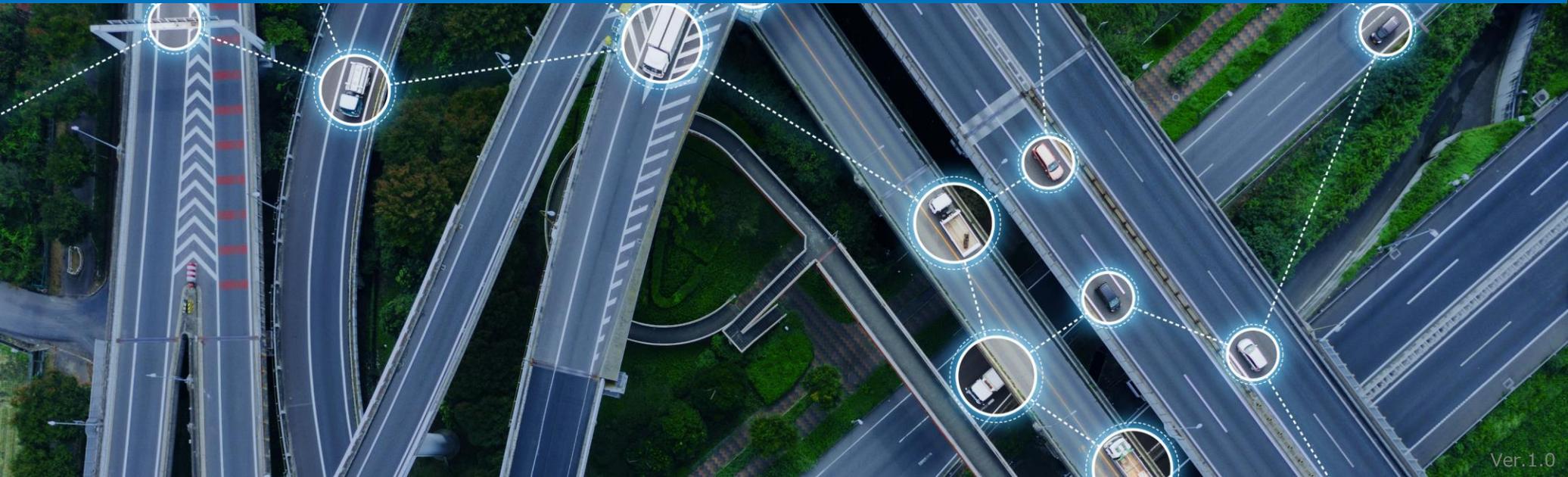
**TOYOTA**



# Technical Document

## The ICT Platform For Connected Vehicles

2021.11.29



The automobile industry is facing a once-in-a-century period of transformation at the moment. With the introduction of many connected vehicles in the market in recent years, various services have been realized for drivers, passengers, automakers, insurance companies, and repair factories. In the future, further widespread and market expansion are expected.

The spread of connected vehicles enables utilization of big data detected by sensors.

The utilization of data analysis results using ICT platform technology is expected to contribute to safe transportation (zero traffic accidents/congestion), solutions to social issues including disaster damage mitigation, and mobility services that provide new value.

This document presents an overview of technical results on the following platform system.

- High-speed/effective collection of big data generated from connected vehicles
- Verification of use cases with high technical difficulties as ICT platform for connected vehicles (for example, real world simulation in cyberspace)
- System supporting End to End verification tests with actual vehicles in the real field

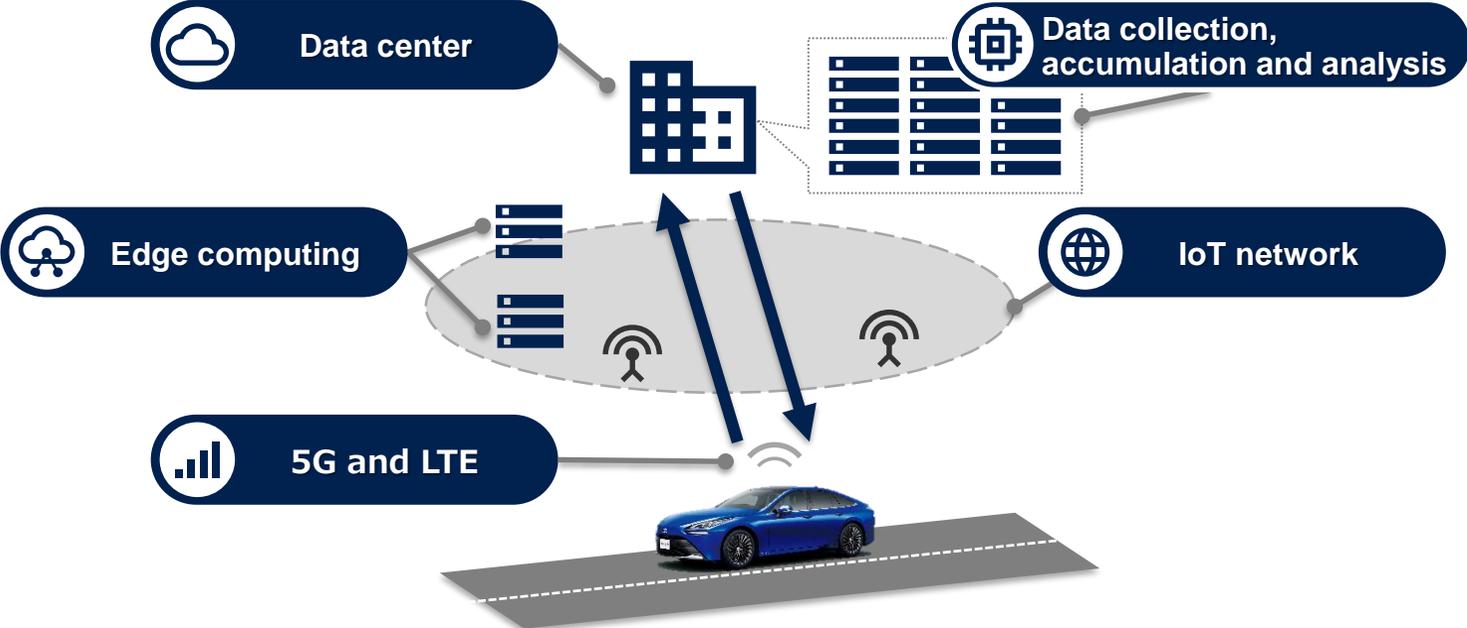
We hope this technical document will be helpful broadly not only for those engaged in connected vehicles, ICT, and mobility services, but also for other industries.

- ICT Platform for Connected Vehicles
- Field Trials
  - Overview and Architecture
  - System, Hardware, and Software Configurations
  - Use Case 1: Static Map Generation
  - Use Case 2: Obstacle Detection
  - Use Case 3: Congestion Detection
- Future Initiatives

# ICT Platform for Connected Vehicles

# Overview of ICT Platform for Connected Vehicles

The ICT platform for connected vehicles connects a vehicle with data communication modules, edge computing, and the cloud via LTE and 5G mobile networks and IoT networks. It also applies computing resources to collect, accumulate, and analyze vehicle data.



# Requirements for ICT Platform for Connected Vehicles

The major requirements for developing an ICT platform for connected vehicles are ensuring the capacity for the number of vehicles connected to the system, real-time execution, and precision, as the value and use of data depend on their being obtained in real time from a sufficiently large number of vehicles and processed with a high level of precision.

<b>Requirement 1</b>	<b>Number of connected vehicles</b>	Handling of ultra-large volumes and ultra-high transactions to collect and apply vehicle and image data from tens of millions of vehicles
<b>Requirement 2</b>	<b>Real-time execution</b>	Real-time collection and compilation of the position information of vehicles driving at high speeds to send real-time information to companies so they can provide their services
<b>Requirement 3</b>	<b>Precision</b>	Estimation of the position of landscape features, including vehicles, signs, and traffic signals, at a higher precision than that of GPS for managing the information on the data center side

# Field Trials

# Objectives of Field Trials

The objective of the field trials was to assess technologies used in cloud and communications infrastructure and to clarify requirements of next-generation vehicles toward a full-scale rollout of connected vehicles and autonomous driving cars.

## Objectives of Field Trials

1

**Determining cloud and communications infrastructure technologies for large-scale platforms and identifying technological requirements**

Determining **what cloud technologies will contribute to the development of connected vehicles and autonomous driving as well as the technological requirements for realizing a next-generation communications infrastructure**. To this end, we fully applied the expertise in design, verification, and evaluation cultivated through our collaboration to date while setting 2023 as the target year.

2

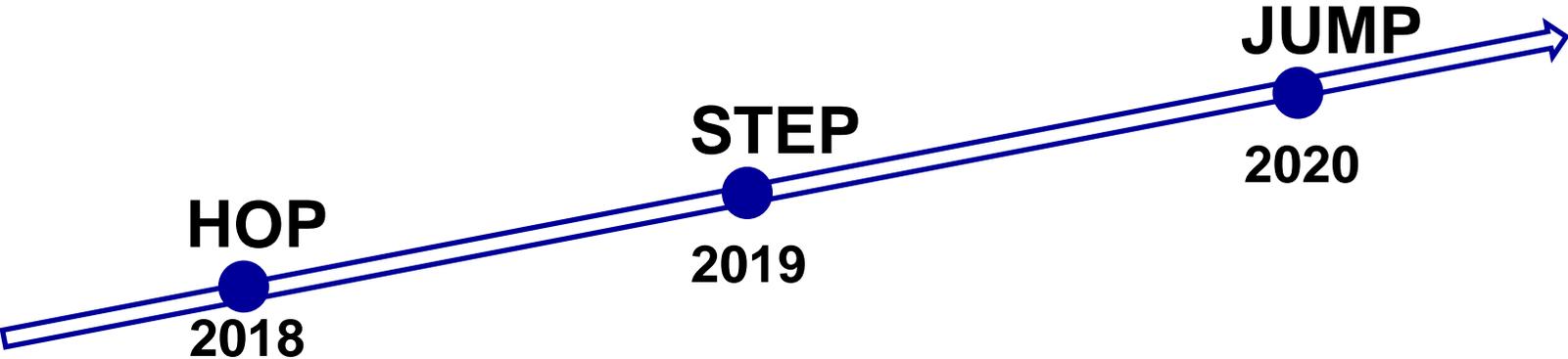
**Identifying requirements for next-generation vehicle development**

Identifying **technical restrictions on the cloud and communications infrastructure and requirements for vehicle development at an early stage, as a longer process is taken to address these needs compared to those of ICT**, toward the full-scale rollout of connected vehicles targeted in 2023 and increased integration of autonomous driving vehicles.

# Roadmap of Field Trials

The field trials were conducted by setting phased targets over a mid- to long-term period up to FY2020.

## Mid- to Long-Term Roadmap of Field Trials



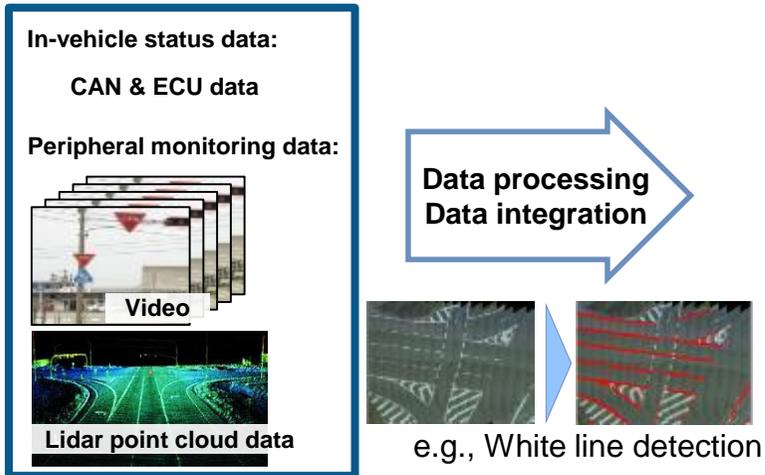
### Performance

Vehicles connected:	> 1,000,000 vehicles	> 10,000,000 vehicles	> 100,000,000 vehicles
Processing time:	About 10 minutes	About 10 seconds	A few seconds
Data processed:	> 100 items	> 1,000 items	> 10,000 items

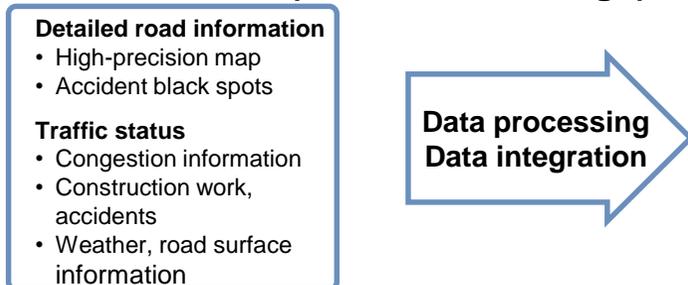
# Overview of Field Trials

The goals of the field trials include a scalability evaluation of a large-scale infrastructure technology, which is used for the cyber-physical system (CPS), toward the full-scale rollout of the connected vehicles. The top priority was to reproduce the necessary conditions for automated driving and mobility services in the future. Vehicle data was obtained from test cars driven in the Odaiba area.

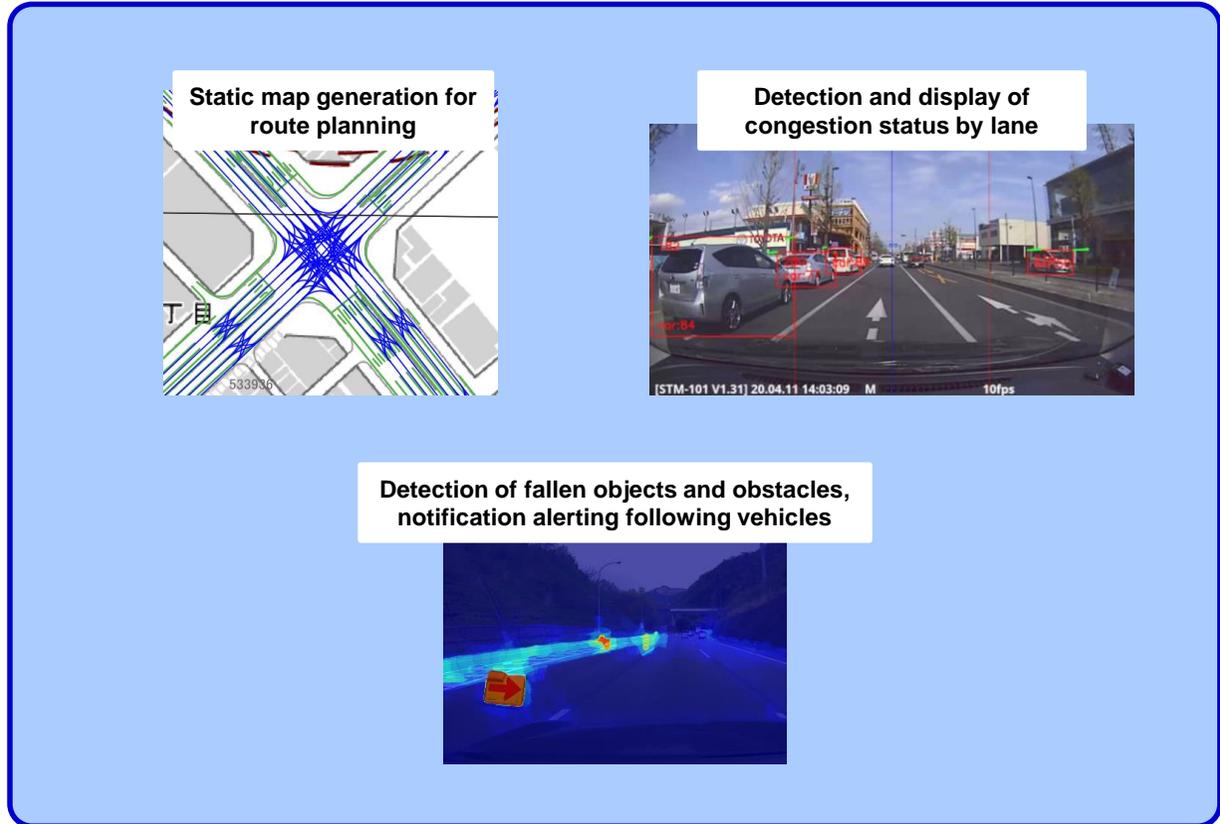
## Data Collection (from Vehicles)



## Data Collection (from Surroundings)

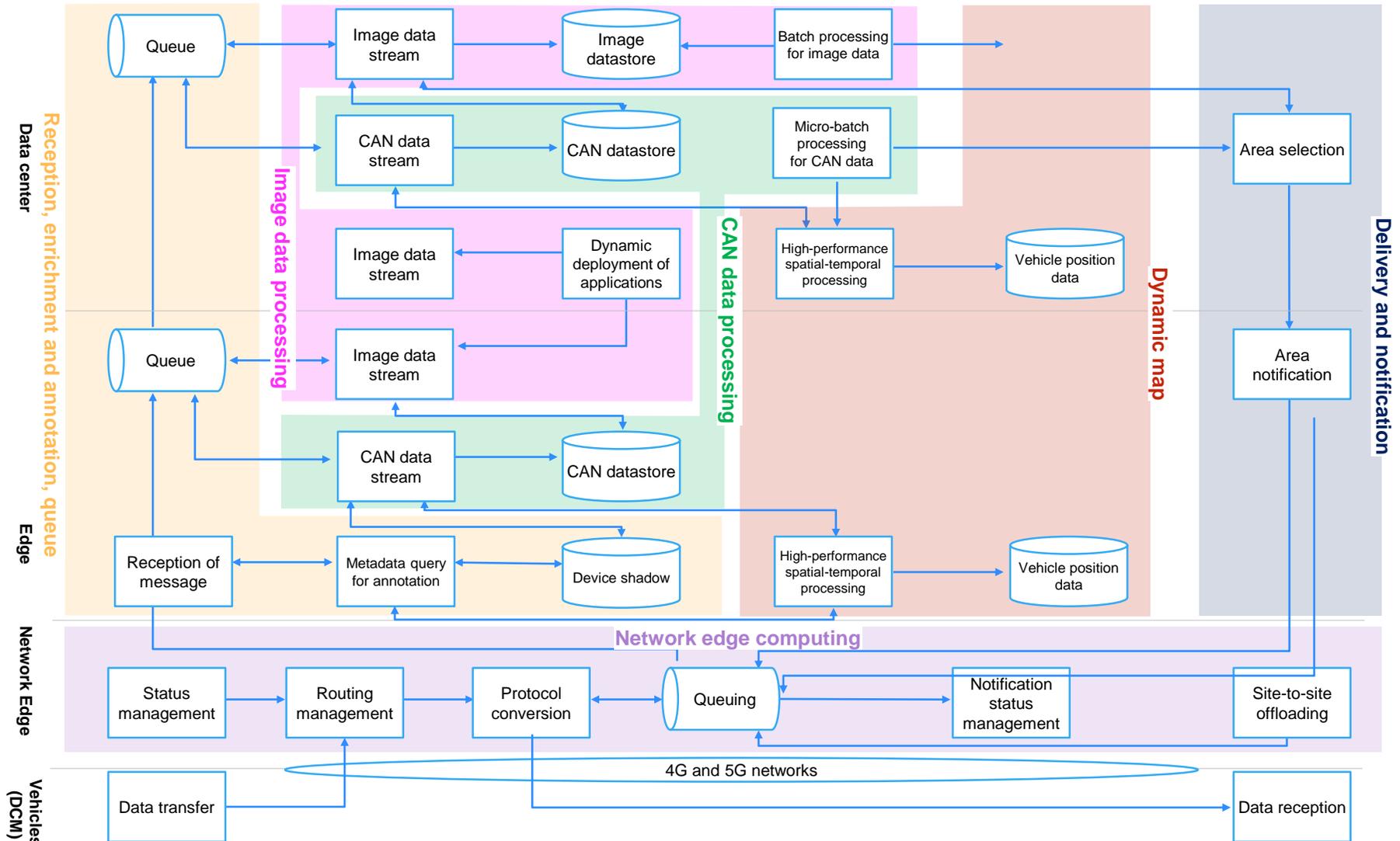


## Current Status and Prediction (Example)



# System Architecture for Field Trials

Verifications were based on the following architecture.



# Verification Based on Use Cases

# Overview of Verification Process

Verification was conducted on three use cases that focused on the number of connected vehicles, real-time execution, and precision as the key issues for developing an ICT platform for connected vehicles.

No.	Use Cases	Overview
1	<b>Static map generation</b>	Images are collected from the front-view camera of vehicles moving through the Odaiba area to generate static maps using a map generation application and image processing platform.
2	<b>Obstacle detection</b>	Vehicles moving through the Odaiba area detect obstacles in a pseudo manner and send video data of the obstacles to the cloud. The cloud side identifies the obstacles and notifies vehicles that are expected to drive on the road where the obstacles are, in order to alert them.
3	<b>Congestion detection</b>	Congestion data for a specific road is created based on data obtained from a vehicle moving through the Odaiba area, and other vehicles on the same road are requested to send video data. The cloud side determines the level of congestion for each lane based on the video and notifies vehicles that are expected to take that lane, in order to alert them about the traffic jam.

# Verification of Use Case for Static Map Generation

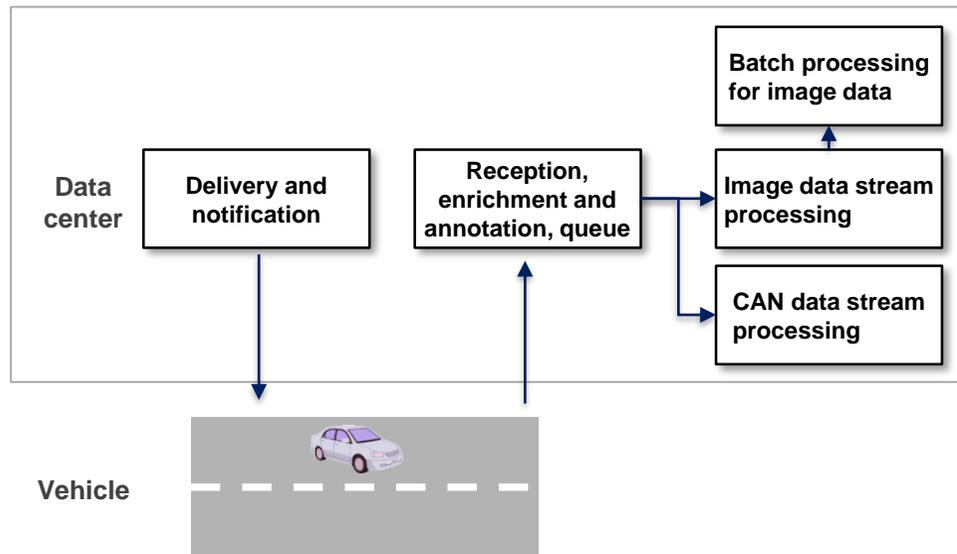
# Verification Overview

In this use case, we identified the technical issues involved in static map generation by running the system, from the collection of video taken by vehicle onboard cameras to static map storage, under actual field conditions.

## Overview

We verified the capability of collecting video data from specific vehicles in the area designated by an operator (Odaiba) for the purpose of static map generation.

We also verified that the center requested the transferring video data to the vehicles and handle the series of processing, including receiving, converting, and analyzing video data as well as the production of a static map properly.



## Verification Points

1. **Functionality**  
Confirm flow of processes involved in static map generation.
2. **Real-time execution**  
Confirm processing time from data collection instructed by the operator to static map generation.

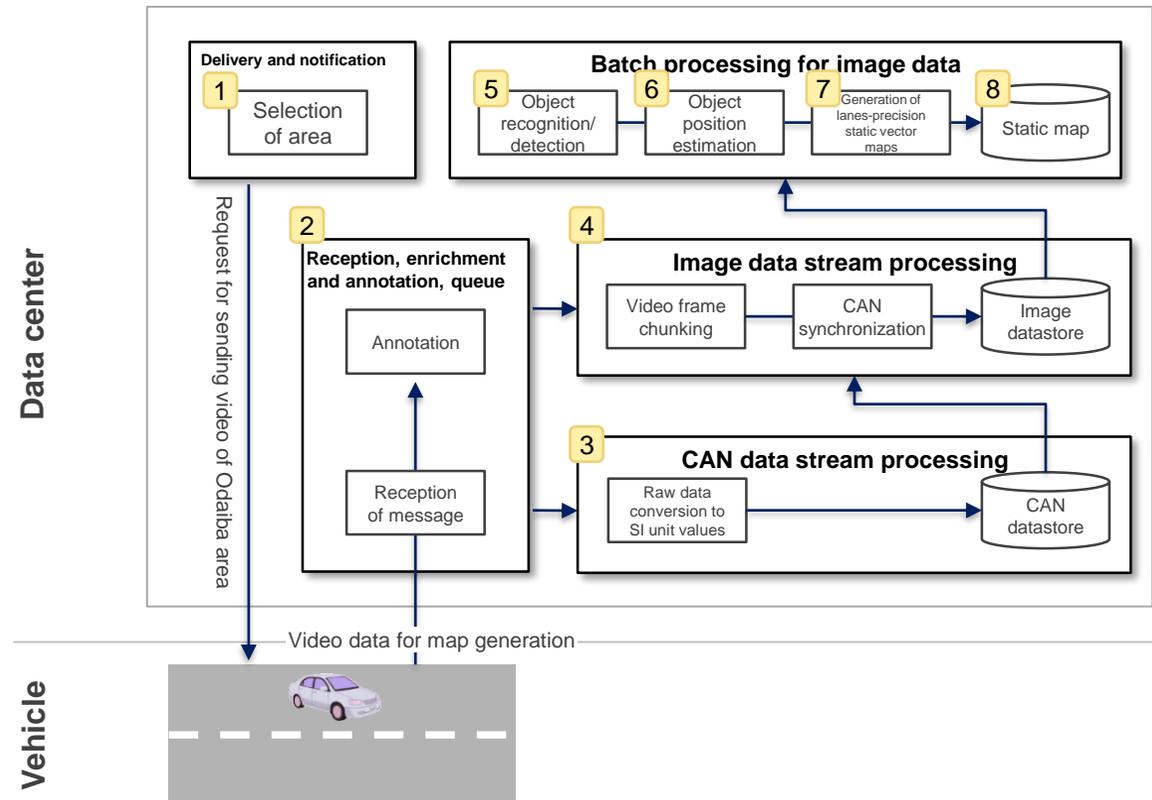
# Verification 1: Description and Results **TOYOTA**

As for functionality, we conducted verification for the series of processes (1 to 8) from notification and delivery to the generation of lane maps using test cars, and we confirmed that all processes operated as required.

## Verification Targets

Category	No.	Function
Delivery and notification	1	Processing of delivery and notification
Reception, enrichment, and annotation	2	Reception of message, enrichment, and annotation, queue
CAN data stream processing	3	Processing of CAN data stream
Image data stream processing	4	Processing of image data stream
Batch processing of image data	5	Object recognition/detection
	6	Object position estimation
	7	Generation of lane maps
	8	Storage into static map datastore

## Verification Diagram



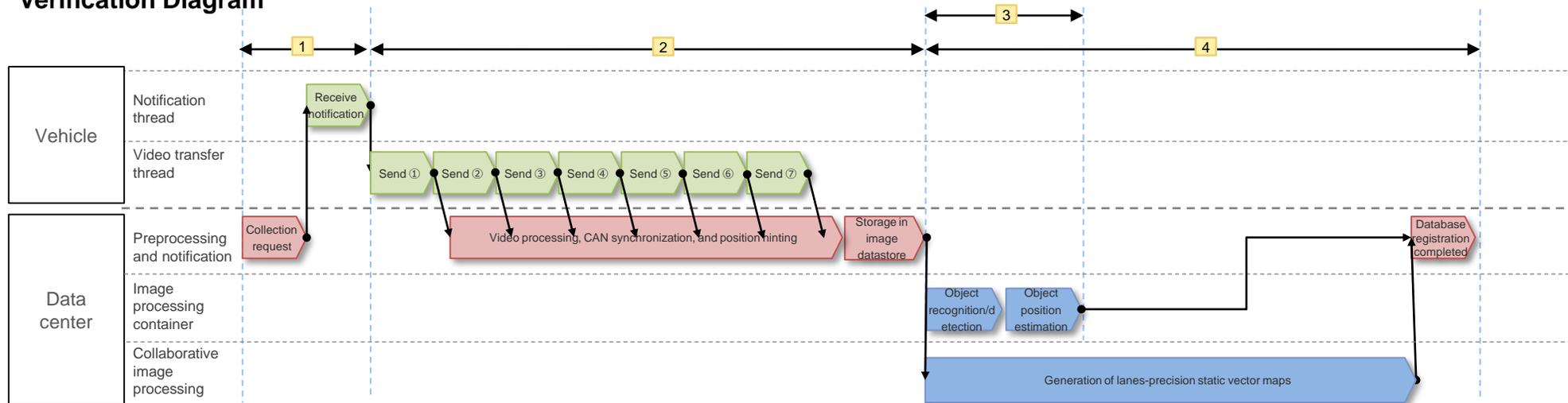
# Verification 2

To determine the time required for map generation, we measured and analyzed the time for each process, from the request for video collection to static map generation.

## Verification Description

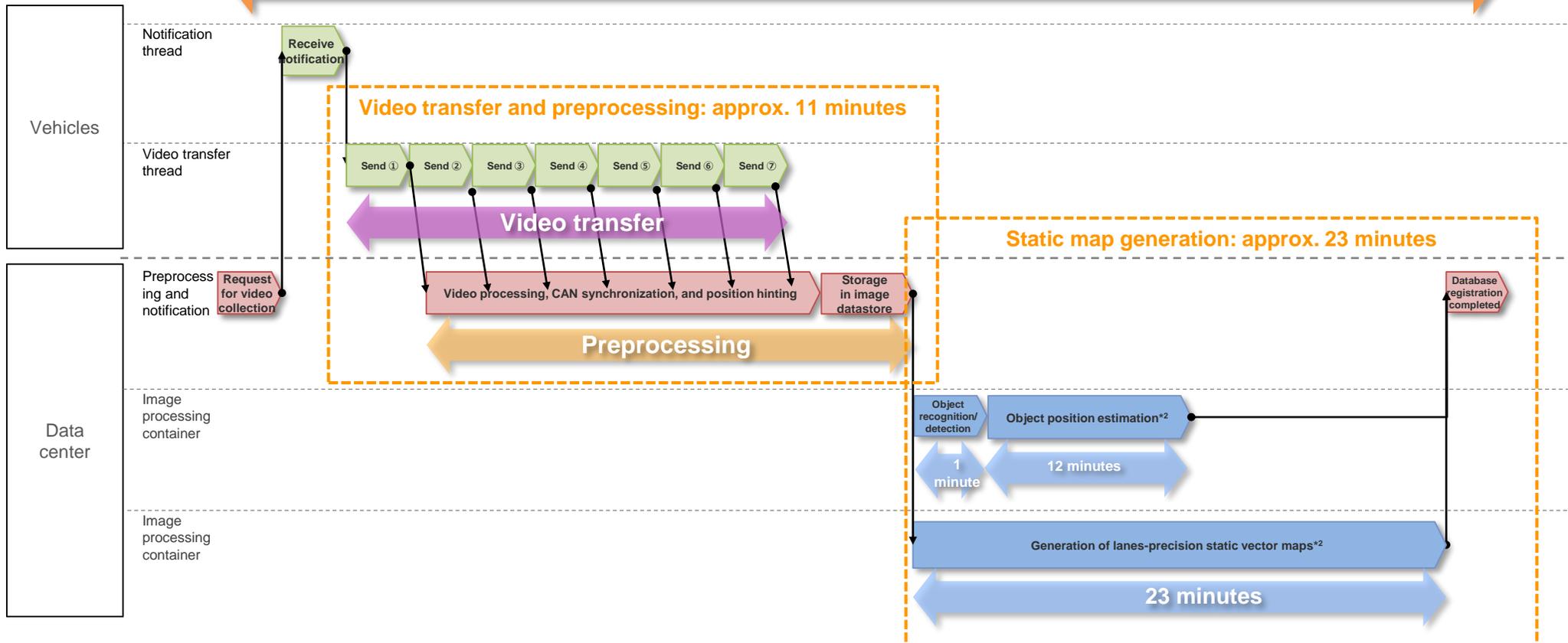
Category	No.	Item	Description
Request for video collection	1	Time taken to request video collection	Time elapsed between the time when a message requesting video collection is sent by the platform side and the start of the first video transmission
Static map generation and registration	2	Time taken from video transfer to preprocessing	The time elapsed between when all chunks of the video have been transferred and the completion of preprocessing (storing into image datastore)
	3	Time taken for object recognition/detection	The time taken for recognition/detection of an object to the delivery of all video chunks
	4	Time taken for lane map generation	The time taken for preprocessing (data shaping and other processes up to when the data is sent to the congestion detection container), including waiting for the delivery of data chunks

## Verification Diagram



# Verification 2: Results

In the field trial, the actual time taken from the request for video collection to static map generation for the trial run route\*1 was approx. 35 minutes\*2.



\*1 A single vehicle driving approximately 2 km. For more information, see Appendix: Trial Routes for Verification.

\*2 An additional 12 minutes or so are required when adopting serial processing of position estimation using Visual SLAM. Actual processing time is not affected in this verification, in which parallel processing for lane map generation was used.

# Summary

Verification Points	Summary	Future Issues
1. Functionality	<p>We confirmed process flow from video collection to static map generation using test cars. The processes from video collection to static map generation included:</p> <ul style="list-style-type: none"><li>• Delivery and notification</li><li>• Reception, enrichment, and annotation</li><li>• CAN data stream processing</li><li>• Image data stream processing</li><li>• Object recognition/detection</li><li>• Object position estimation</li><li>• Lane map generation</li><li>• Storage in the static map datastore</li></ul>	<p>While we were able to verify the basic operation of the functions, the following future issues must be resolved for commercialization.</p> <ul style="list-style-type: none"><li>• Precision measurement and enhancement for maps generated using only data obtained from the front-view camera (monocular camera) of vehicles and GPS data</li><li>• Realization of processing for extracting and sending map data to vehicles based on their current location</li><li>• Realization of efficient data collection by road or by lane, as opposed to the current assumption of mesh-based data collection</li></ul>
2. Real-time execution	<p>Approx. 35 minutes overall were required from the request to send video to map generation, including time for interval operations that needed to be executed manually.</p> <ul style="list-style-type: none"><li>• Video transfer and preprocessing: approx. 11 minutes</li><li>• Static map generation: approx. 23 minutes</li></ul>	—

# Verification of Use Case for Obstacle Detection

# Verification Overview

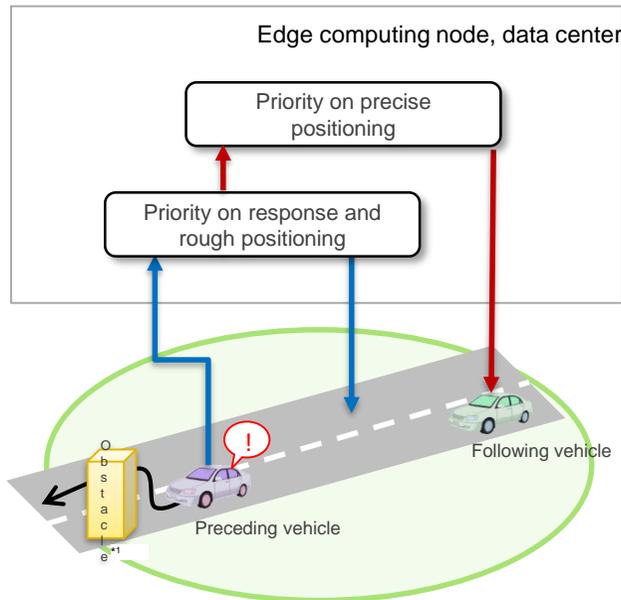
In this use case, video data of obstacles taken by the onboard camera were sent from the vehicles to the platform, and the platform provided notification to the vehicles after identifying the obstacles to identify issues in real-time stream processing.

## Overview

The platform identifies an obstacle from the video taken by an onboard-camera and sent by a vehicle and then notifies following vehicles about the obstacle. Notification takes place in two stages.

Stage 1: Priority on responsiveness

Stage 2: Priority on precise positioning



## Verification Points

1. Functionality  
Confirm the operation of processes from obstacle detection to notification.
2. Real-time execution  
Confirm that the following vehicle is notified within seven seconds of an obstacle being detected by a preceding vehicle to prioritize responsiveness.

\*1 Due to the difficulty of creating obstacles on a public road, traffic lights were used as hypothetical obstacles.

# Verification 1: Description and Results TOYOTA

As for functionality, we conducted verification tests for the series of processes from obstacle detection to notification using test cars. We verified the following items and confirmed the correct operation of all processes.

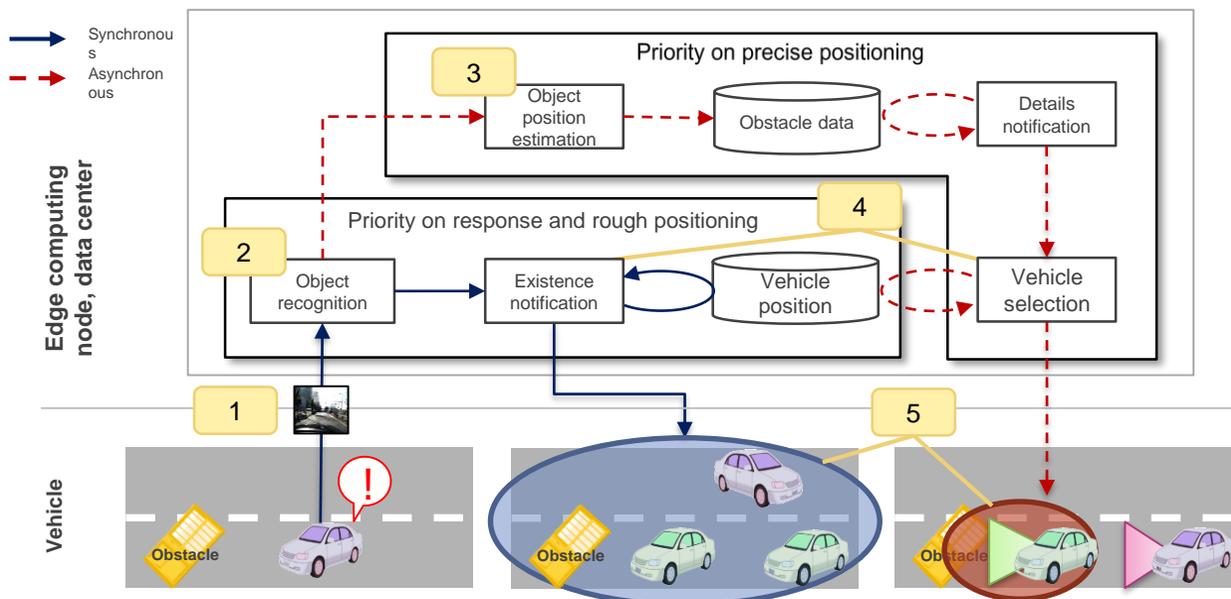
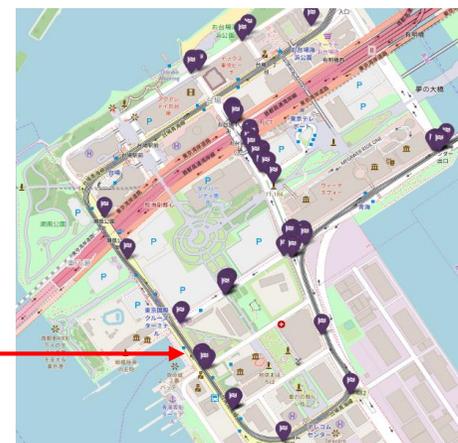
## Verification Points

Verification Points	
1	Receiving video data of obstacles sent from a vehicle (DCM emulation)
2	Identifying traffic lights based on object recognition
3	Identifying the position of traffic light based on object position estimation
4	Searching for vehicles to notify about the obstacle
5	Sending requests for video collection and output to the console

(2) Processing of video data for object recognition



(3) Traffic lights identified by object position estimation are shown



# Verification 2: Description

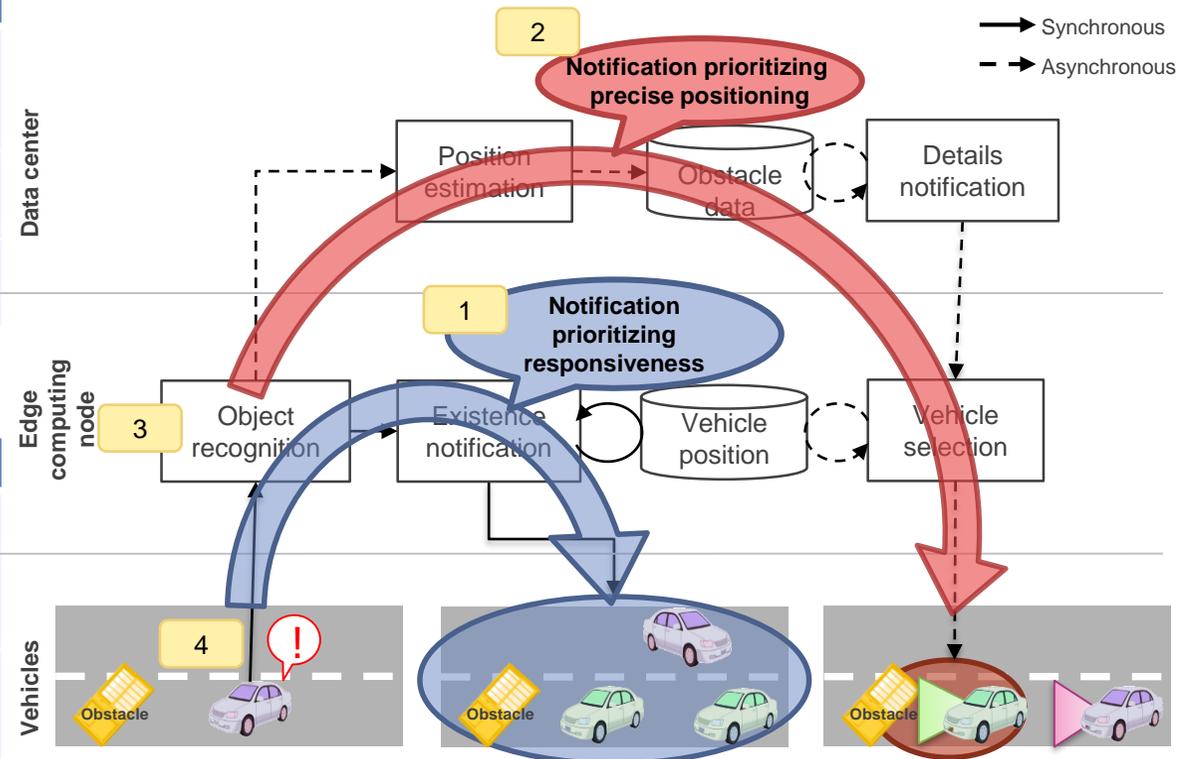
As for real-time execution, we measured and verified response time from the obstacle detection to notification.

## Verification Points

No.	Verification Points
	Measure time taken for the following stages of video data collection, image processing, and the notification process
1	Priority on responsiveness and rough positioning (from obstacle detection to existence notification)
2	Priority on precise positioning (from obstacle detection to details notification)

## Key Technical Elements

No.	Technical Elements
3	<b>Task offloading to edge computing node</b> Minimum required functions are offloaded to the edge computing node to ensure real-time responsiveness to accelerate notification*1
4	<b>Video chunking transfer</b> Sending video data in shorter chunks to shorten the waiting time for the platform side to begin processing, to accelerate overall processing*2



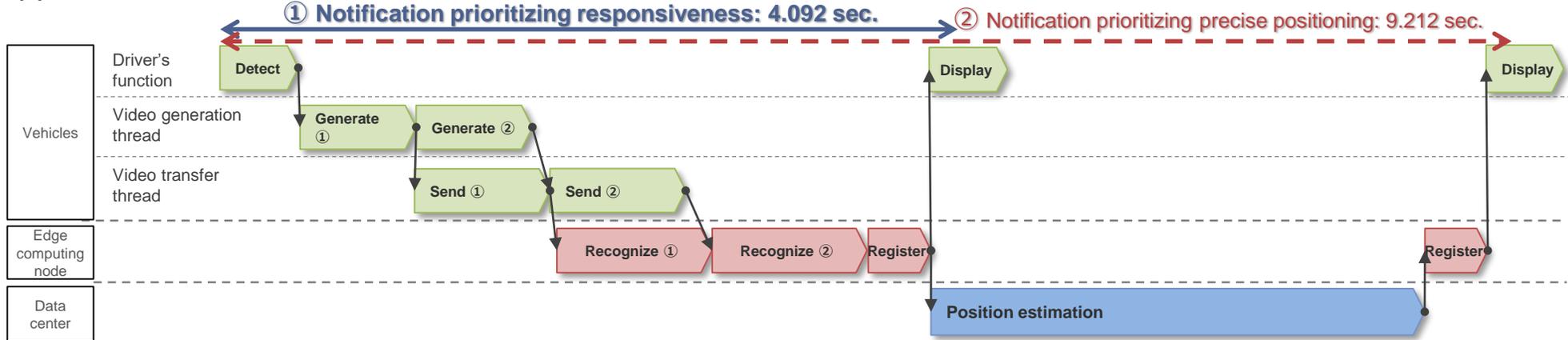
\*1 For details, see Appendix 4: Vertical-distributed computing technology

\*2 For details, see Appendix 7: Video chunking transfer technology

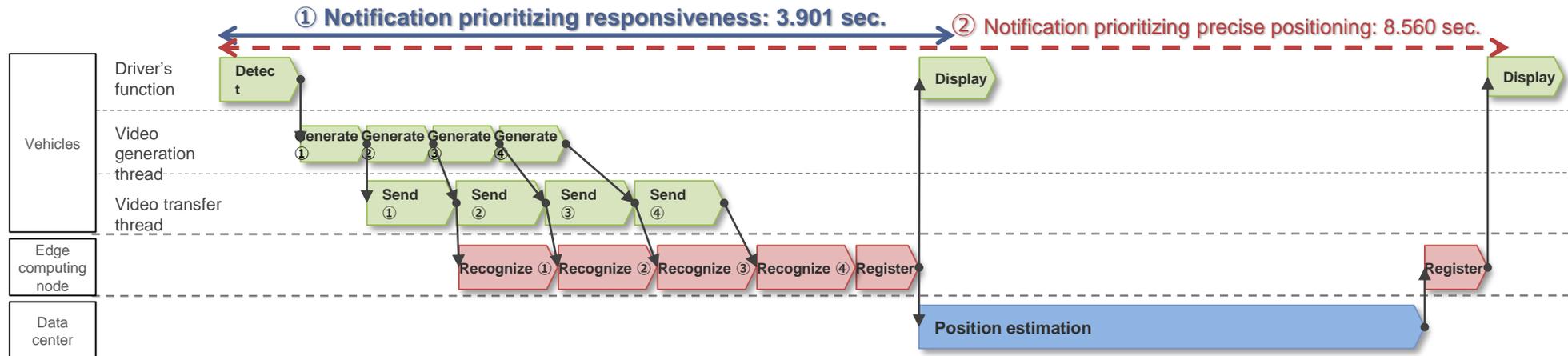
# Verification 2: Results

We conducted two patterns of chunked video transfers to measure processing times from obstacle detection to notification. The verification results are shown below. For notifications prioritizing responsiveness, we achieved our target of notifying vehicles within seven seconds\* of detecting the obstacle.

## (1) 1 second × 2 chunks



## (2) 0.5 seconds\* × 4 chunks



\*1 While this document lists the fastest value from the results of several tests conducted, all tests achieved the target of notification within seven seconds.

# Summary

Verification Points	Summary	Consideration and Future Issues
1. Functionality	<p>We confirmed process flow from obstacle detection to notification of nearby vehicles using test cars. The results of the verification for the two-stage notification process are as follows.</p> <ol style="list-style-type: none"> <li>1. Priority on responsiveness               <ul style="list-style-type: none"> <li>- Identification of <b>obstacles</b> by object recognition processing</li> <li>- Search of nearby vehicles based on <b>vehicle position</b> and notification</li> </ul> </li> <li>2. Priority on precise positioning               <ul style="list-style-type: none"> <li>- Identification of <b>obstacle position</b> by object-position-estimation processing</li> <li>- Search of nearby vehicles based on <b>obstacle position</b> and notification</li> </ul> </li> </ol>	<p>While we were able to verify the basic operation of the functions, the following future issues must be resolved for commercialization.</p> <ul style="list-style-type: none"> <li>• Evaluation of precision of object recognition and position estimation           <ul style="list-style-type: none"> <li>- Deviation from the actual object (position, quantity)</li> <li>- Degree of deterioration in precision due to weather and time zones</li> </ul> </li> <li>• Need to limit scope of data collection under current assumptions for obstacle monitoring Collecting video data from all vehicles receiving notification will result in high transfer and processing costs, so there is a need to limit the scope of data collection. For example, notification targets can be limited to vehicles approaching the obstacle by combining vehicle identification within the polygon range of high-performance spatial-temporal data management technology*<sup>1</sup> with a selective data collection algorithm*<sup>2</sup>.</li> </ul>
2. Real-time execution	<p>We were able to achieve the target of notification within seven seconds of obstacle detection by combining the following technologies.</p> <ol style="list-style-type: none"> <li>1. Task offloading to the edge computing node</li> <li>2. Video chunking transfer</li> </ol> <p>Although not included in this document, we also constructed an architecture that effectively uses edge computing by applying the following technologies.</p> <ol style="list-style-type: none"> <li>3. High-performance spatial-temporal data processing*<sup>1</sup></li> <li>4. Dynamic node selection*<sup>3</sup></li> </ol>	<ul style="list-style-type: none"> <li>• Functions can be offloaded to vehicles to accelerate processing and reduce data traffic</li> <li>• Load distribution of edge functions and optimal configuration of applications           <ul style="list-style-type: none"> <li>- Offloading of partial processes in response to load status</li> <li>- Data collaboration between edge computing nodes</li> </ul> </li> </ul>

\*1 For details, see Appendix 1: High-performance spatial-temporal data management technology and high-performance spatial-temporal query technology

\*2 For details, see Appendix 2: Data acquisition technology with target vehicle selection algorithm

\*3 For details, see Appendix 4: Vertical-distributed computing technology

# Verification of Use Case for Congestion Detection

# Verification Overview

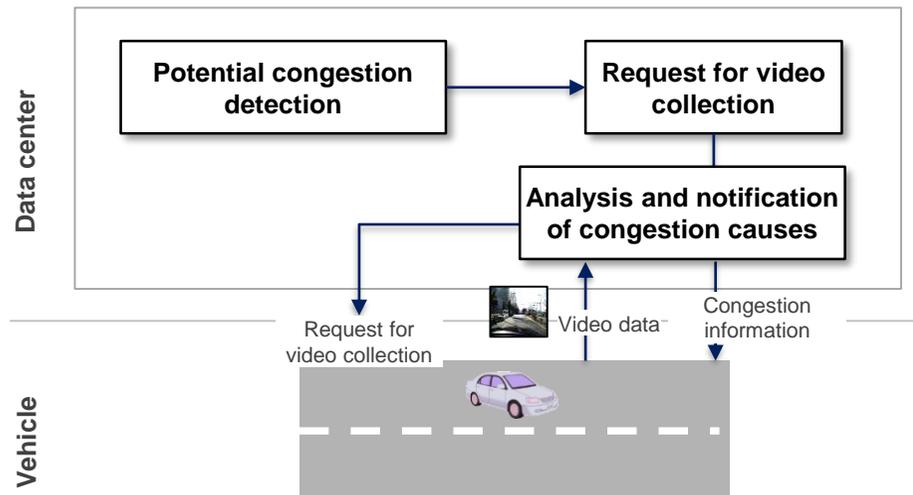
In this use case, we identified the technical issues involved in real-time execution of stream processing by running processes from the collection of video captured by the vehicles' onboard cameras to notification of analysis results, under actual field conditions.

## Overview

Platform identifies congestion from data captured by onboard cameras and sent by vehicles and then notifies following vehicles. The verification was executed in two steps.

Step 1. Potential congestion detection up to the notification and request for video collection

Step 2. Request for sending video up to the analysis and notification of congestion causes



## Verification Points

1. Functionality  
Confirm process flow from detection of congestion to notification.
2. Response time  
Confirm that the process of requesting videos to be sent to analyze congestion causes and notify vehicles is executed within five minutes.

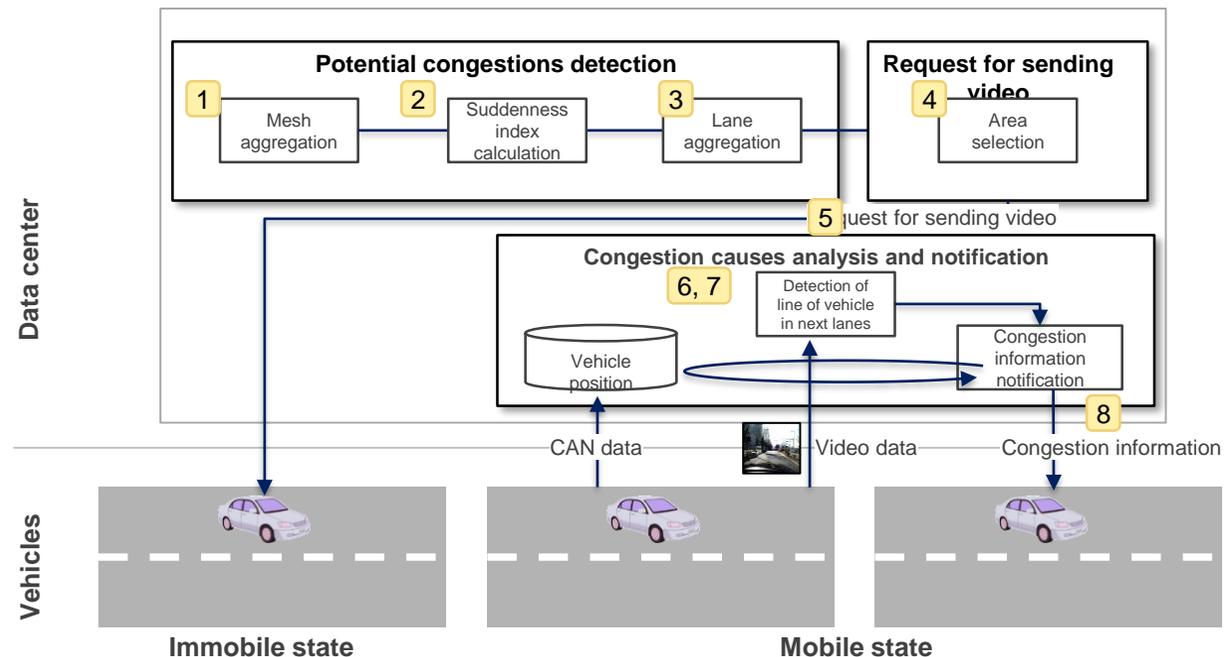
# Verification 1: Description and Results **TOYOTA**

Regarding functionality, we conducted verification for the series of processes (1 to 8) from detection of congestion to notification, and we confirmed that all processes operated as required.

## Verification Targets

Category	No.	Function
Potential congestions detection	1	Mesh aggregation
	2	Suddenness index calculation*1
	3	Lane aggregation
Request for sending video	4	Area selection
	5	Request for video collection
Analysis and notification of congestion cause	6	Detection of line of vehicles in next lanes
	7	Congestion cause analysis
	8	Notification of congestion

## Verification Diagram



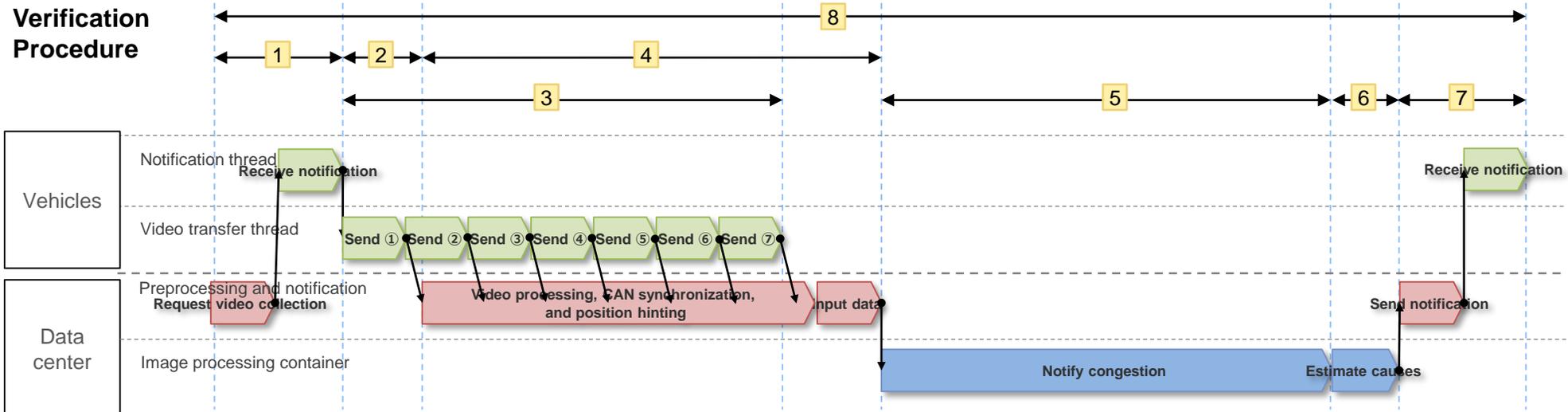
\*1 For more details, see Appendix 5: Suddenness Index Calculation Method

# Verification 2

We measured and analyzed the time taken for each process involved in requesting video, analyzing congestion causes, and notifying vehicles.

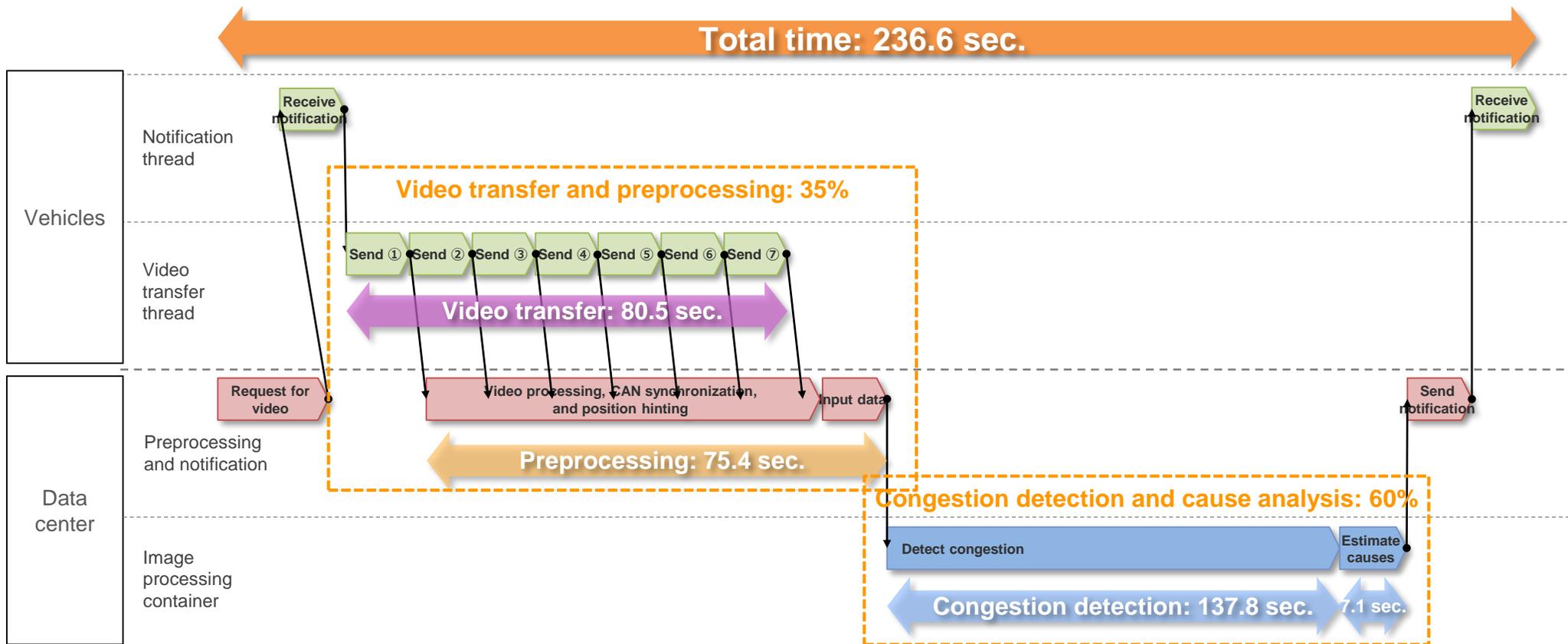
## Verification Points

Category	No.	Item	Description
Requesting video	1	Time taken to request video	Time from the request for sending video issued by the platform side to the start of first video transfer
Analysis and notification on congestion cause	2	Waiting time for initial chunk transfer	Time from the start of the first video transfer to the start of processing the chunk on the platform
	3	Time taken for video transfer	Time for sending all video chunks
	4	Time taken for preprocessing	Preprocessing time (data shaping and other processes up to when the data is sent to the congestion detection container), including waiting time for the delivery of data chunks
	5	Time taken for congestion detection	Processing time to detect congestion
	6	Time taken for cause estimation	Processing to estimate congestion cause
	7	Time taken for notification	Time from the start of notification processing by the platform side to the reception of notification by the vehicles
	8	Total time	Time required for the entire process



# Verification 2: Results

We measured and analyzed the processing time of each stage for the case\*1 in which we achieved the fastest time under actual field conditions. We found that congestion detection accounted for roughly 60% of total processing time, while video transfer and preprocessing accounted for about 35% of the total.



\*1 Overview of the trial run results under actual field conditions: successful processing: 6 trials; fastest time: 236.6 seconds; slowest time: 311.4 seconds; average time: 264.7 seconds

# Summary

Verification Points	Summary and Consideration	Future Issues
1. Functionality	<ul style="list-style-type: none"> <li>For the suddenness index calculation method, a desk study confirmed that target lanes for aggregation can be reduced by up to 80%</li> <li>We used image data captured by real vehicles to verify operations for this function and confirmed that it can recognize lines of vehicles occurring in the real world</li> <li>We were able to verify the entire process, from collecting video data from appropriate vehicles and delivering congestion information to appropriate vehicles, in collaboration with the existing logic for vehicle selection and notification</li> </ul>	<ul style="list-style-type: none"> <li>Use of machine learning to upgrade the suddenness index table</li> <li>Refinement of detecting congestion in a vehicle's own lane, adjacent lanes, and opposite lanes in area selection, and selection of optimal lanes</li> <li>Consideration of selection logic for an optimal vehicle in video data collection</li> <li>Response to unavailability and insufficient resolution of CAN data</li> <li>Assessment of significance between search results for a large amount of video data</li> <li>Integration with edge computing</li> <li>Integration with dynamic node selection technology</li> </ul>
2. Real-time execution	<ul style="list-style-type: none"> <li>We constructed the system within the framework of Spark<sup>*1</sup> and Kubernetes<sup>*1</sup> to enable parallel distributed computing in the next phase of the process and confirmed that the system was operating properly through collaboration between technologies</li> <li>We confirmed that it takes four to five minutes to execute congestion detection and cause analysis for 70-second videos (video collected in response to requests for 60-second videos)</li> <li>We analyzed the processing time and identified several areas of improvement; in particular, the process can be shortened by about one minute by improving the interface to reduce waiting time for processing on the platform side</li> </ul>	<ul style="list-style-type: none"> <li>Enhancement of throughput and turnaround time (TAT) based on parallel computing for the suddenness index calculation</li> <li>Enhancement of throughput and TAT based on improved interface for the suddenness index calculation</li> <li>Outsourcing of the suddenness index table calculation to raise scalability</li> <li>Enhancement of throughput and TAT by eliminating file entries from the preprocessing phase for recognizing congestion in adjacent lanes</li> <li>Enhancement of throughput and TAT by integrating the preprocessing phase for recognizing congestion in adjacent lanes with functions</li> <li>Enhancement of throughput and TAT by avoiding redundant video processing from the preprocessing phase for recognizing congestion in adjacent lanes</li> </ul>

\*1 For details on architecture using Spark/Kubernetes, see Appendix Architecture (Detailed Version)

# Evaluation of Platform

# Evaluation Overview

We conducted an evaluation focused on five components of the architecture for the ICT platform for connected vehicles\*1.

No.	Evaluated Components	Overview
1	Reception, enrichment and annotation, queue	Conduct a load test on the collection platform for CAN data and image data sent from vehicles under the load of a simulated five million vehicles sending data at intervals of ten seconds or less.
2	CAN data processing	Execute stream processing of CAN data sent from vehicles under the load of a simulated five million vehicles to evaluate possibility of real-time storage to the dynamic database. Position data stored in the database are aggregated based on micro-batch processing to ascertain performance trends.
3	Image data processing	Measure the maximum capacity of the image processing platform when executing stream processing of video data sent from vehicles and batch-processing of massive amounts of image data accumulated in the image datastore.
4	Delivery and notification	Evaluate performance of search and selection processes for vehicles targeted in the video request and data transfer phases.
5	Network edge computing	Evaluate the functionality and performance of the platform system featuring several network functions at edge sites, (called 'Network Edge'), to determine feasibility of optimizing data flow.

\*1 For details on system configuration, see Appendix: Platform Verification System

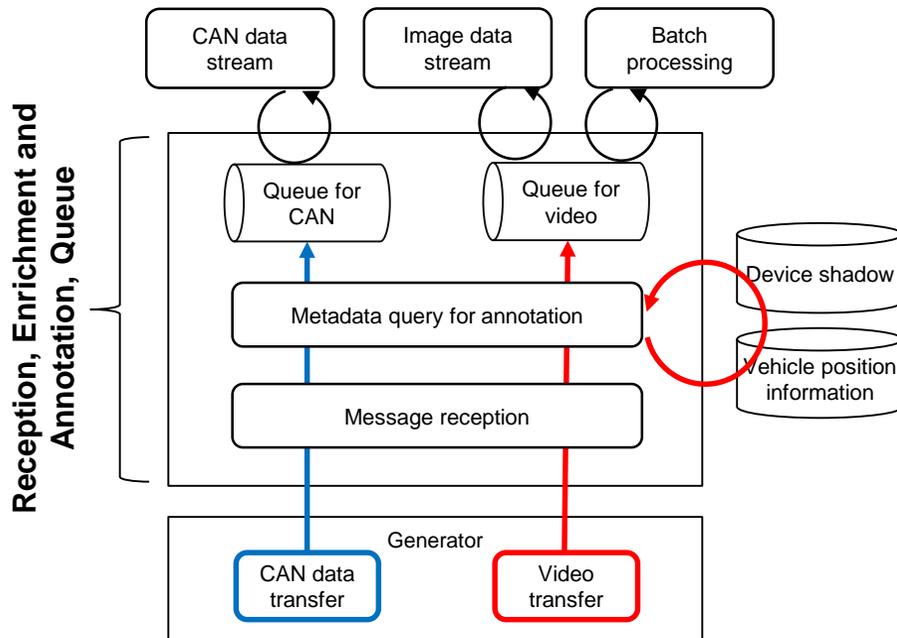
# Evaluation of Annotation Platform

# Evaluation Overview

Conduct a load test on the annotation platform under the load of a simulated five million vehicles sending data at intervals of ten seconds or less.

## Overview

Massive CAN data and image data sent from vehicles must be processed in real-time. We evaluated the performance of the annotation platform, which serves as the preprocessing phase for utilizing data in stream processing.



## Evaluation Points

Performance testing of the annotation platform by data type

1. Stream processing (preprocessing) of CAN data  
Confirmed potential for simultaneously handling 600,000 driving vehicles (5 million × 12% of peak operational capacity) and performance of each component under the various loads.
2. Stream processing of image data (preprocessing)
  - a. Conduct a load test and measure the processing time of dynamic prioritization based on vehicle position and speed information as well as simultaneous processing within the area.
  - b. Conduct a load of the queue server and confirm resource bottlenecks.

# Evaluation 1

We examined the potential for simultaneously handling a quantitative target of 600,000 driving vehicles (5 million × 12% of peak operational capacity) and the performance trends for each component.

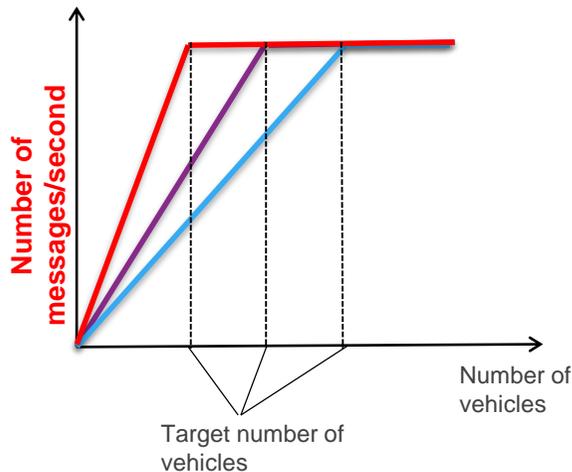
## Description

We tested performance trends for CAN data processing by dividing the process into two components.

- Annotation server performance trends
- Queue server performance trends

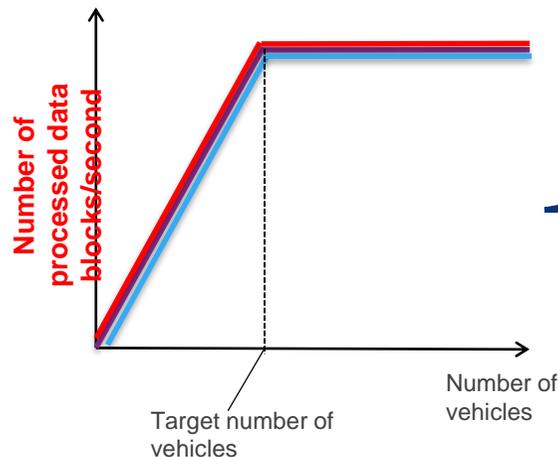
### Annotation Server Performance Trends (Hypothesis)

Shorter send intervals will result in increasing the number of messages, thereby reducing the maximum number of vehicles.



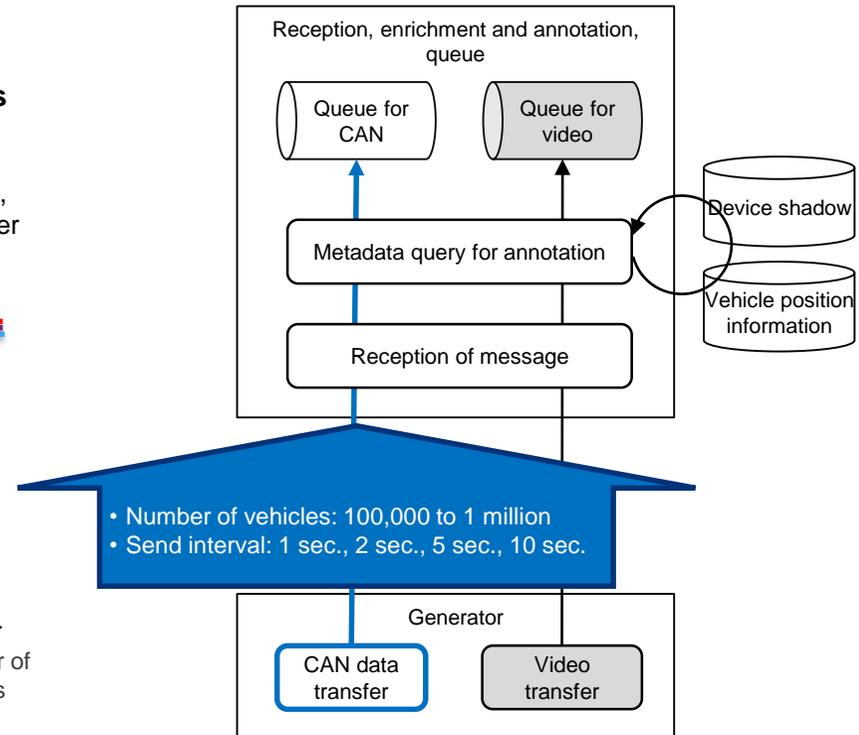
### Queue Server Performance Trends (Hypothesis)

Shorter send intervals will not affect the overall number of processed data blocks, thereby maintaining the maximum number of vehicles.



## Conditions

Measure the maximum limit for the number of vehicles for four separate cases of send intervals ranging from one to ten seconds.



# Evaluation 1: Results

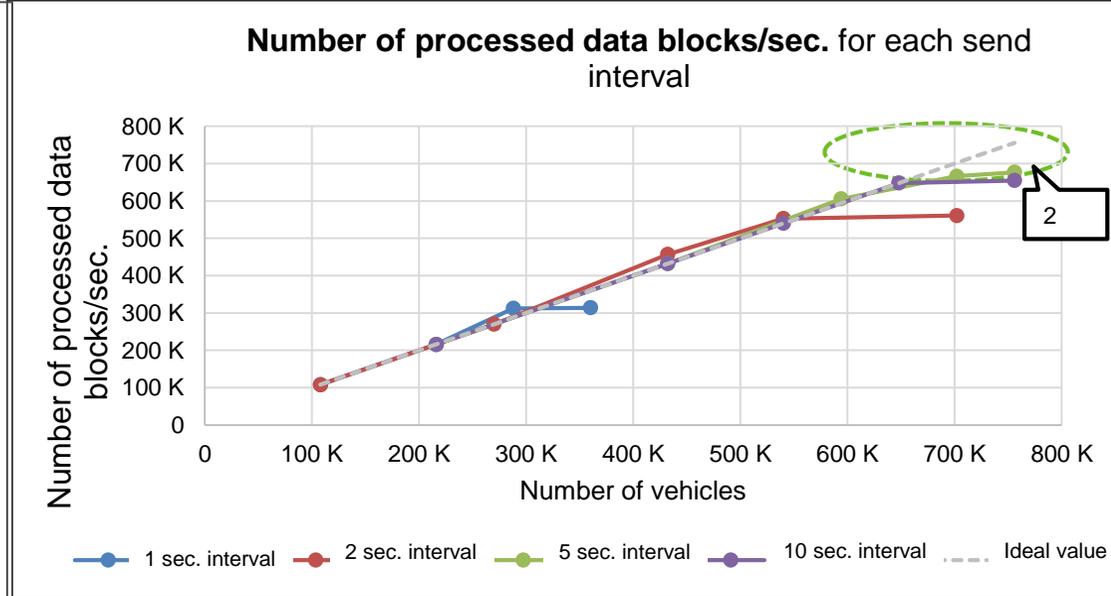
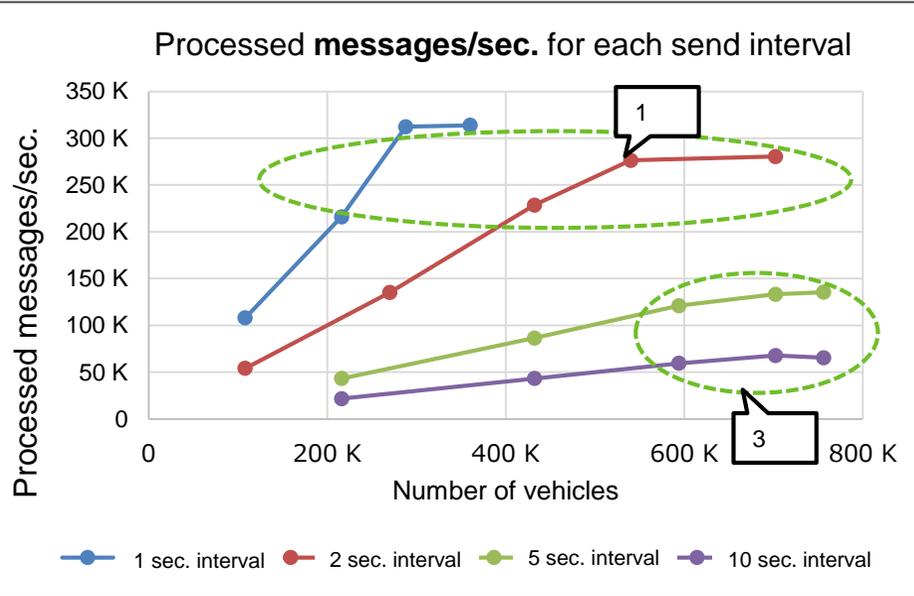
We measured the maximum number of vehicles for each send interval and found that our performance target of 600,000 vehicles can be attained when messages were sent at an interval of at least five seconds.

Message send interval (seconds)	Maximum number of vehicles	Results	Bottlenecks
1	306,000	Unattained	Annotation server
2	540,000	Unattained	Annotation server
5	648,000	Attained	Queue server
10	648,000	Attained	Queue server

1. Marginal performance: 300,000 messages/sec.
2. Marginal performance: 640,000 data blocks/sec.
3. Annotation server is not the bottleneck

## Annotation Server Performance Trends

## Queue Server Marginal Performance



# Evaluation 2-a

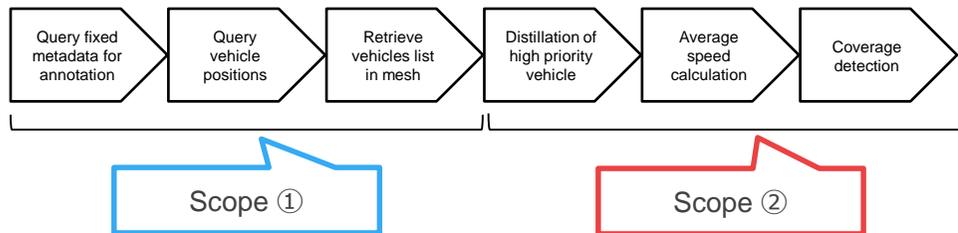
We measured the critical load and processing time of dynamic prioritization based on vehicle position and speed information as well as simultaneous processing within the area.

## Description

In dynamic prioritization processing, vehicles are prioritized according to statistical data aggregated from position information. Performance is expected to be affected by the number of vehicles located within the scope of statistic data (inside the spatial-temporal mesh and the circular area surrounding an event), so verification tests are conducted by altering the number of vehicles under the following conditions.

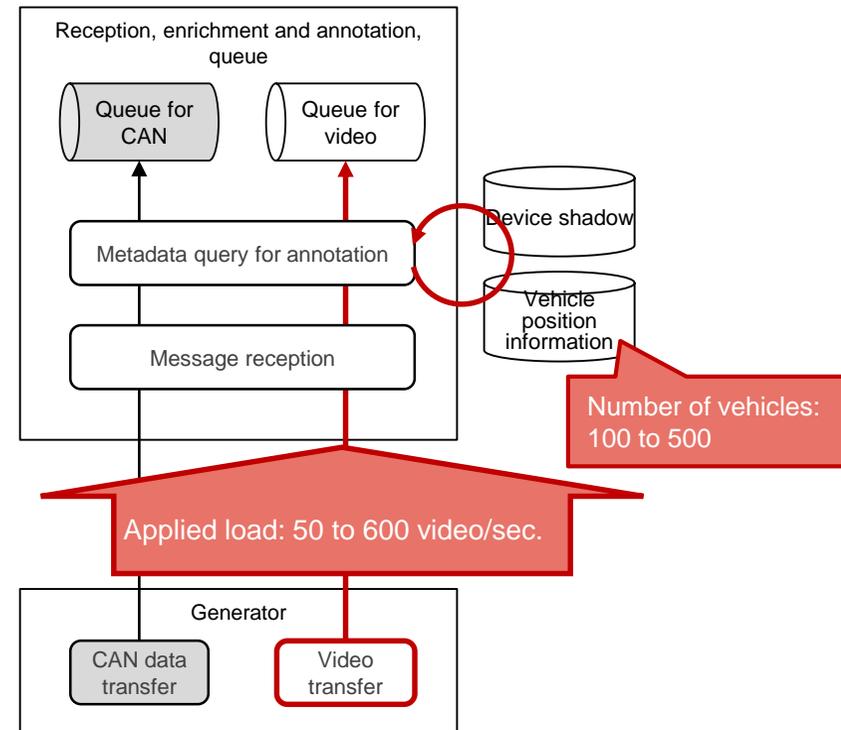
(1) Increase in the number of vehicles in each mesh  
Measure the performance involved in the process of retrieving the vehicles list by **increasing the number of vehicles in each mesh**.

(2) Increase in the number of vehicles inside the circular area  
Measure the performance involved in average speed calculation and coverage detection for the retrieved vehicles list by **increasing the number of vehicles inside the circular area while fixing the number of vehicles in each mesh**.



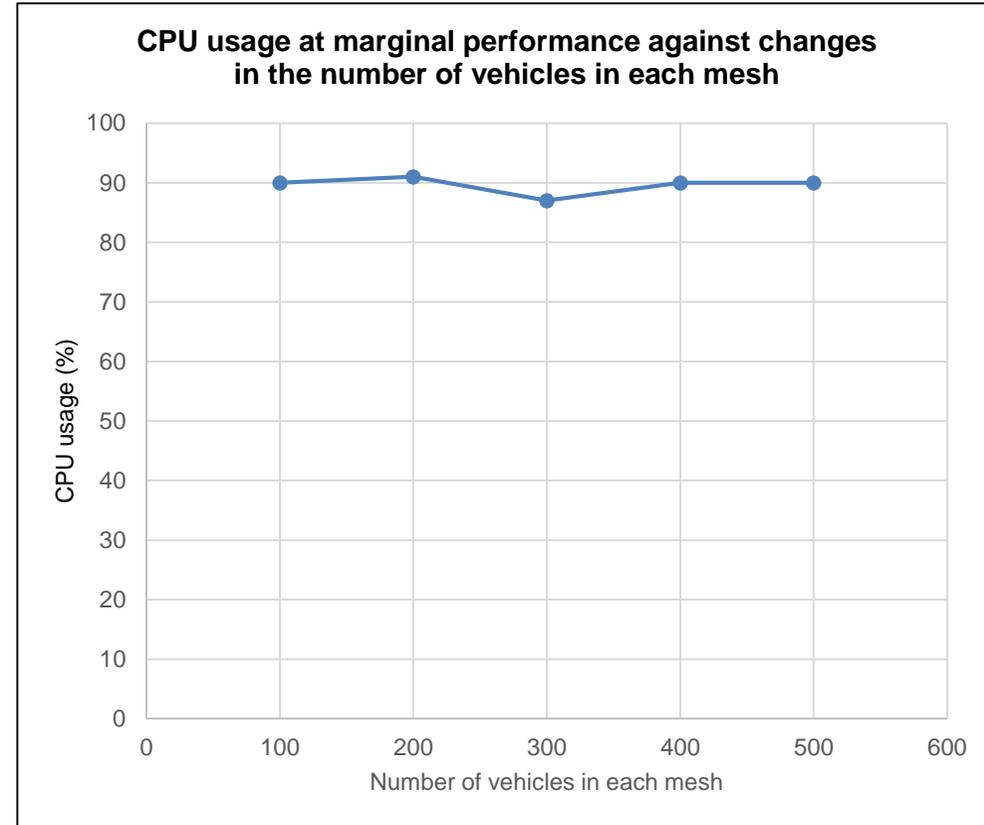
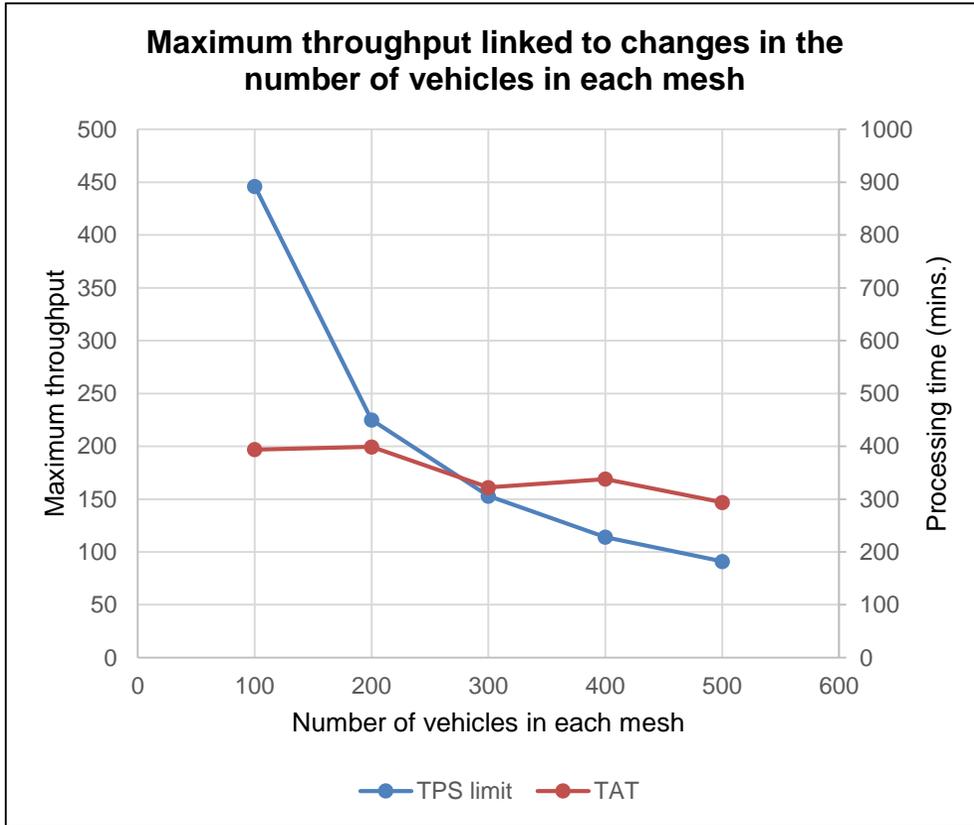
## Conditions

The processing limit of vehicles with regard to load per second was measured after designating vehicle position information.



# Evaluation 2-a: Results (1/2)

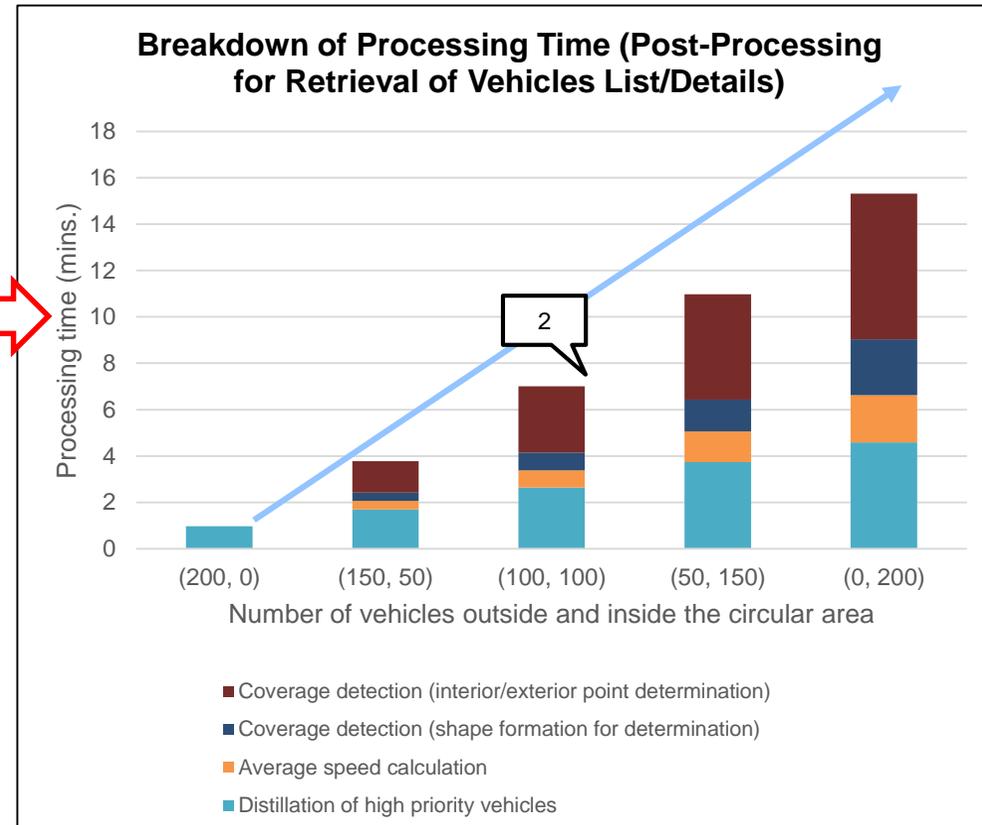
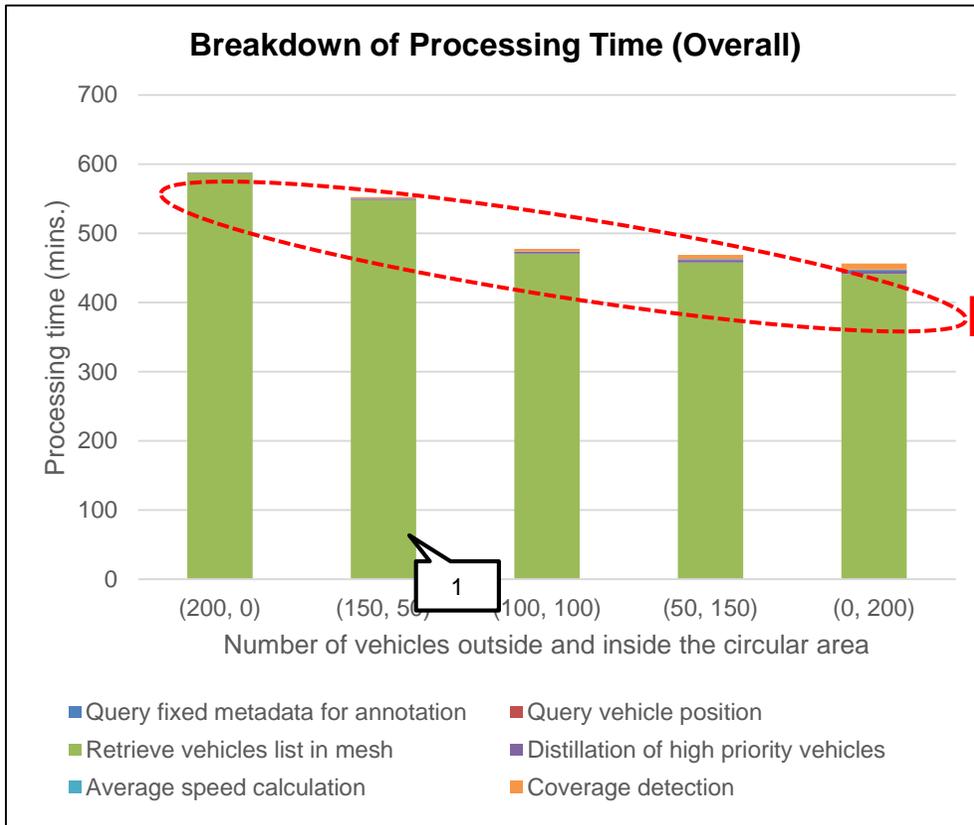
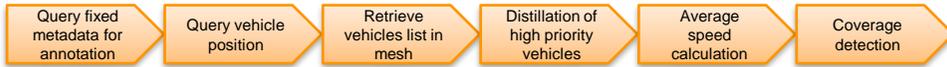
By measuring the performance, we found that the maximum throughput declines as the number of vehicles in each mesh rises. The bottleneck was the CPU of the annotation server.



# Evaluation 2-a: Results (2/2)

As for the processing time, we observed the following two characteristics.

1. Processing for retrieving the vehicles list from the dynamic database accounts for over 95% of overall processing time.
2. The processing time of the average speed calculation and coverage detection for vehicles inside a circular area increases linearly with respect to the number of vehicles inside the area.



1. Nearly all are retrieval processes for vehicles list (impact of the number of vehicles is within margin of error)

2. Linear increase

# Evaluation 2-b

Conduct a load test of the queue server and identify resource bottlenecks.

## Description

The marginal performance value of the queue server was ascertained by sending video data on a regular basis. Dynamic prioritization processing was not performed to position the queue server as the bottleneck.

### Profile of video data

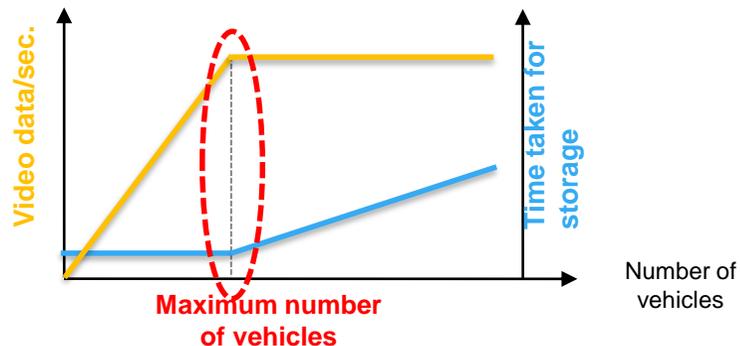
Length of video: 2 seconds (approx. 2 MB)

Frame rate: 10 fps

Bit rate: standard (6,200 kbps)

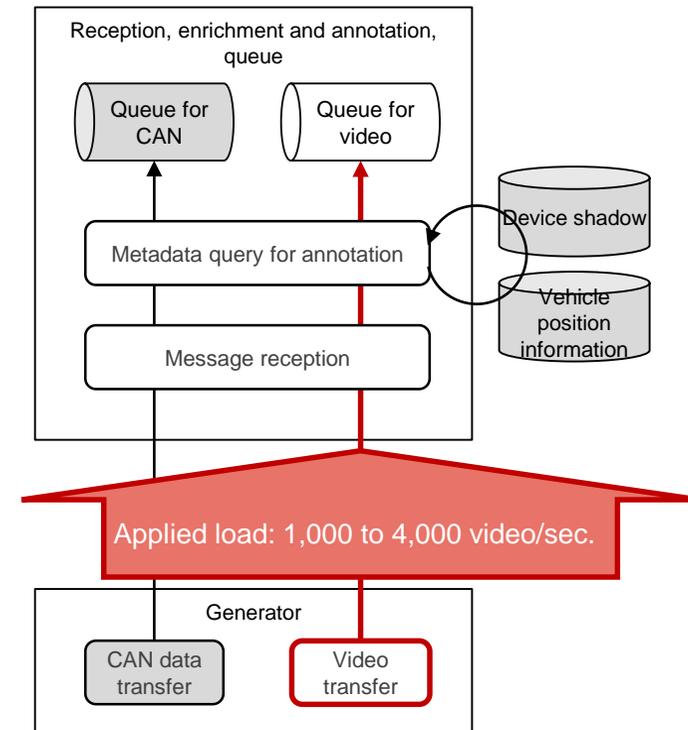
### Key area for ascertaining the critical load of the queue server

Critical load value was determined by measuring processable load per unit time and time taken to store data in the queue.



## Conditions

The maximum number of vehicles processable for load per second was measured.

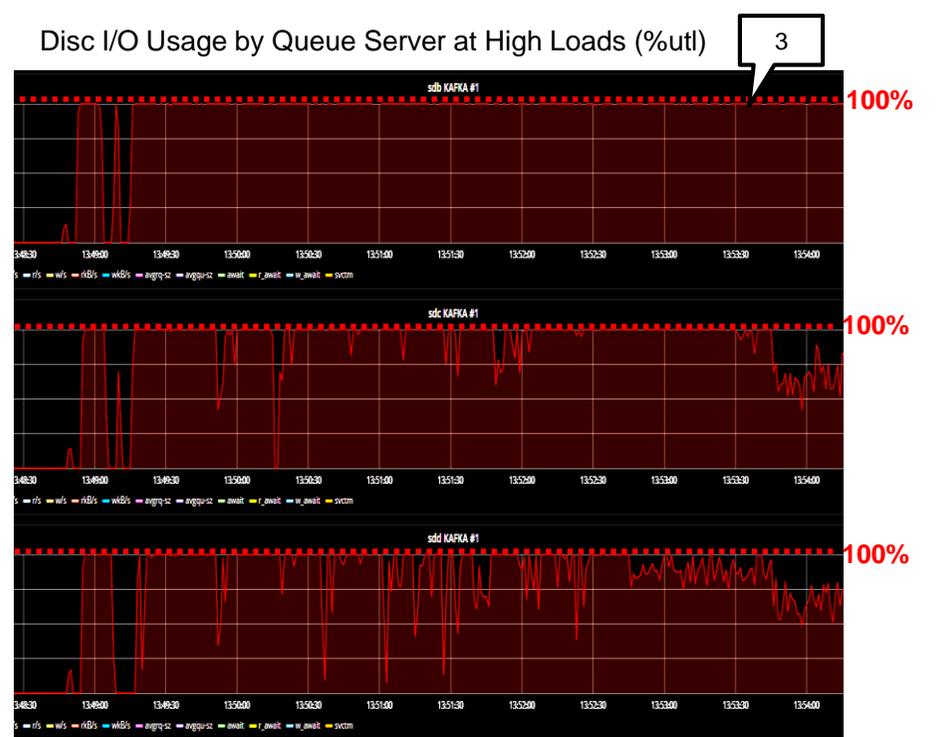
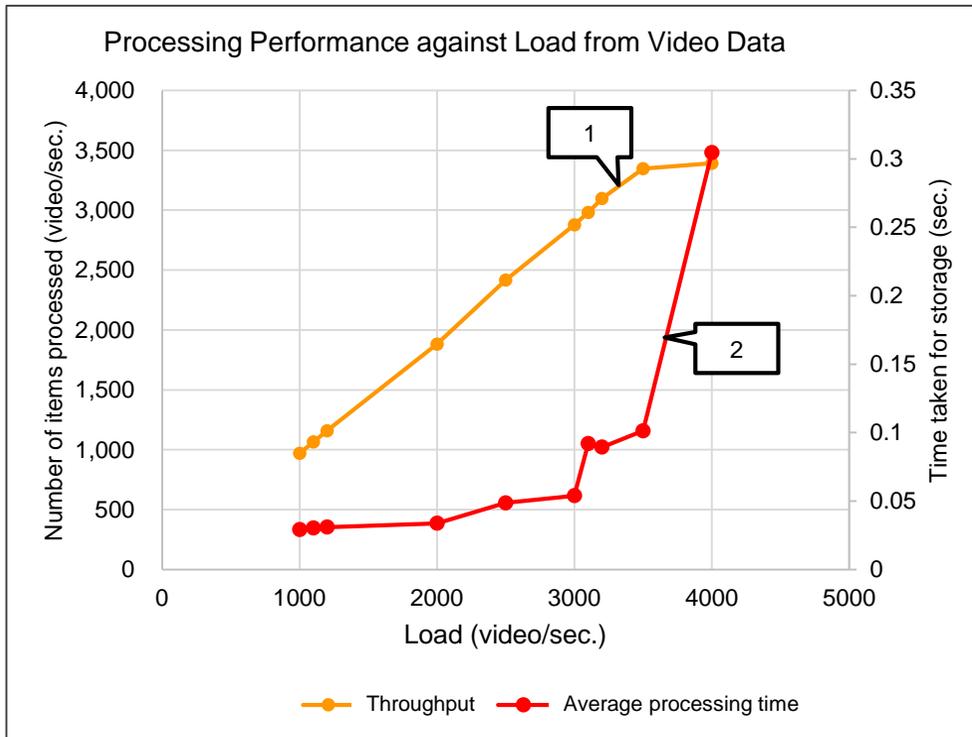


# Evaluation 1-b: Results

We measured the marginal processing performance of the queue server with respect to load applied by video data. We found that the server was capable of continuous processing up to loads of around 3,500 video/sec., but performance peaks out above this threshold.

Load (video/sec.)	Preprocessing time (ms)	Waiting time for sending to queue (ms)	Time taken to complete storage to queue (ms)
2000	0.918	2.121	30.336
3000	1.312	5.700	48.175
4000	2.653	<b>54.743</b>	<b>213.420</b>

- 1: Peak-out at 3,500 (video/sec.)
- 2: Rising load leads to longer processing time
- 3: Bottleneck: disc input/output



Evaluation Points	Summary
1. Stream processing of CAN data (preprocessing)	<p>Performance target of 600,000 vehicles was achieved when messages were sent at intervals of five and ten seconds. The performance trend hypothesis confirmed that the bottleneck changes with send intervals.</p> <ul style="list-style-type: none"><li>• Message send interval: one second, two seconds<ul style="list-style-type: none"><li>– Bottleneck: <b>Annotation server</b></li><li>– Reason: More messages are sent at shorter intervals</li></ul></li><li>• Message send interval: five seconds, ten seconds<ul style="list-style-type: none"><li>– Bottleneck: <b>Queue server</b></li><li>– Reason: Total volume of data processed is unaffected by changes in send intervals</li></ul></li></ul>
2. Stream processing of image data (preprocessing)	<p>a. Dynamic prioritization processing Process of retrieving the vehicles list from the dynamic database accounts for over 95% of the total processing time. As expected, the processing time of average speed calculation and coverage detection for vehicles inside a circular area increases linearly with respect to the number of vehicles inside the area.</p> <p>b. Storage processing to queue server Server was confirmed to be capable of continuous processing up to loads of around 3,500 video/sec. The bottleneck was confirmed to be <b>queue server</b> disc I/O.</p> <p>Common considerations: The current use case assumes that video data is only sent when an event occurs, so it can be assumed that the marginal performance of the overall system does not affect trial results since it applies to image processing in a later phase. An effective method for data collection must be considered, however, for cases involving the regular collection of video data.</p>

# CAN Data Processing

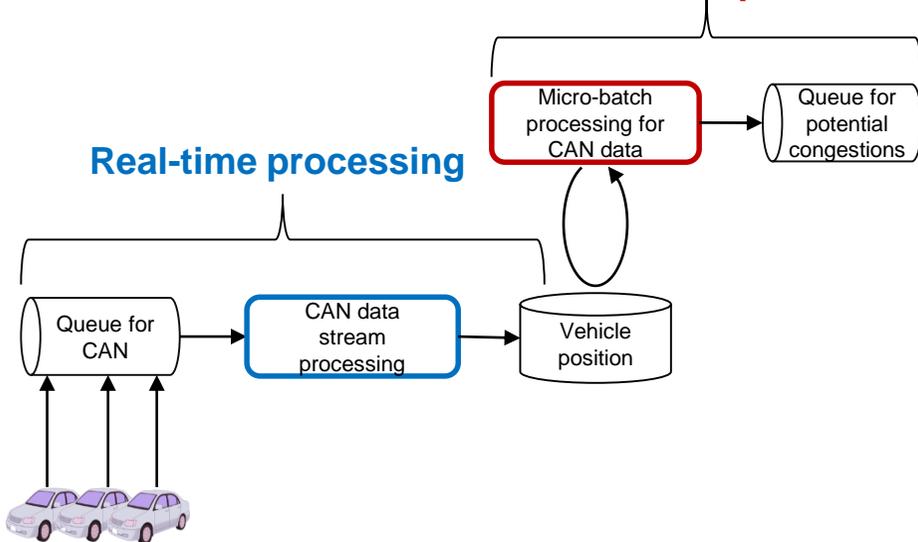
# Evaluation Overview

The CAN data processing platform was verified by measuring the performance of each processing method (stream processing, micro-batch processing).

## Overview

Vehicles collect massive volumes of CAN data, and in some cases the response takes priority over throughput. Therefore, the processing method must be determined according to performance requirements. This verification involved measuring the performance of stream processing and micro-batch processing.

### High throughput and Semi-real-time processing



## Evaluation Points

1. CAN data stream processing
  - a. The marginal performance for a guaranteed response was confirmed by using the process of raw data conversion to SI unit values for regular CAN data as an example.
  - b. Possibility of achieving the target of 600,000 vehicles (5 million  $\times$  12% of peak operational capacity) during the storage processing of vehicle data was examined.
2. Micro-batch processing for CAN data  
The effectiveness of detection during search processing was confirmed by using potential congestions detection by mesh and lane aggregation as an example.

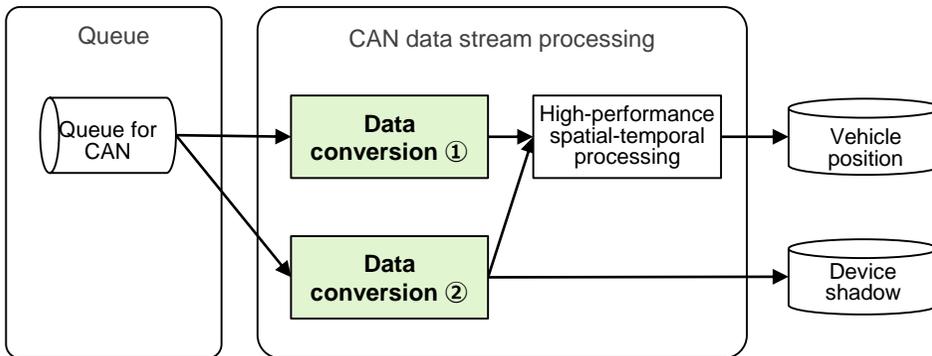
# Evaluation 1-a

Confirm the marginal performance for guaranteed response using the process of raw data conversion to SI unit values for regular CAN data as an example.

## Description

We will realize a high-performance search by storing the latest vehicle position data in the device shadow. On the other hand, this requires an additional process of storing data in the device shadow during raw data conversion to SI unit values. Therefore, we will also verify the level of performance deterioration in the overall conversion process. Marginal performance is defined as the maximum number of vehicles for which raw data conversion to SI unit values is completed within the span of fixed processing intervals and compared performance for the following two patterns.

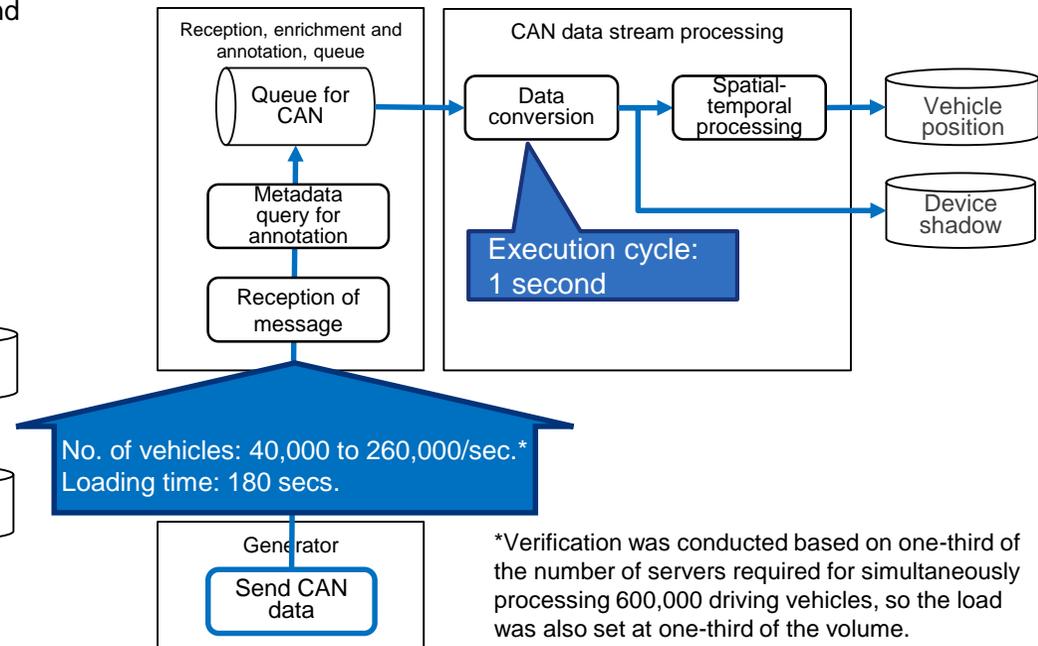
- ① Without entries into the device shadow
- ② With entries into the device shadow



## Conditions

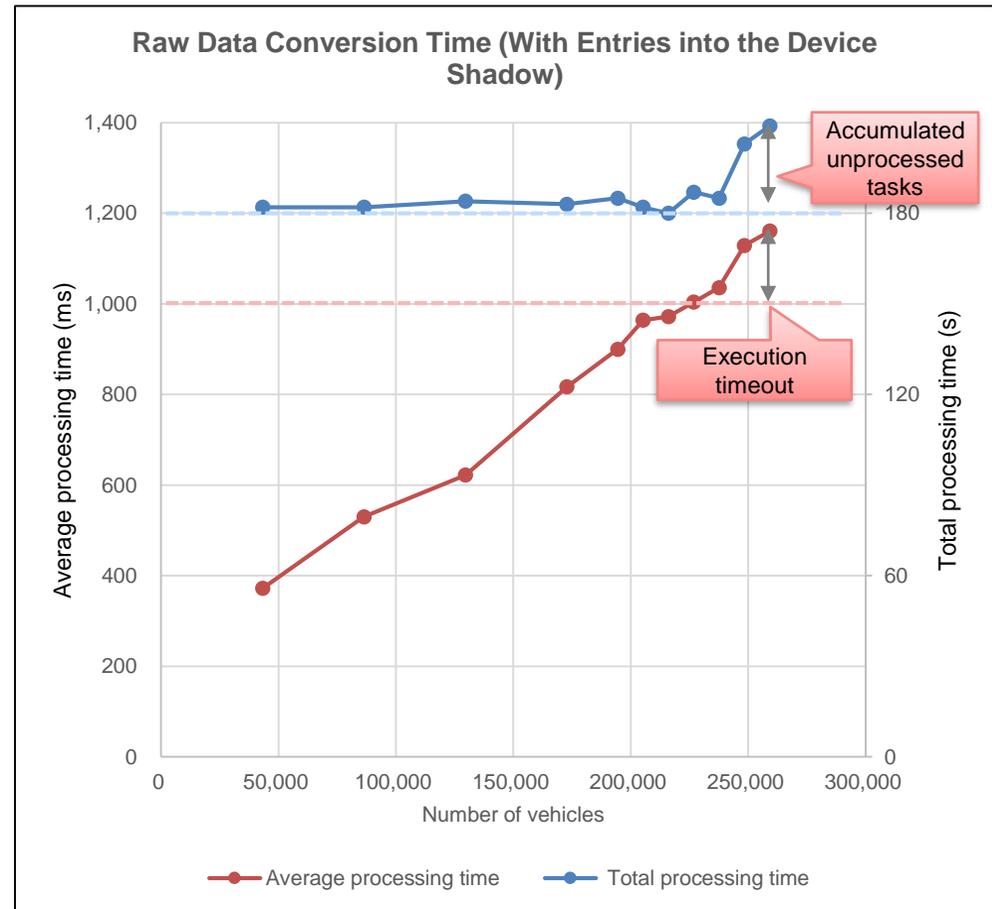
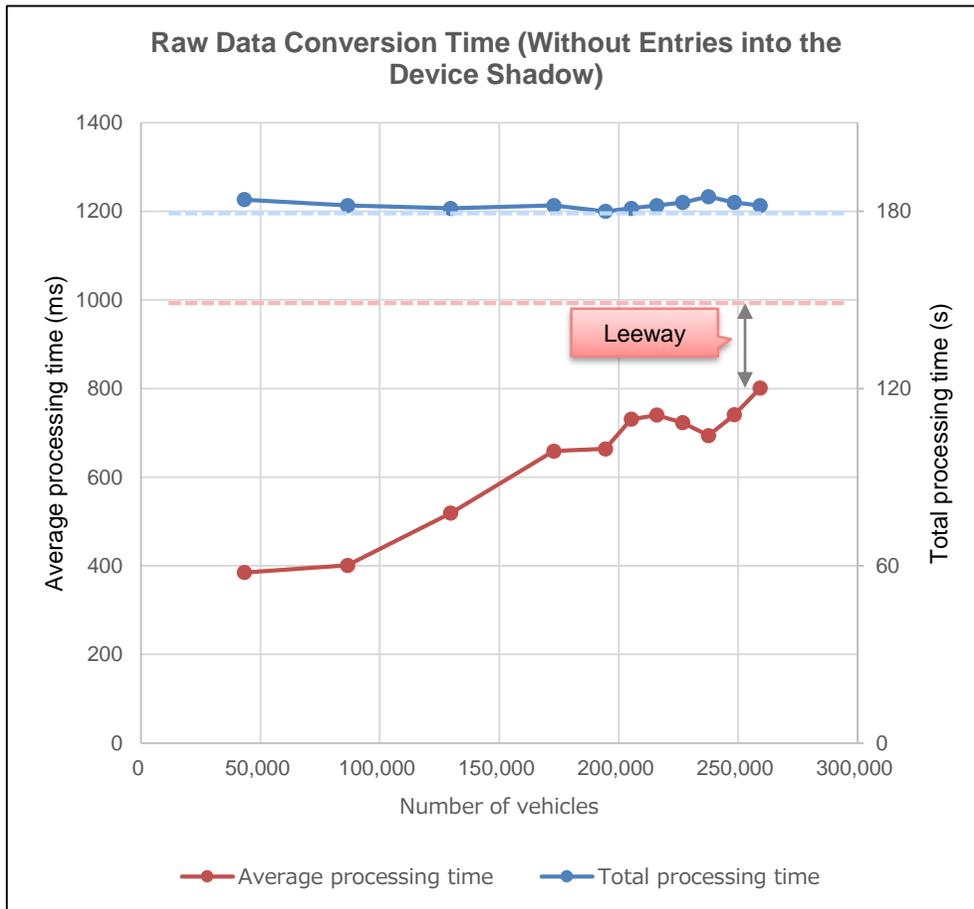
The maximum number of vehicles that meet the following conditions was measured.

- Average processing time for one second of data must be **one second or faster**
- Total processing time for inputted message load must be around **180 seconds**



# Evaluation 1-a: Results

We increased the number of vehicles to 260,000 and measured the average processing time for one second of data and the total processing time for the inputted message load. We confirmed that performance deteriorated due to the additional process for entering data into the device shadow, reaching the marginal performance at around 240,000 vehicles/second.



# Evaluation 1-b

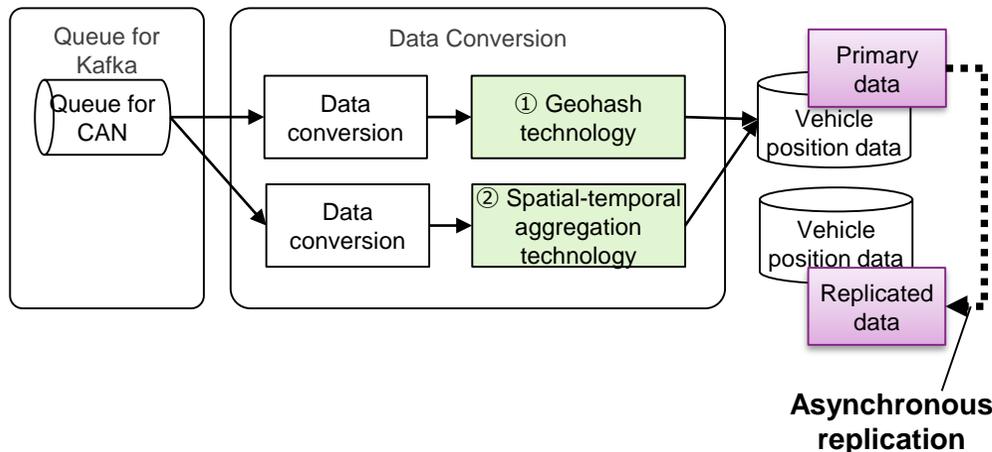
We confirmed the possibility of processing vehicle data storage for the quantitative target of 600,000 simultaneously driving vehicles (5 million × 12% of peak operational capacity).

## Description

We verified storage processing for vehicle position data by comparing performance based on the following technologies.

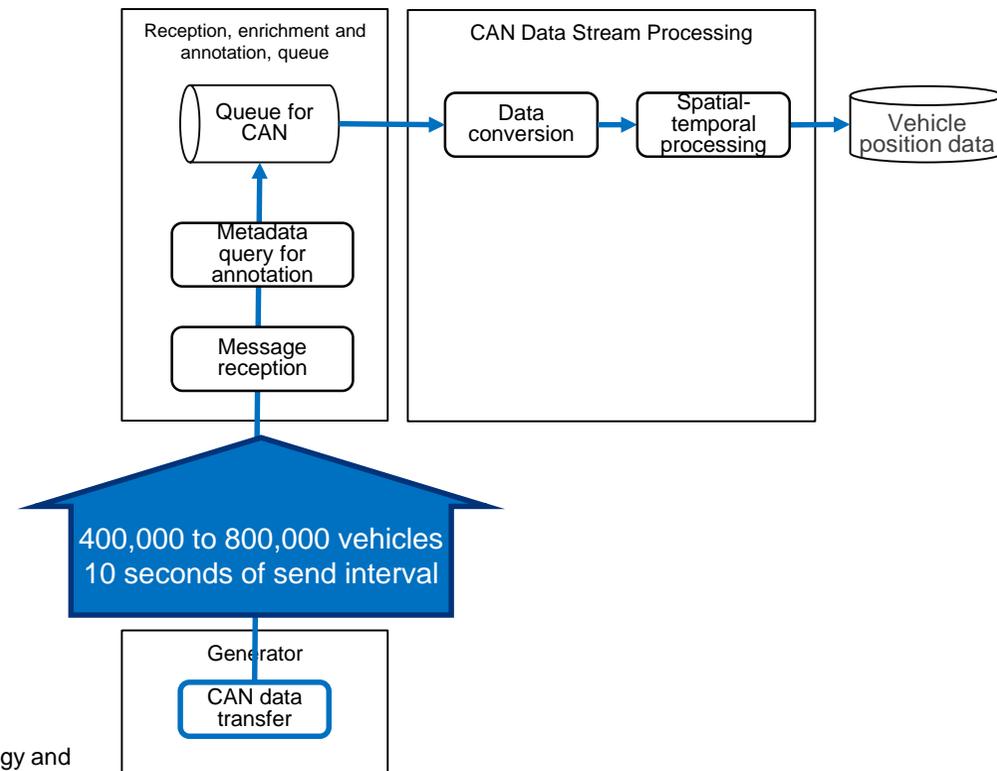
- ① Geohash
- ② Spatial-temporal aggregation\*1

Marginal performance was defined as the maximum load at which the processing can be maintained with replicating vehicle position data.



## Conditions

The maximum number of items processed per second was measured.



\*1 For details, see Appendix 1: High-performance spatial-temporal data management technology and high-performance spatial-temporal query technology

# Evaluation 1-b: Results

We measured the maximum number of vehicles and confirmed the possibility of processing 504,000 vehicles per second based on the Geohash method and 684,000 vehicles per second based on spatial-temporal aggregation. Storage performance was enhanced by up to 1.36 times by adopting spatial-temporal aggregation.

## (1) Geohash

Number of vehicles	Processing within fixed intervals*1	Status of replication*2	Results
414,000	Pass	Pass	Attained
<b>504,000</b>		Pass	<b>Attained</b>
540,000		Fail	Unattained
594,000		Fail	Unattained
648,000		Fail	Unattained
684,000		Fail	Unattained
774,000		Fail	Unattained
864,000		Fail	Unattained

## (2) Spatial-Temporal Aggregation

Number of vehicles	Processing within fixed intervals*1	Status of replication*2	Results
414,000	Pass	Pass	Attained
504,000		Pass	Attained
540,000		Pass	Attained
594,000		Pass	Attained
648,000		Pass	Attained
<b>684,000</b>		Pass	Attained
774,000		Fail	Unattained
864,000		Fail	Unattained

\*1 Pass ... Processing time  $\leq$  Continuous processing possible at startup intervals, Fail ... Processing time  $>$  Continuous processing difficult at startup intervals

\*2 Pass ... Continuous processing possible without stagnating replication, Fail ... Replication stagnates, continuous processing difficult without stagnating replication

# Verification 2

We confirmed the effectiveness of detection during search processing through potential congestion detection by mesh and lane aggregation as an example.

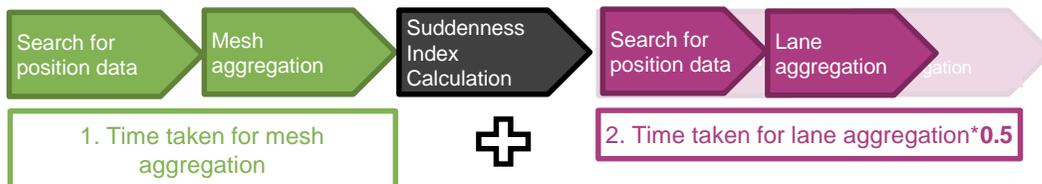
## Description

We adopted mesh aggregation and the Suddenness Index Calculation Method for Aggregated Values for reducing the load on lane aggregation in the latter stage. In this verification, we measured aggregation times as shown below and calculated total processing time to confirm the effect of using mesh aggregation for the preceding stage.

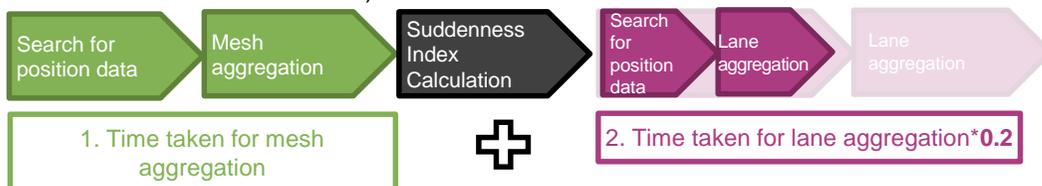
(1) Lane aggregation only (without detection)



(2) Mesh aggregation + around 50% of total lane aggregation (detection for mesh aggregation only)

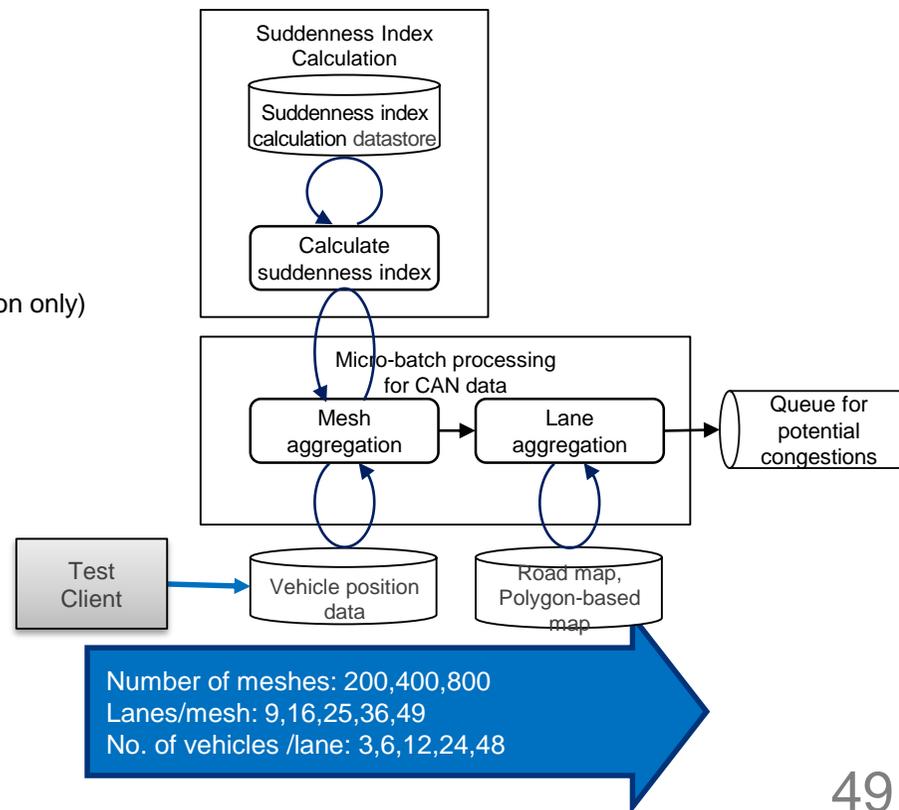


(3) Mesh aggregation + 20% of total lane aggregation (detection for mesh aggregation + Suddenness Index Calculation)



## Conditions

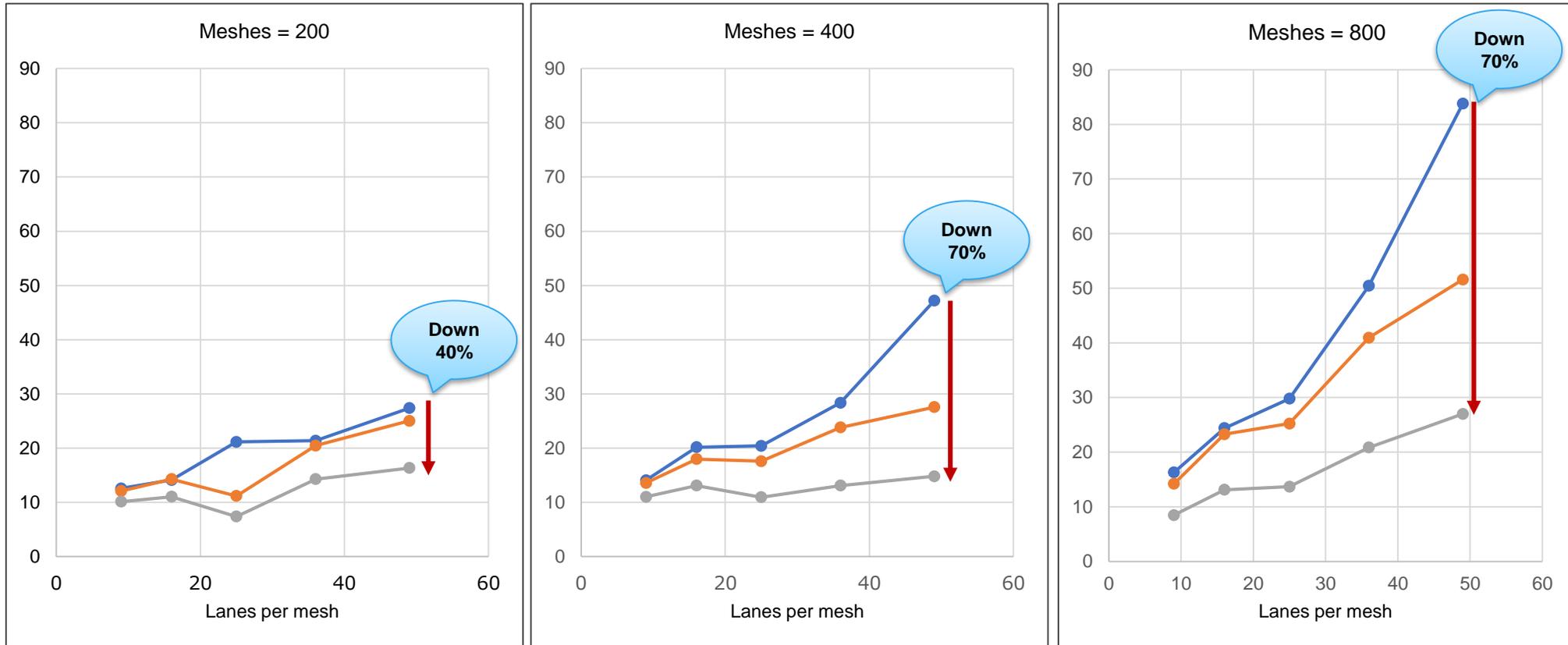
Aggregation time was measured by changing position data within the mesh and lanes.



# Evaluation 2: Results

We confirmed that the processing time can be reduced by using mesh aggregation in advance to detect target vehicles for lane aggregation. We consider the effect of detection is more effective with the increase in the number of lanes and vehicles due to the growing time gap between mesh aggregation and lane aggregation.

Estimates of Processing Time According to Changes in the Number of Meshes (Seconds)



① Only lane aggregation    ② Mesh aggregation + 50% of lane aggregation    ③ Mesh aggregation + 20% of lane aggregation

# Summary

Evaluation Points	Summary	Future Issues
<p>1. CAN data stream processing</p>	<p>We confirmed the possibility of expanding CAN data stream processing to 30 million vehicles according to estimates based on the results of platform performance verification.</p> <ul style="list-style-type: none"> <li>Processing for raw data conversion to SI unit values</li> </ul> <p>We compared performance in cases with and without entries into the device shadow and confirmed the marginal performance without entries to be 240,000 TPS. We can raise the processing time by 1.3 to 1.5 times by adding processing for device shadow storage to processing for raw data conversion to SI unit values.</p> <ul style="list-style-type: none"> <li>Storage of vehicle position data</li> </ul> <p>We compared the performance of storage processing of vehicle position data based on the Geohash method and spatial-temporal aggregation and confirmed the possibility of handling up to 504,000 vehicles under the Geohash method and up to 684,000 vehicles under the spatial-temporal aggregation. Adoption of spatial-temporal aggregation enhanced storage performance up to 1.36 times.</p>	<ul style="list-style-type: none"> <li>Improve architecture to address abnormal circumstances including server failures</li> <li>Improve architecture to enable response to spikes</li> <li>Develop architecture for economically rational data management, such as shifting infrequently used data to low-cost storage areas</li> </ul>
<p>2. Micro-batch processing for CAN data</p>	<p>To confirm the effect of detection, we calculated the following estimates based on the results of platform performance verification. The results of platform performance verification, however, have been excluded from this document.</p> <ol style="list-style-type: none"> <li>Lane aggregation only (without detection)</li> <li>Mesh aggregation + around 50% of total lane aggregation (detection for mesh aggregation only)</li> <li>Mesh aggregation + 20% of total lane aggregation (detection for mesh aggregation + Suddenness Index Calculation)</li> </ol> <p>We were able to reduce the number of target lanes for aggregation by including a detection process in advance as in case (3), thereby achieving a maximum 70% reduction in processing time compared to case (1), in which aggregation was conducted on all lanes.</p> <p>We consider detection to be more effective with the increase in the number of lanes and vehicles due to the growing time gap between mesh aggregation and lane aggregation.</p>	<ul style="list-style-type: none"> <li>Improve throughput and turnaround time (TAT) by adopting parallel processing for raw data conversion to SI unit values</li> <li>Improve throughput and turnaround time (TAT) by enhancing the interface</li> <li>Improve scalability by storing the suddenness index table to distributed datastores</li> <li>Use machine learning to upgrade the suddenness index table</li> </ul>

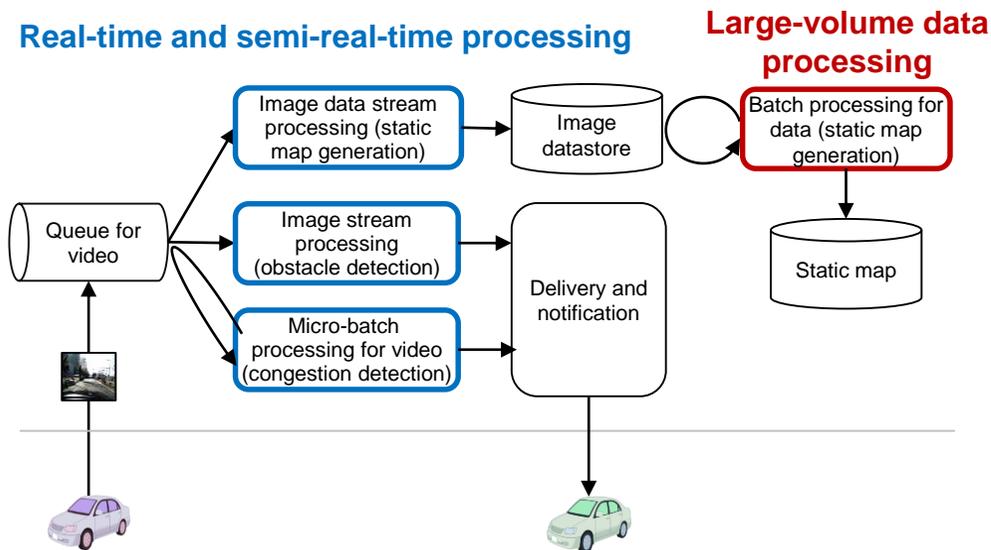
# Image Data Processing

# Evaluation Overview

We measured the performance of each processing method (stream processing, micro-batch processing, and batch processing) executed by the platform side to process video sent by vehicles.

## Overview

The volumes of video data collected from vehicles is massive, so the processing method must be determined according to performance requirements. In this verification, we measured the performance of stream, micro-batch, and batch processing.



## Evaluation Points

1. Image data stream processing
  - a. Executed stream processing for video data sent by vehicles and measured the maximum load of real-time processing. Since GPU processing represents the bottleneck for the entire system, processing tasks were off-loaded to off-site GPU nodes depending on operational status.
  - b. Verified basic performance by changing data input to the application for detecting lines of vehicles and cause of congestion.
2. Batch processing for video  
Executed batch processing for large volumes of video data accumulated in the image datastore to verify the marginal performance of GPU processing.

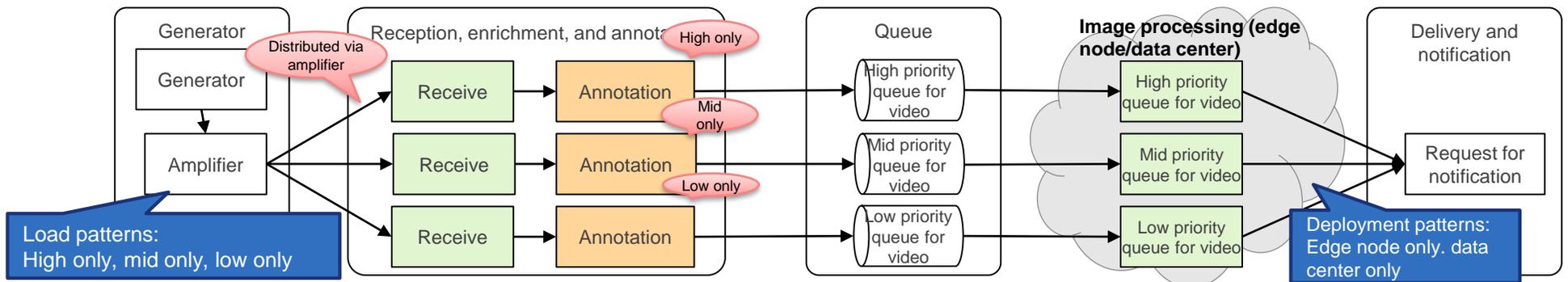
# Evaluation 1-a

Confirmed the marginal performance for a guaranteed response by executing stream processing for video data sent from vehicles. Since GPU processing represents the bottleneck for the entire system, processing tasks were off-loaded to off-site GPU nodes depending on operational status. We also verified processing time, with or without task off-loading.

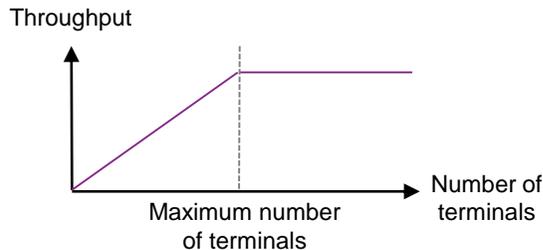
## Description and Conditions

The load was applied in the form of image data by enabling dynamic selection technology for processing nodes\*1. Priorities (processing sites) were changed by modifying the input criteria in the preceding process of dynamic prioritization, and processing throughput for each site ① and data processing time ② were measured.

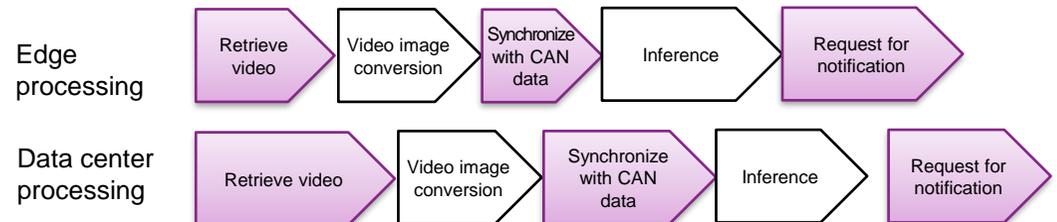
Changes in throughput and processing time were verified by adjusting prioritization rules and load patterns to ensure the appropriate measurement of performance for each processing site.



(1) Measure throughput of image processing



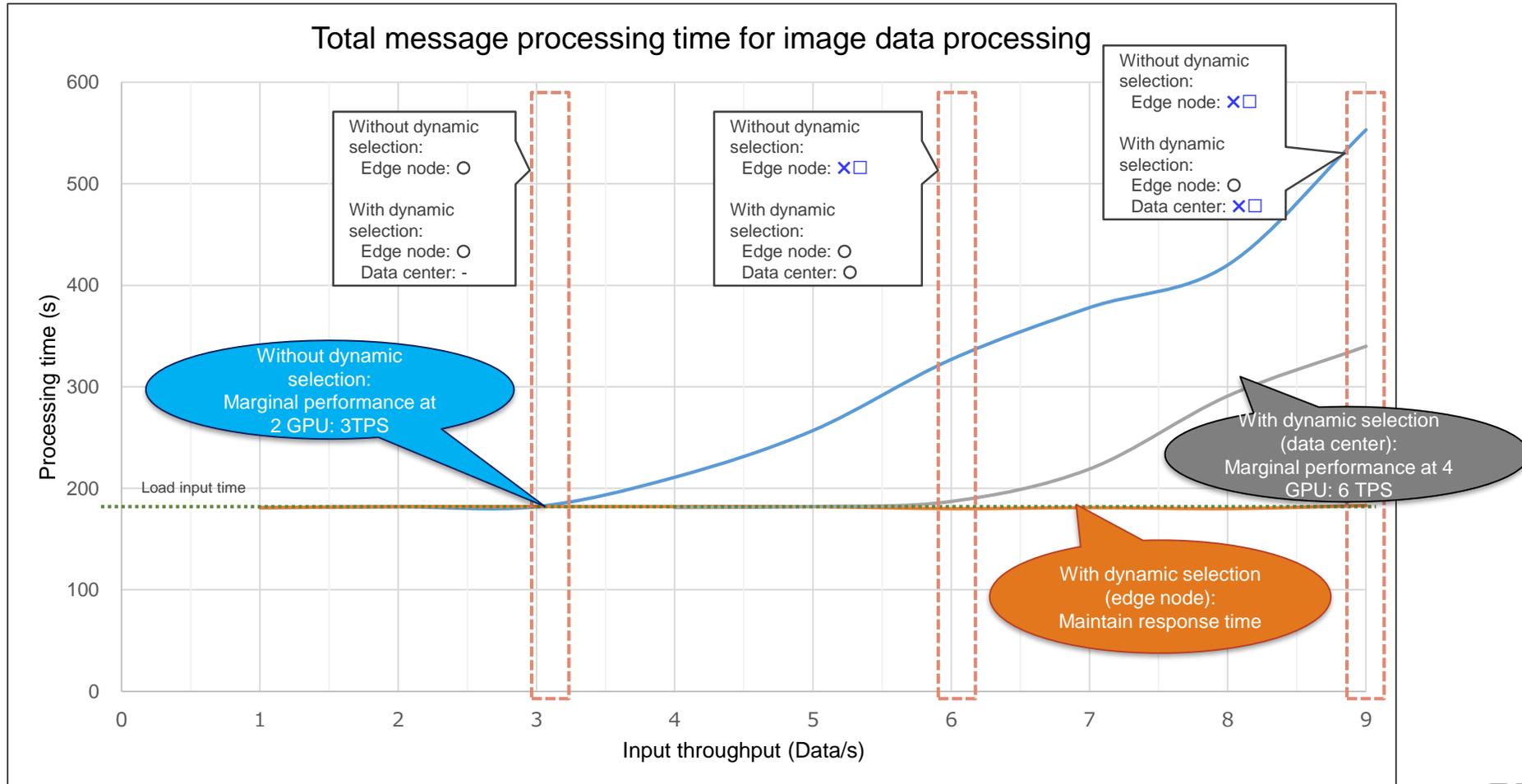
(2) Analyze processing time for each task



\*1 For details, see Appendix 4: Vertical-distributed computing technology

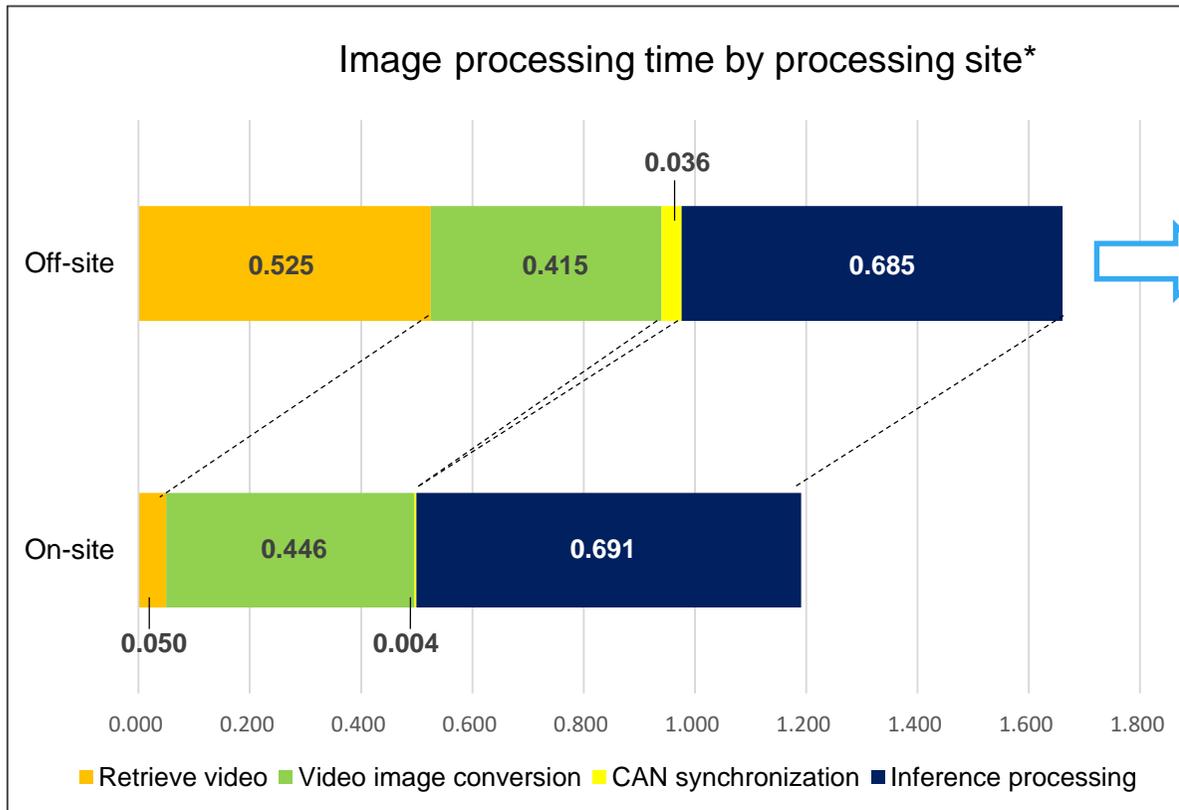
# Evaluation 1-a: Results (1/2)

The comparison between the performance with and without the use of dynamic selection technology for processing nodes found that by using dynamic selection technology for processing nodes for high priority on-site processing we can maintain the response time through keeping the load at a certain level, thereby ensuring that processing is completed within a fixed amount of time regardless of load status. We also confirmed that marginal performance values can be enhanced by off-loading image data processing off-site to increase the number of GPU nodes handling the tasks.



# Evaluation 1-a: Results (2/2)

As a result of analyzing time required for on-site and off-site image data processing, we found there were no effects other than on the time taken to retrieve the video. Therefore, the time required for data transfer is the only additional value when processing off-site.



## Effect of off-site deployment on processing time

- Increased:
  - Retrieve video
  - CAN synchronization (marginal increase in transfer time due to small data size)
- No change:
  - Video image conversion
  - Inference processing

\*Processing time is the average value resulting from multiple trials.

# Evaluation 1-b

We verified basic performance when the data input is changed to the application for detecting lines of vehicles and cause of congestion.

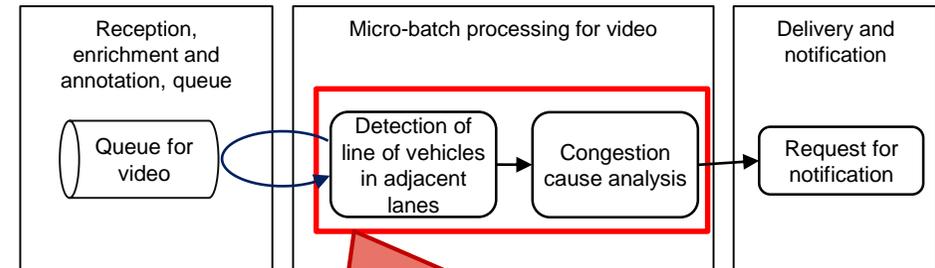
## Description and Conditions

We measured the processing time and throughput of the technology for detecting vehicle lines and cause of congestion by varying the following data input conditions.

- i. Length of video
- ii. Frame rate
- iii. Bit rate (resolution)

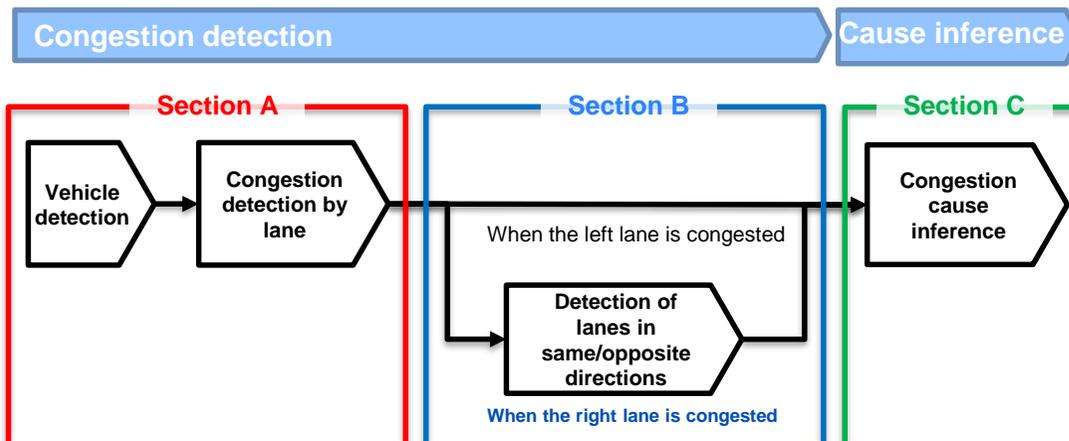
Since GPU use is differed in three sections during image processing, we defined them as A, B, and C and separately analyzed processing times.

### Variations in data input



Confirmed basic performance for variations in data input

- i. Length of video: 10 s, 30 s, 60 s
- ii. Frame rate: 10 fps, 30 fps, 60 fps
- iii. Bit rate:
  - $1,920 \times 1,080$  (1,080 p)
  - $3,840 \times 2,160$  (2,160 p)
  - $7,680 \times 4,320$  (4,320 p)

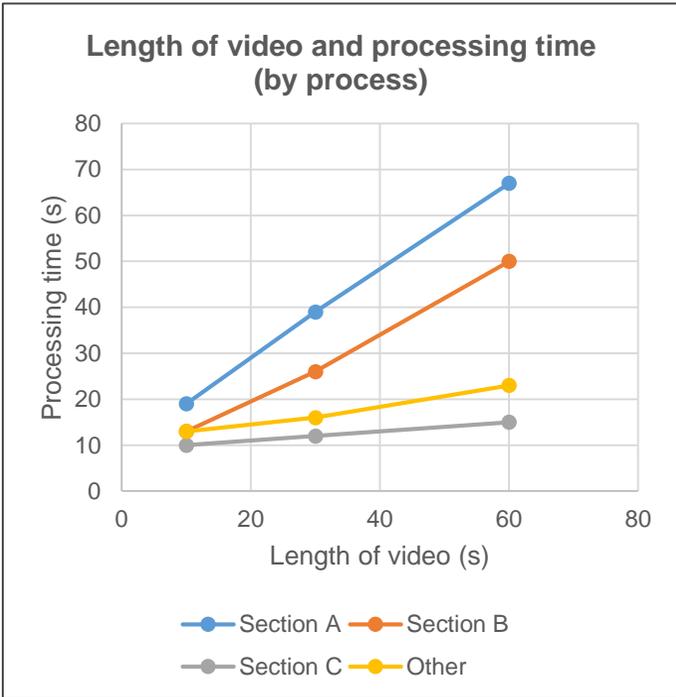


# Evaluation 1-b: Results

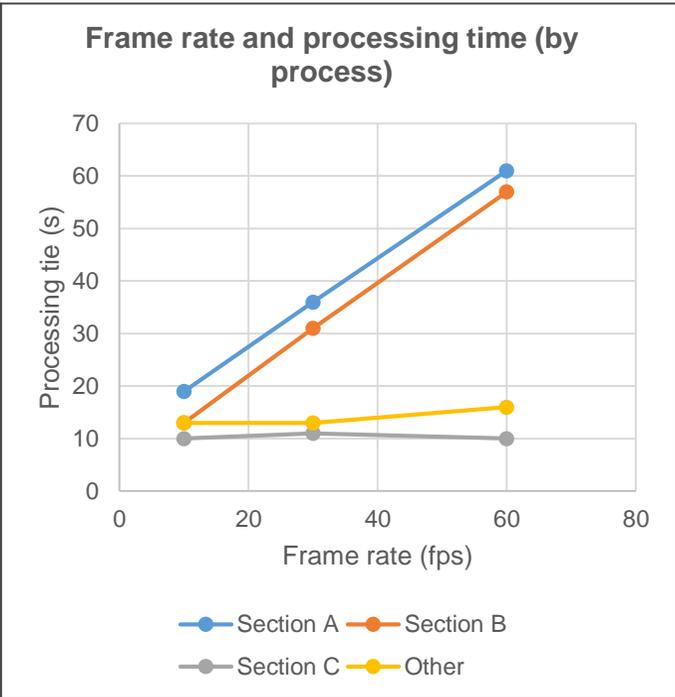
Through the evaluation, we found the following relationships between processing time and variations in data input.

- i. Length of video: Processing time increases linearly in proportion to the length of video in Sections A, B, and C
- ii. Frame rate: Processing time increases linearly in proportion to the frame rate in Sections A and B, and remains within a fixed band for Section C
- iii. Bit rate: Processing time increases linearly in proportion to the bit rate in Sections A and B, and remains within a fixed band for Section C

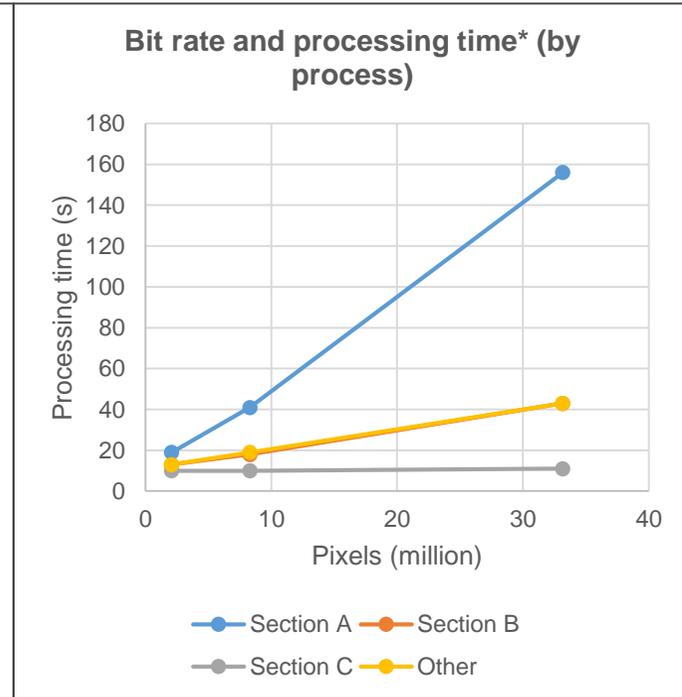
## I. Length of video



## II. Frame rate



## III. Bit rate



No.	Length of video	Section A	Section B	Section C	Other	Total
1	10.1	19	13	10	13	55
2	30.2	39	26	12	16	93
3	60.6	67	50	15	23	155

No.	Frame rate (fps)	Section A	Section B	Section C	Other	Total
1	10	19	13	10	13	55
2	30	36	31	11	13	91
3	60	61	57	10	16	144

No.	Resolution	Pixels (million)	Section A	Section B	Section C	Other
1	1,080 p	2.07	19	13	10	13
2	2,160 p	8.29	41	18	10	19
3	4,320 p	33.17	156	43	11	43

\*Time measured in seconds

# Evaluation 2

We verified the marginal performance of GPU processing by executing batch processing for large volumes of video data accumulated in the image datastore.

## Description and Conditions

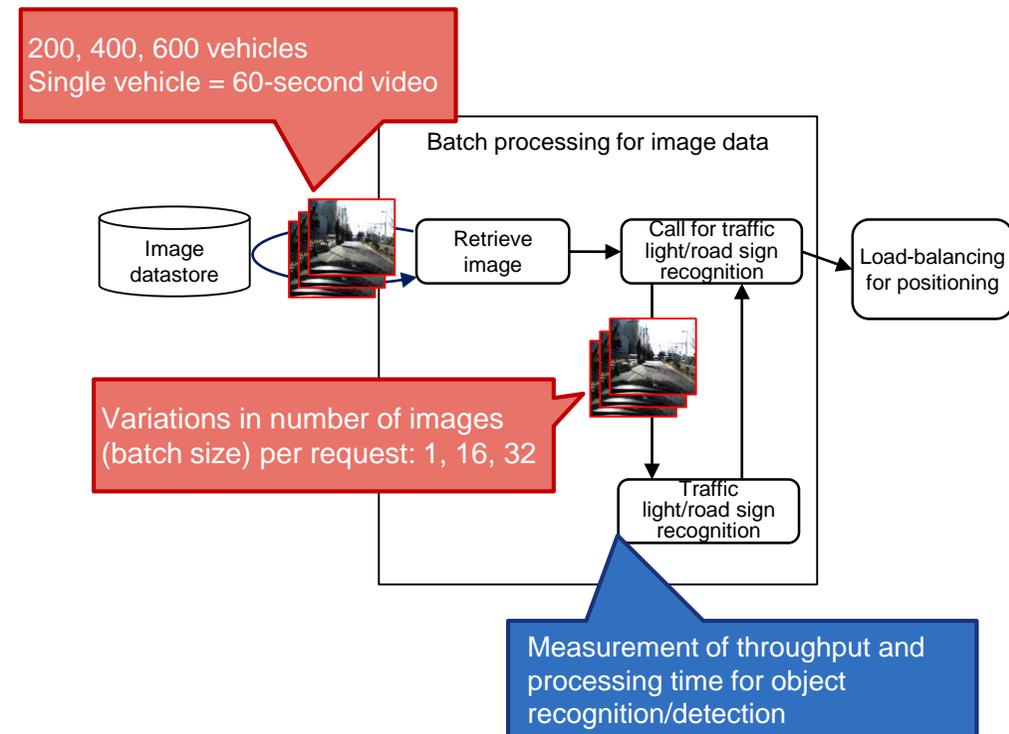
Obtained marginal performance values for batch processing of large volumes of video data accumulated in the image datastore. Variable parameters were used as follows to measure throughput and processing time for object recognition/detection.

- (1) Number of images processed in a single inference request (batch size)
- (2) Image data size

### Type of image data

No.	Frame rate	Bit rate	Resolution	Image size (average)
1	10 fps	6,200 kbps	High	295 KB
2	10 fps	6,200 kbps	Low	77 KB
3	30 fps	6,200 kbps	High	295 KB
4	10 fps	4,800 kbps	Low	291 KB
5	10 fps	8,000 kbps	High	298 KB

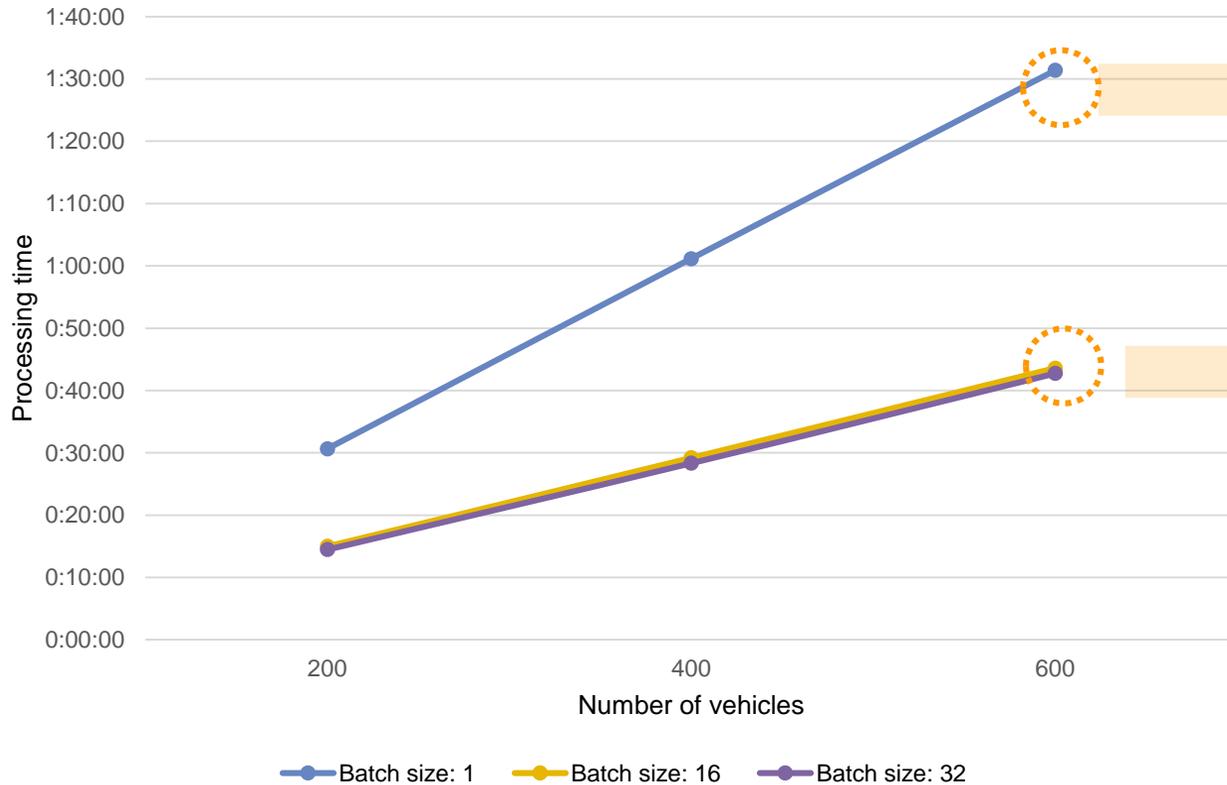
### Variations in parameters



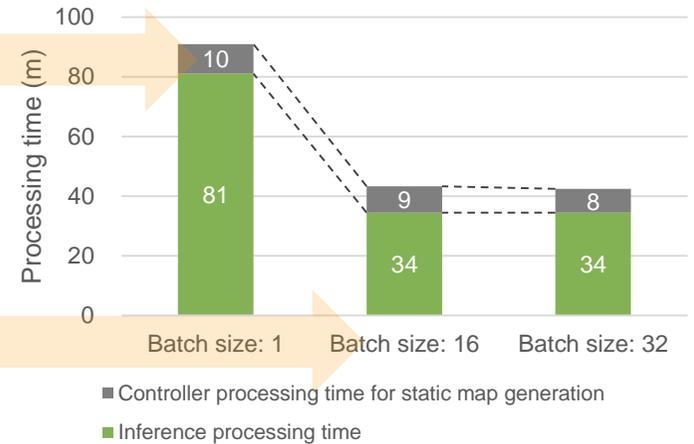
# Evaluation 2: Results (1/2)

We found that the inference processing time can be reduced by increasing batch size. We also confirmed that batch processing time increases in proportion to the number of vehicles. Performance peaks at a batch size of around 32, and an OOM error occurs when the batch size exceeds 40, due to lack of GPU memory (16 GB).

Processing time according to batch size



Breakdown of batch processing time (600 vehicles)



Batch size	Inference processing time (average)	Inference processing time (average) per image
1	53.9 ms	53.9 ms
16	363 ms	22.69 ms
32	705 ms	22.03 ms*

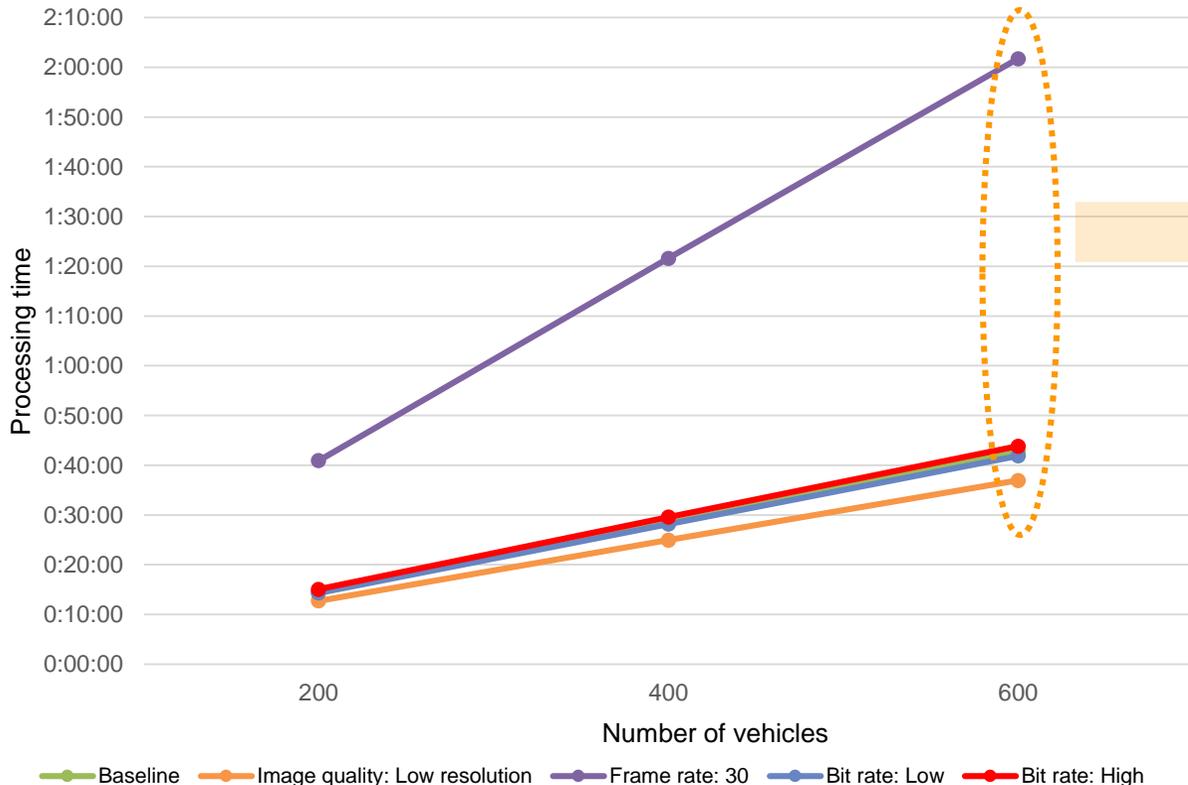
\*Larger batch size requires more inference processing time due to the greater number of images processed per request.

# Evaluation 2: Results (2/2)

Performance characteristics for variations in image data were as follows.

- Resolution: Although the size of low-resolution image data was roughly 0.26 times that of high-resolution image data, processing time remained at around 0.86 times that of high-resolution image data (since image resolution is an element that contributes to raising precision in subsequent object-position-estimation processing, high-resolution images are preferable if the effect on processing time is limited).
- Frame rate: In line with our hypothesis, an increase in the frame rate raises the volume of image data to be subjected to inference processing, thereby extending batch processing time.
- Bit rate: The effect on batch processing time is marginal. We believe this is because image quality is fixed when extracting the frames, so the bit rate causes no difference in data size.

Processing time according to differences in video data size



Magnification of processing time for baseline video data and variable parameters

Resolution	Processing time (m)	Magnification
High resolution (baseline)	43	-
Low resolution (0.26 times)	37	0.86

Frame rate	Processing time (m)	Magnification
10 fps (baseline)	43	-
30 fps (3 times)	122	3

Bit rate	Processing time (m)	Magnification
6,200 kbps (baseline)	43	-
4,800 kbps (0.77 times)	42	0.98
8,000 kbps (1.3 times)	44	1.02

# Summary

Evaluation Points	Summary	Future Issues
1. Image data stream processing	<p>a. In the past, the limit of a single edge node was the marginal performance of the entire system. However, we confirmed the following correlations by adopting vertical-distributed computing.</p> <ul style="list-style-type: none"> <li>For high priority on-site processing, response time can be maintained by keeping the load at a certain level, thereby ensuring that processing is completed within a fixed amount of time regardless of load status.</li> <li>Marginal performance values are enhanced by off-loading image data processing off-site to increase the number of GPU nodes handling the tasks (marginal performance = <math>1.5 \times</math> number of GPU nodes).</li> </ul> <p>b. The following is a summary of the basic performance verification for variations in data input to the application for detecting lines of vehicles and cause of congestion.</p> <p>i. Length of video</p> <ul style="list-style-type: none"> <li>Congestion detection: Processing time increases linearly in proportion to video length</li> <li>Cause inference: Processing time is not dependent on video length and remains fixed*</li> </ul> <p>*In this verification, we used the same video on repeat as input data, which resulted in magnifying the increase in the number of congestions in proportion to video length, thereby extending processing time.</p> <p>ii. Frame rate</p> <ul style="list-style-type: none"> <li>Congestion detection: Processing time increases linearly in proportion to frame rate</li> <li>Cause inference: Processing time is not dependent on the frame rate and remains fixed</li> </ul> <p>iii. Bit rate</p> <ul style="list-style-type: none"> <li>Congestion detection: Processing time increases linearly in proportion to the video length</li> <li>Cause inference: Processing time is not dependent on the bit rate and remains fixed</li> </ul>	<p>Improvement in throughput can be accelerated further by combining the following.</p> <ul style="list-style-type: none"> <li>Reducing traffic by off-loading inference processing to vehicles</li> <li>Boosting GPU resources through cloud integration</li> <li>Reducing the cost of video transfer and processing by selective uploading by vehicles</li> </ul> <ul style="list-style-type: none"> <li>Improving throughput and turnaround time (TAT) by eliminating the two-step disc entries during preprocessing</li> <li>Improving throughput and turnaround time (TAT) by executing chunk processing via functional integration with preprocessing</li> <li>Supplementing deficiencies and lack of resolution in the CAN data from the platform side</li> </ul>
2. Batch processing for video	<ul style="list-style-type: none"> <li>Variations in batch size We confirmed that throughput can be improved by increasing batch size to effectively use GPU. However, GPU becomes the bottleneck when batch size is raised too high, causing throughput to peak.</li> <li>Variations in data size We confirmed that by increasing frame rate, processing time increased at the same magnitude. Also, since increasing the resolution of image quality to a given level did not affect processing time, high-quality image processing for the improved precision of object-position-estimation processing is realistic.</li> </ul>	—

# Delivery and Notification

# Evaluation Overview

In the obstacle detection use case, the platform sends a request to collect video data prior to receiving video data from vehicles. To reduce traffic volume for the data collection, we applied a method of selecting the target vehicles that use information about the relative positions of the vehicles and obstacles.

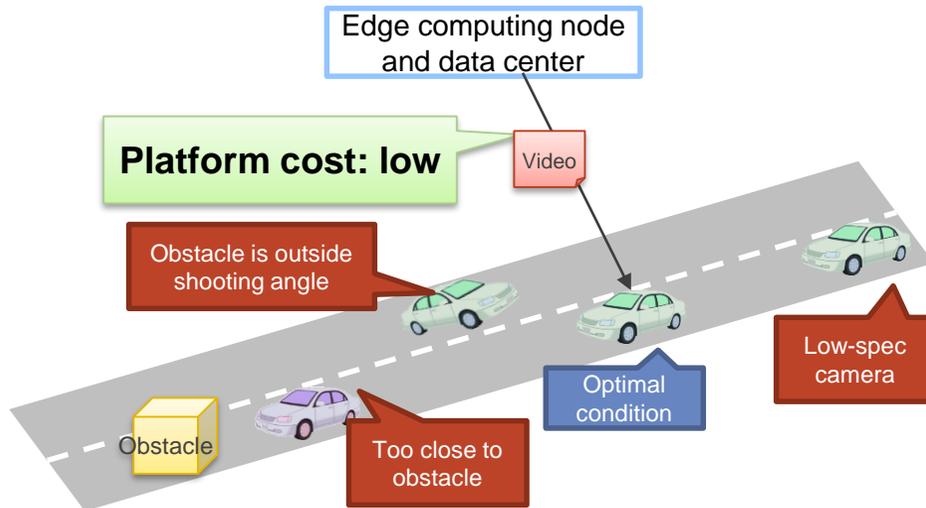
## Overview

We evaluated the performance of the data acquisition system architecture, which incorporates a target vehicle selection algorithm\*1.

## Evaluation Points

1. Processing time for vehicle selection  
The relationship between the number of vehicles to be selected and processing time for vehicle selection was verified by reproducing a situation that involves searching a large volume of vehicle data.

### Overview of Evaluation



\*1 The use of this algorithm is expected to reduce platform costs caused by the concentration of loads and processing times by collecting video data exclusively from optimal vehicles based on their direction, angle of view, and other information, in addition to collecting vehicle position data. For details, see Appendix 2: Data acquisition technology with target vehicle selection algorithm

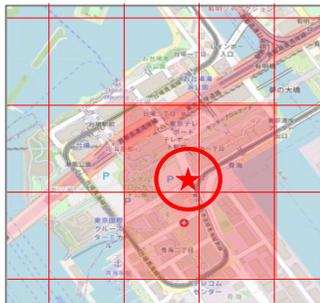
# Evaluation 1

We evaluated the relationship between the number of vehicles to be selected and the processing time for vehicle selection by reproducing a situation that involves searching a large volume of vehicle data.

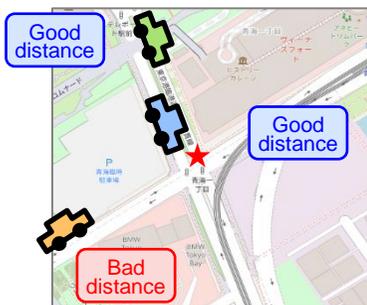
## Description

The video collection process using the target vehicle selection algorithm consists of these four steps. We measured processing time for each step to verify the relationship between the number of vehicles to be selected and processing time for vehicle selection.

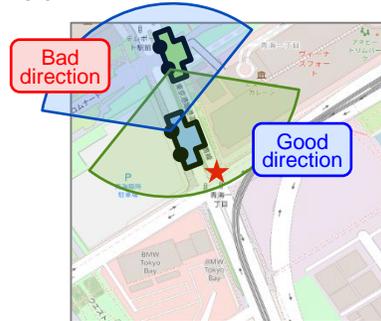
(1) Search vehicle position database



(2) Distance detection



(3) Direction detection



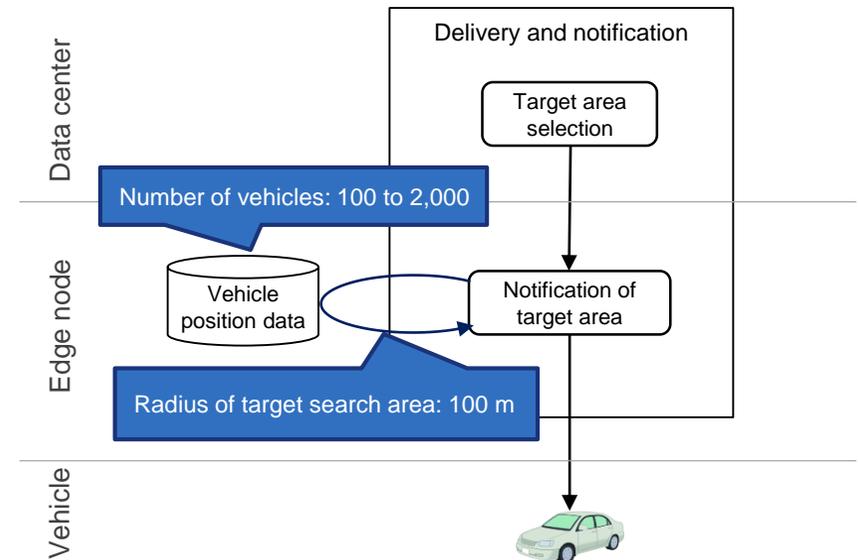
(4) Coverage detection



© OpenStreetMap contributors

## Conditions

Processing time was measured by varying the number of vehicles in the search area.



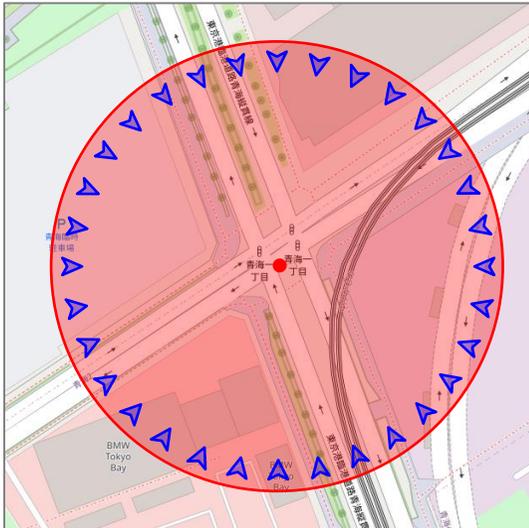
### Rationale for setting the upper limit at 2,000 vehicles

- Area of circle with 100 m radius:  $100 \times 100 \times 3.14 = 31,400 \text{ m}^2$
- Grid for parking lot per vehicle:  $3 \times 5.5 = 16.5 \text{ m}^2$
- Number of vehicles that fits inside the circle:  $31,400 \div 16.5 = 1,903$

# Evaluation 1: Results

We conducted the evaluation by increasing the number of target vehicles and observed an upward trend in processing time for database search and coverage detection. In particular, coverage detection became the dominant factor when the number of target vehicles exceeded 1,000.

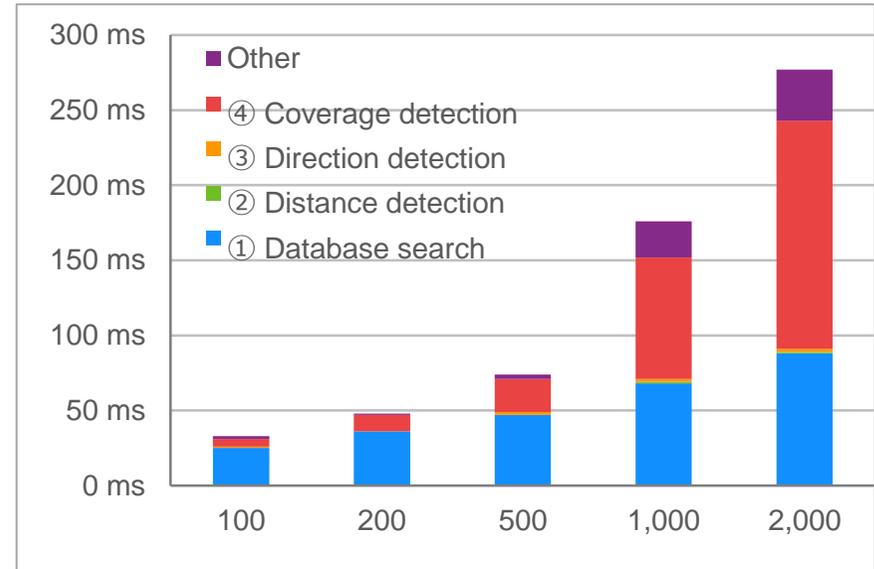
## Arrangement of vehicles used in the evaluation



© OpenStreetMap contributors

- Target search area 
  - Center: “Aomi 1-chome” intersection
  - Radius: 100 m
- Vehicle data 
  - 100 to 2,000 (positioned at a 90 m radius)
  - Distance good, direction good, coverage good

## Relationship between the number of vehicles and processing time



- Processing times for ② distance detection and ③ direction detection were minimal
- Processing times of ① database search and ④ coverage detection were increased by increase of the number of target vehicles
- In particular, the processing time varied significantly depending on the number of vehicles that require coverage detection
  - Around 280 ms for 2,000 vehicles
  - Coverage detection became the dominant factor when the number of vehicles exceeded 1,000
  - Followed the theoretical value for  $O(n)$

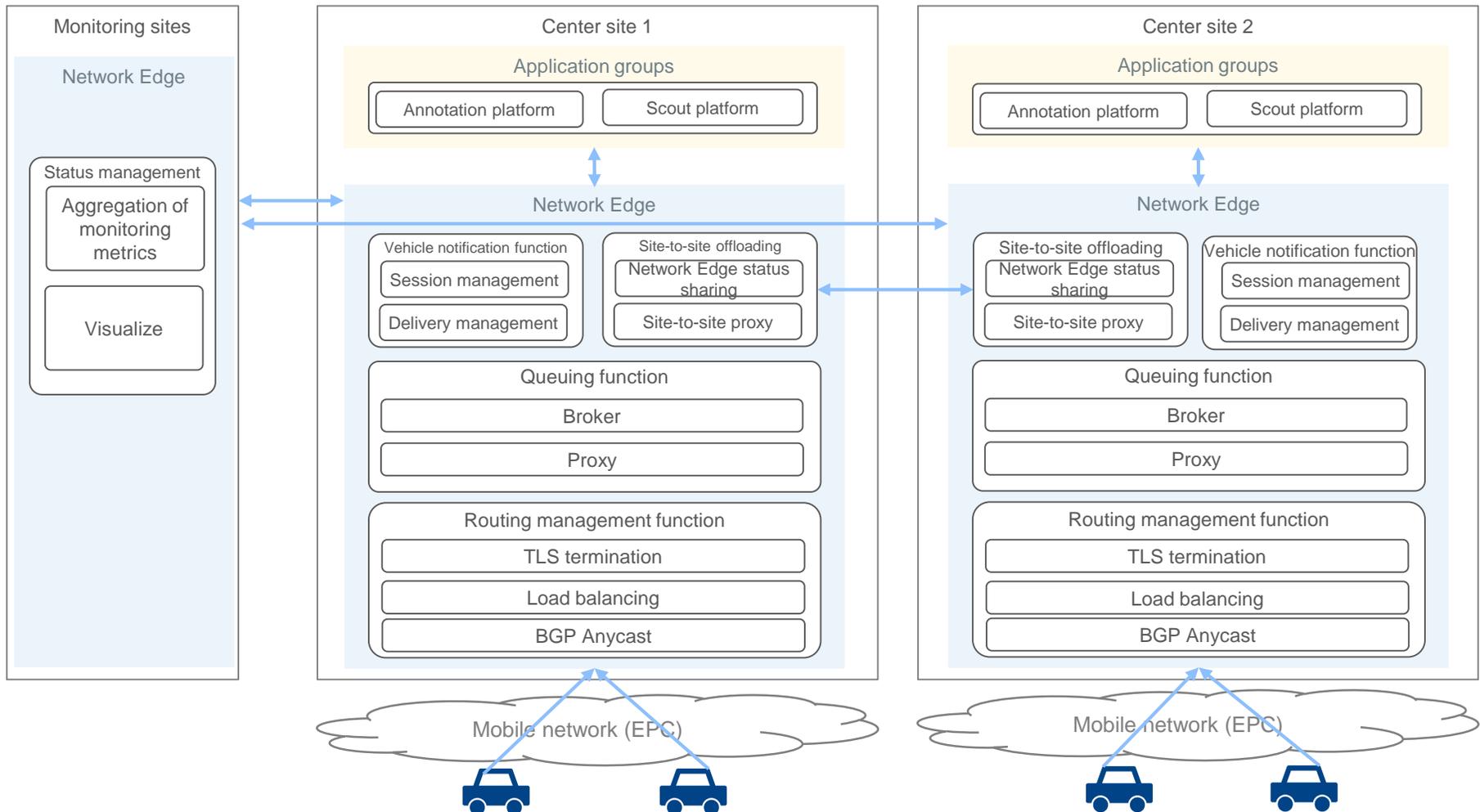
# Summary

Evaluation Points	Summary	Considerations
1. Processing time for vehicle selection	<p>We varied the number of vehicles to be selected for collecting vehicle position data and measured the processing time for each step of the target vehicle selection algorithm. As a result, we observed the following results.</p> <ul style="list-style-type: none"><li>• Processing times for “Distance detection” and “Direction detection” are minimal</li><li>• There was an upward trend in “Database search” and “Coverage detection”</li><li>• In particular, the processing time varies significantly depending on the number of vehicles that require coverage detection, measuring around 280 ms when 2,000 vehicles are selected</li></ul>	<p>The following points were considered in terms of effectively applying the target vehicle selection algorithm.</p> <ul style="list-style-type: none"><li>• “Distance detection” and “Direction detection” require minimal processing time, so from the perspective of eliminating excess notifications, the algorithm should be applied in use cases where the priority is to provide a real-time response and in use cases that only require notification to be completed.</li><li>• Given the nature of processing “Coverage detection,” the algorithm should be applied in use cases where the priority is on the precision of notification as well as in use cases where the collection of large-volume data, such as video, is linked to the notification. When applying the algorithm to use cases prioritizing a real-time response, the trade-off between speed and the effect on processing time must be considered.</li></ul>

# Evaluation of Network-Edge Computing Platform

# Evaluation Overview (1/4)

To realize an optimal flow of CAN and video data within the widely dispersed sites of the connected vehicle system, the architecture between the vehicles and application groups was designed to include a multiple-function platform system, and we conducted an experiment to verify the platform's effectiveness at addressing the technical issues of the connected vehicle system. This platform will be called "Network Edge" to distinguish it from application groups that are also arranged on the edge node.

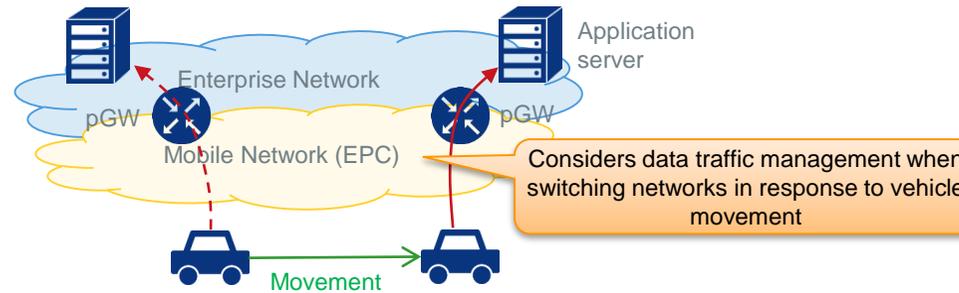


# Evaluation Overview (2/4)

To define the data flow from vehicles to application groups dispersed at multiple sites, we defined basic use cases as those that involve switching application server in response to vehicle movement and to wide-area routing (switching due to the localized concentration of loads).

## Vehicle movement

We switched application server in response to vehicle movement. Tasks that require processing based on network edge computing were defined as applied use cases related to the basic case of vehicle movement.

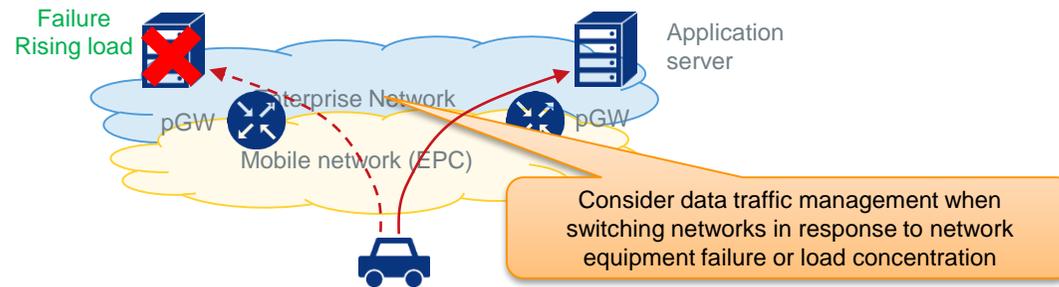


Basic Use Case	Applied Use Case
Vehicle movement	Supports change in access from one edge platform to another to enable tracking moving vehicles
	Supports fast response and manual switching while tracking rapidly moving vehicles

# Evaluation Overview (3/4)

## Wide-area routing

The application server is switched from one to another in response to triggers such as site failures and load distribution. Tasks that require processing based on network edge computing were defined as applied use cases related to the basic use case of wide-area routing.



Basic Use Case	Applied Use Case
Wide-area routing	Reduce data volume and execute preprocessing by using the edge platform to reduce the load on network infrastructure
	Limit redundant/similar data sent from vehicles when there is a localized concentration of vehicles to prevent such data from affecting the platform
	Select the edge platform based on the equipment's load status, processing time, and access quality
	Switch some of the vehicles to another edge server when network congestion is detected
	Equipment units will expand resources under high load through the manual and automatic scaling of resources

# Evaluation Overview (4/4)

We established an environment with multiple network edge computing sites to realize the basic use case and verified the optimal method for routing management between vehicles and each Network Edge site as well as the synchronization method between edge nodes. We also evaluated Transport Layer Security (TLS) performance, assuming that communication with vehicles will be encrypted based on TLS to ensure security.

Evaluation Points	Overview	Basic Use Case
1. Optimal routing management for sending data from vehicles to center site and application servers	<p>We considered two methods for routing management and verified the system's behavior in response to vehicle movement and wide-area routing (equipment failure) as well as the basic functions and performance of each method.</p> <ul style="list-style-type: none"> <li>• DNS method: Requests are guided by BGP Anycast to the nearest DNS server, which selects the optimal application groups (annotation server platform) to notify vehicles.</li> <li>• Load Balancer(LB) method: Requests are guided by BGP Anycast to the nearest LB server, which forwards the request from vehicles to the optimal annotation server.</li> </ul>	<p>Vehicle movement</p> <p>Wide-area routing</p>
2. Metrics-based routing management	<p>We verified the following aspects of network edge computing by adopting the LB method for routing management.</p> <ul style="list-style-type: none"> <li>• Combine the applications of DCM emulation, edge nodes, and data centers and confirm completion of processing along with our scenario</li> <li>• Obtain basic performance values</li> </ul>	<p>Wide-area routing</p>
3. Performance of TLS termination (CPU processing)	<ul style="list-style-type: none"> <li>• Confirm the possibility of scale-out based on multiple units</li> <li>• Confirm behavior in case of failures in network edge computing and application server</li> <li>• Route control and load balancing that utilize the collected metrics at the application level in addition to status information such as site failures.</li> </ul>	<p>—</p>
4. Message Queue (MQ)-based data flow between edge sites	<p>We considered an MQ-based method for data aggregation between sites in an environment with multiple distributed Network Edge sites and confirmed its behavior and performance.</p> <ul style="list-style-type: none"> <li>• Confirmed behavior of communication between vehicles and application groups via wide-area distributed MQ (including site-to-site data synchronization)</li> <li>• Verified performance of data transmission from vehicles</li> <li>• Verified performance of message notification to vehicles</li> </ul>	<p>Vehicle movement</p> <p>Wide-area routing</p>
5. TLS offload	<p>We verified the behavior of TLS offload based on two methods and the performance of each method for reducing CPU usage.</p> <ul style="list-style-type: none"> <li>• TLS Accelerator: Processing mode that uses the OpenSSL EVP application programming interface</li> <li>• SmartNIC (ASIC): Processing mode that uses Kernel TLS</li> </ul>	<p>—</p>
6. 5G field trials	<p>We uploaded and downloaded large volumes of data between vehicles mounted with LTE/5G modems and the platforms and measured the performance value.</p>	<p>—</p>

# Evaluation 1

We confirmed the behavior of routing management based on the DNS and LB methods and measured basic performance as follows.

## Description

Based on the assumptions of the basic use cases (vehicle movement, wide-area routing), we confirmed the behavior of routing management using DNS and LB methods, and compared switching time for varying parameters. We also measured the basic performance based on DNS and LB methods.

Vehicle movement:

- (1) Checking the behavior of routing management when using DNS or LB method
- (2) Effect of different parameters on switching time

Wide-area routing:

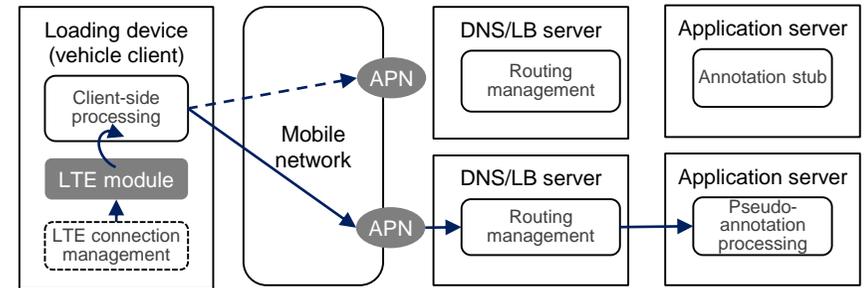
- (1) Management operation for varying failure patterns under DNS and LB methods
  - Total failure of application server
  - Partial failure of application server
  - Failure of control unit
- (2) Effect of different parameters on switching time

## Conditions

Vehicle movement: APN was switched by the vehicle client to confirm the behavior of routing management via BGP Anycast.

### Parameters

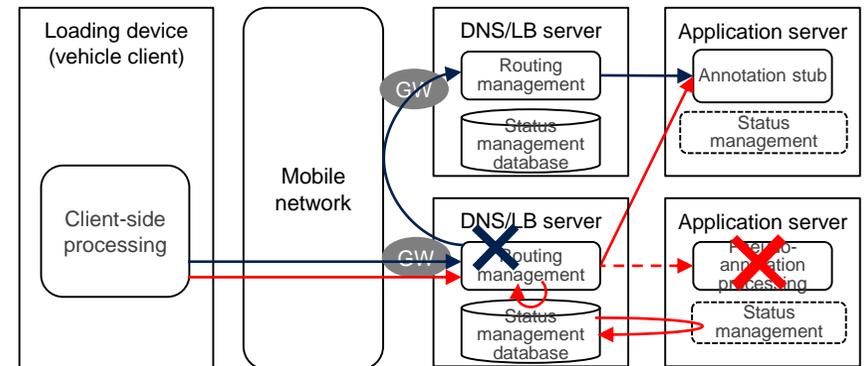
HTTP	1.1, 2.0
TLS	With/Without
Session continuity	With/Without
Data volume (KB/s)	16,256
TTL	0,60



Wide-area routing: Pseudo-equipment failures were created to confirm aggregated routing management with the status management database.

### Failure scenarios

Pattern	Assumed behavior
Total failure of Application server	Route management by switching to an alternative site
Partial failure of Application server	Route management by switching to an alternative Application server at the same site
Routing management failure	Switching to an alternative site by BGP Anycast



# Evaluation 1: Results

The evaluation results for each use case and basic DNS and LB methods performance are shown below.

## Vehicle Movement

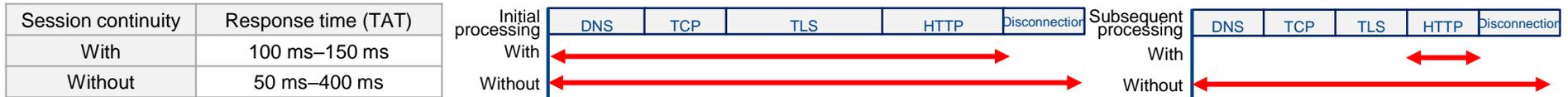
(1) Checking the behavior of routing management when using DNS or LB method

We confirmed operations in line with assumptions based on both methods. Switching time was around two seconds for either method, and we found that the difference in the routing management method had no effect on switching time.

(2) Effect of different parameters on switching time

We found that the difference in HTTP versions, TLS, session continuity, and data volume had no effect on switching time.

Response time was affected by the difference in parameters, and we confirmed that session continuity had a particularly significant effect.



## Wide-area Routing

(1) Management operation for varying failure patterns under DNS and LB methods

We confirmed operations in line with assumptions based on both methods. We found that the failure detection threshold (polling interval and timeout value) had a particularly significant effect on switching time.

(2) Effect of different parameters on switching time

We found that the difference in HTTP versions, TLS, session continuity, and data volume had no effect on switching time.

Processing time	DNS	LB
Routing processing	10 s–12 s	4–5 s
Timeout value	10 s	4 s

## Basic Performance

### DNS method

- Stable operational limit was around 10,000 queries per second
- We believe that the load on a specific CPU during UDP reception was the bottleneck
- While it is possible to process around 50,000 queries per second when large loads are received without interruption, we assumed it was not applicable in practice

### LB method

- Stable operational limit was around 10,000 queries per second
- We believe that the CPU load of TLS processing was the bottleneck
- Scale-up and tuning method are future issues

# Evaluations 2 and 3

We evaluated the routing management function based on metrics data and the basic performance of edge sites.

## Description

We confirmed the collection of metrics data during server failures and behavior of routing management based on LB method.

Verification of basic performance:

Confirmed the number of connections that can be processed by placing a connection load on a single edge site. We used a loading device for this verification.

Verification of routing management using metrics data:

Confirmed routing management operations and processing performance during the following failures under the connected load state measured in the basic performance verification.

### (1) Edge node failure\*

Confirmed distribution of processing to an alternative node within the same site and verify processing performance during distribution.

\*Edge node failure includes the following three patterns.

- LB server failure (power outage)
- Failure in load balancing function of the LB server
- Failure of BGP connection processes on the LB server which is used for BGP connection with Anycast router

### (2) Site failure

Confirmed distributed processing at multiple sites due to the following failures.

- Overload of application server
- Total site failure

## Conditions

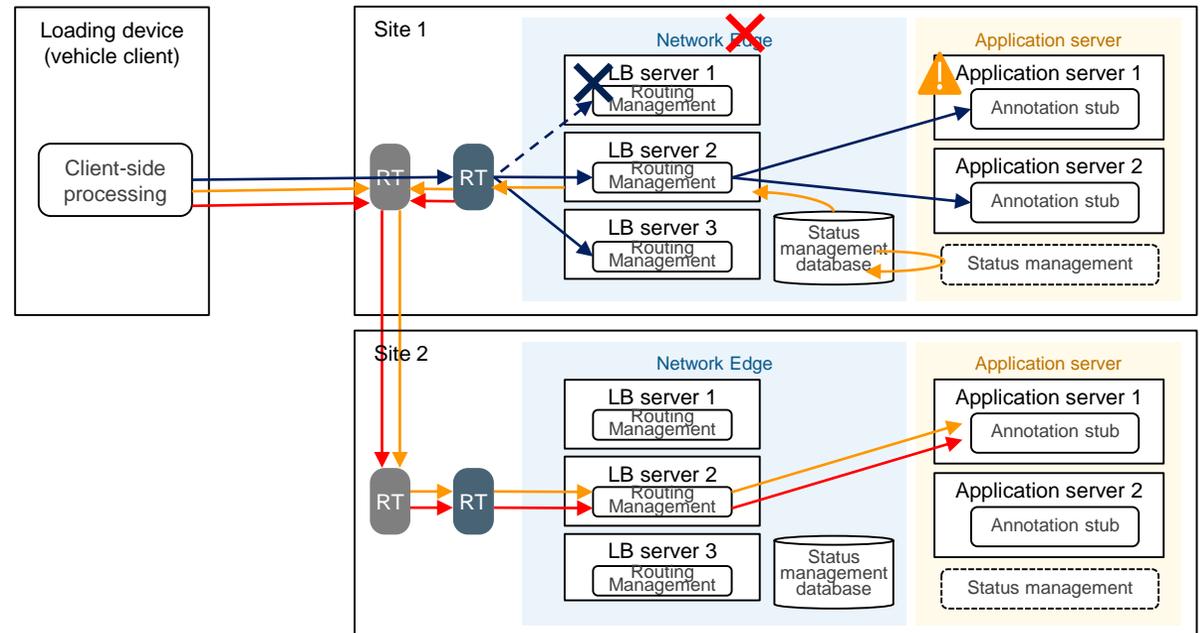
### Load criteria

Concurrent connections	200,000 TPS*
Data size	8 KB
Protocol	HTTPS (POST)

\*Concurrent connections are defined as the maximum processing performance at a single site. Based on the results of the basic performance verification, concurrent connections for verifying routing management using metrics data were set at 150,000 TPS.

### Assumed distributed operation during failures

→ Edge node failure	Processing is maintained by normal nodes within the same site
↔ Site failure	Processing is distributed to an alternative site
→ Overload	Part of the data traffic is distributed to an alternative site
→ Total failure	All data traffic is distributed to an alternative site



# Evaluations 2 and 3: Results

The results of the basic performance evaluation, node failure tests, and verification of routing management and load distribution at multiple sites are shown below.

## Basic Performance Evaluation

- The processing performance of a single site was concurrent connections of 150,000 TPS. Although our initial projection was a load of 200,000 TPS, around eight HTTP requests are being processed per TLS connection, which we believe is behind the reduced load.
- Average CPU usage was 25% and did not cause a bottleneck. Using the TLS session resumption, we conducted the verification during session start-up by setting new TLS connections at a maximum of 20,000 TPS. Therefore, CPU usage may cause a bottleneck depending on the proportion of new connections.

## Routing Management Based on Metrics Data

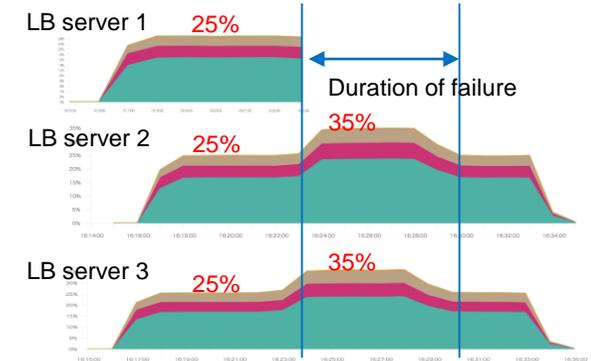
### (1) Edge node failure

- We confirmed that processing was maintained in each of the node failure cases by distributing to normal nodes, but we observed a phenomenon where processing was not possible for several seconds immediately following a power outage in Network Edge
- Compared to an average CPU usage of 25% under normal conditions, it rose to 35% during a failure of two units
- In terms of hardware specifications, 5% of CPU usage was for realizing network reception-transmission (Rx/Tx) of 1 Gbps
- We confirmed that the client resent data in response to a processing failure in the LB server. We must consider the possibility that this may increase the load on vehicle-mounted devices

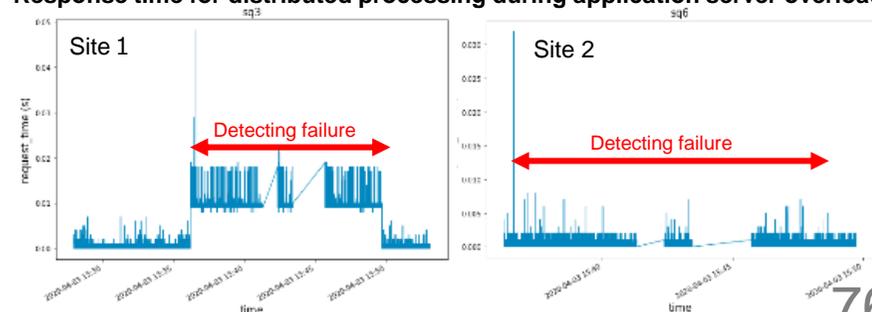
### (2) Site failure

- During an application server overload, we confirmed that some of the data traffic was allocated to an alternative site in line with our assumptions
- This added a delay of approximately 10 ms to the processing response time for transmissions between center sites
- We confirmed that all processing was distributed to an alternative site in response to a site failure
- We observed an increase in processing response time immediately after the switch

CPU usage during node failure



Response time for distributed processing during application server overload



# Evaluation 4

We conducted a trial run using test vehicles to confirm whether the set of scenarios, including the behavior of application groups such as annotation processing and vehicle notification, function as planned. We also verified the performance of wide-area distributed Message Queue (MQ) by uploading pseudo-data and placing a load on the client-side network connection.

## Description

Evaluated operation and basic performance of data transmission from vehicles and message notification to vehicles using network edge computing that incorporates wide-area distributed MQ.

**Operation:**  
Confirmed that we can establish communication between vehicles and application groups via the wide-area distributed MQ in line with the scenario by using test vehicles and the verification platform.

**Basic performance:**  
Used a load testing device to measure processing performance for data transmission from vehicles and message notification to vehicles as well as data arrival times during site-to-site offloading.

- (1) Data transmission from vehicles
  - a. Measured response performance and request processing performance from vehicle side
  - b. Measure data arrival time from LB server to application groups during site-to-site offloading
- (2) Message notification to vehicles  
Measured message arrival time from application groups to vehicles

## Conditions

The evaluation was conducted by implementing Network Edge servers and application groups incorporating wide-area distributed MQ in Tokyo and Osaka.

Scenario for the operation evaluation:

1. Processing by equipment at each site in Tokyo and Osaka
2. Offloading of the data transmission and notification based on assumptions of site failure and other incidents
3. Offloading only the notification based on the assumption of vehicle movement



\*Kudanshita area was selected as the trial route for the operation evaluation

Load criteria for basic performance evaluation:

(1)-a

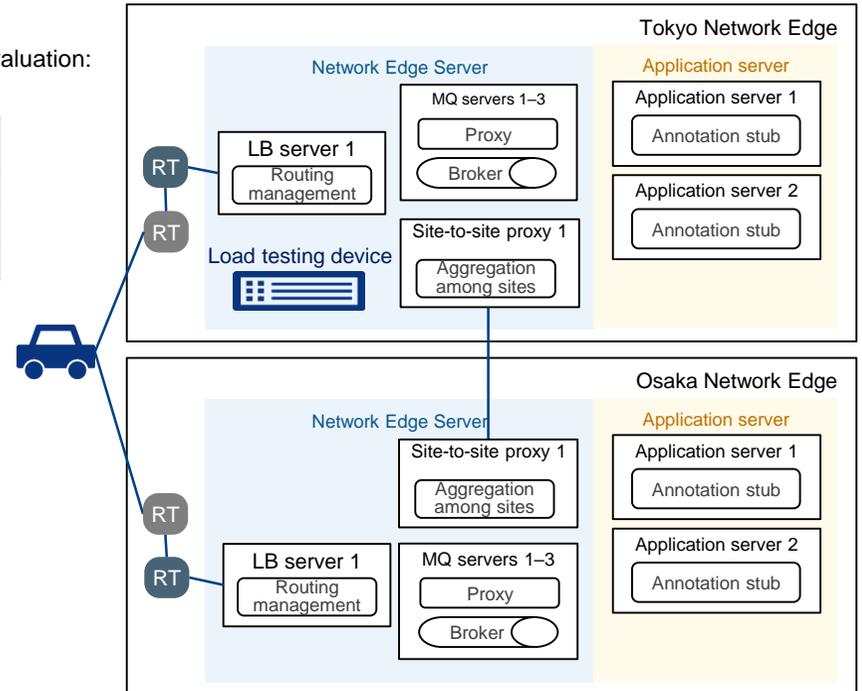
Total number of requests	200,000
Number of clients	200
Message size	128 B-1 MB
Protocol	HTTP

(1)-b

Number of Instantaneous requests	1-2,000
Message size	128 B-1 MB
Protocol	HTTP

(2)

Number of Instantaneous requests	1-1000
Message size	128 B



# Evaluation 4: Results

The results of the operation evaluation for wide-area distributed MQ and basic performance evaluation are shown below.

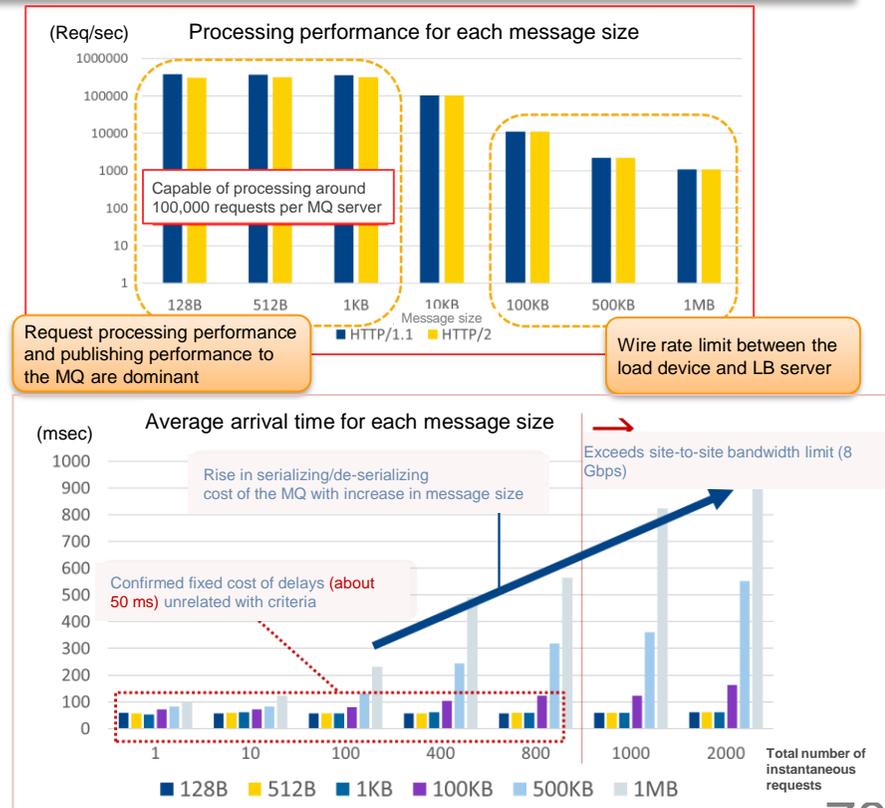
## Operation Evaluation of Wide-Area Distributed MQ

We verified that processing was completed normally in all evaluation scenarios from 1 to 3 within the trial system comprising vehicles, the Tokyo and Osaka edge sites and center site.

No.	Pattern	Test result
1	Processing by equipment at each site in Tokyo and Osaka	Good
2	Offloading of the data transmission and notification based on the assumptions of site failure and other incidents	Good
3	Offloading only the notification based on the assumption of vehicle movement	Good

## Basic Performance Evaluation

- (1) Data transmission from vehicles
  - (1)-a Measurement of response performance and request processing performance from the vehicle side
    - Confirmed the possibility of processing 100,000 requests per second per MQ server when the message size is up to 1 KB
    - Observed no difference in results between HTTP/1.1 and HTTP/2
    - The wire rate limit became dominant once the message size exceeded 100 KB
  - (1)-b Measurement of data arrival time from the LB server to application groups during site-to-site offloading
    - Offloading resulted in a minimum delay of 50 ms as overhead
    - With the same total data transmission and the message size exceeds 100 KB, any increase in the number of instantaneous requests was accompanied by a marked increase in average arrival time  
(e.g., Average arrival time is lower when sending the same number (1,000) of 100 KB requests as 1 MB requests)
- (2) Message notification to vehicles
  - We confirmed that even scaling up the number of simultaneously connected clients in the logic queue to around 1,000 had a limited effect on the arrival time of the notification message.



# Evaluation 5

As described in the basic performance verification of Evaluation 1, CPU processing may become a bottleneck that lowers performance when there is an increase in simultaneous connections to an edge site. We verified the enhancement of the processing performance based on hardware offloading.

## Description

We evaluated the degree of enhancement in processing performance for routing management that can be realized by introducing hardware offloading. We also confirmed CPU usage.

### Expected improvement in performance

We hoped to raise processing performance while lowering CPU usage by adopting hardware offloading.

## Conditions

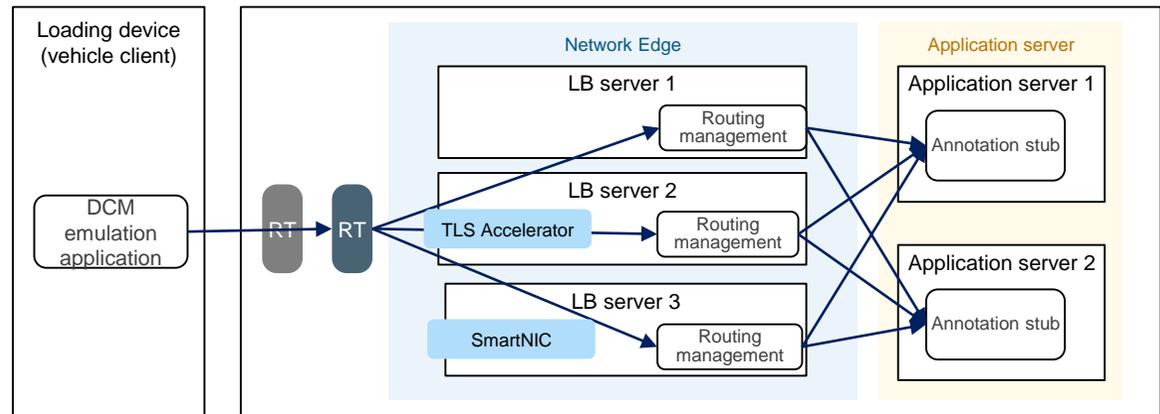
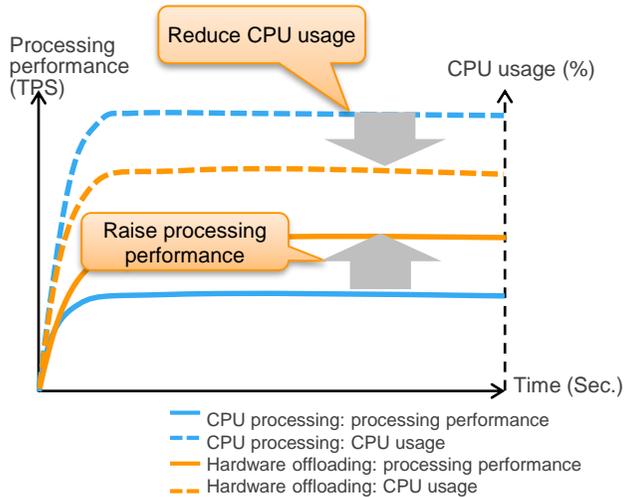
### Load criteria

Maximum number of simultaneous connections per server	22,000 vehicles*
Data size	8 KB
Application protocol	HTTPS (POST)
TLS	TLS: v1.2; Cipher: ECDHE-RSA-AES128-SHA256

\*Set the load scenario for the loading device simulating vehicle-mounted client packets for 22,000 vehicles

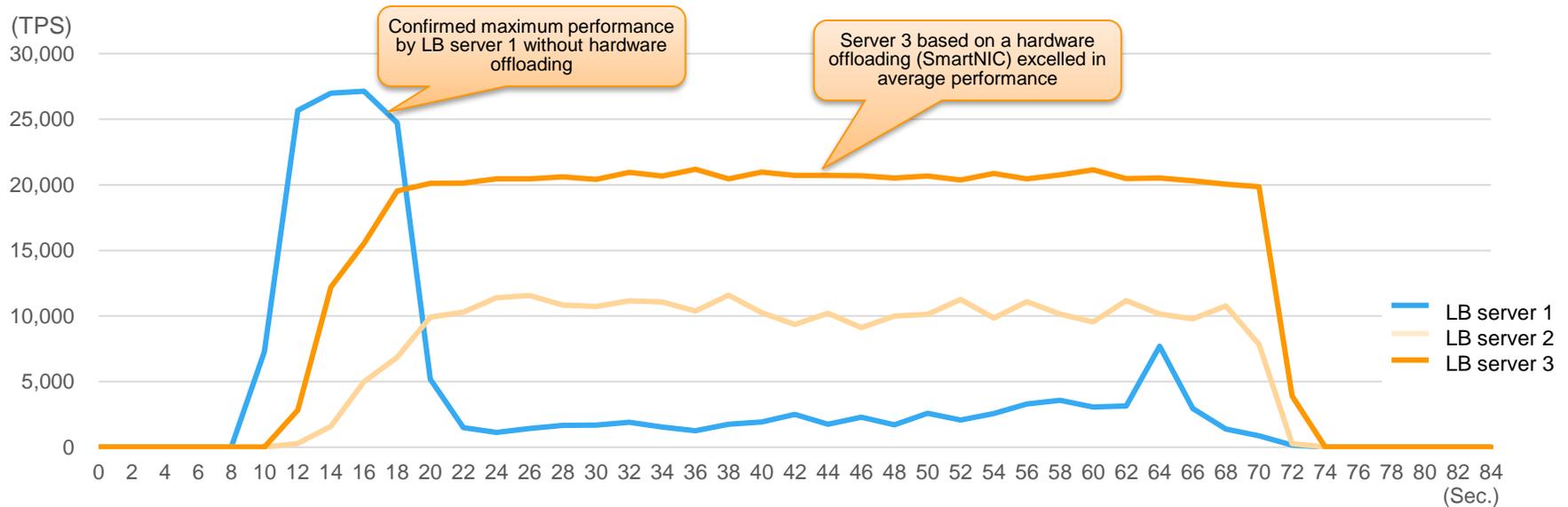
### Environment composition

	Hardware offloading mechanism	OpenSSL version
LB server 1	None	OpenSSL 1.1.1f
LB server 2	TLS Accelerator installed	OpenSSL 1.1.1f
LB server 3	SmartNIC (ASIC) installed	OpenSSL 3.0.0-alpha13



# Evaluation 5: Results

We measured the performance and CPU usage of each edge server. While servers 2 and 3 incorporating hardware offloading demonstrated stable processing, the maximum performance was far below that of server 1 without hardware offloading. We believe we were unable to confirm the expected performance enhancement because verification was conducted based on processing packets with small volumes of data that would not benefit from hardware offloading.



	LB server 1	LB server 2	LB server 3
Avg. CPU (%)	75.87 / 11.15*	19.79	77.57
Avg. transaction rate (TPS)	5,977.57	9,152.61	<b>18,989.47</b>
Max. transaction rate (TPS)	<b>26,993</b>	11,599	21,182

With LB server 3 (offloading mechanism: SmartNIC), verification was conducted only for Tx Offloading. Rx Offloading is a model that leaves processing completely to the CPU, and we assume we can reduce CPU usage by using this model.

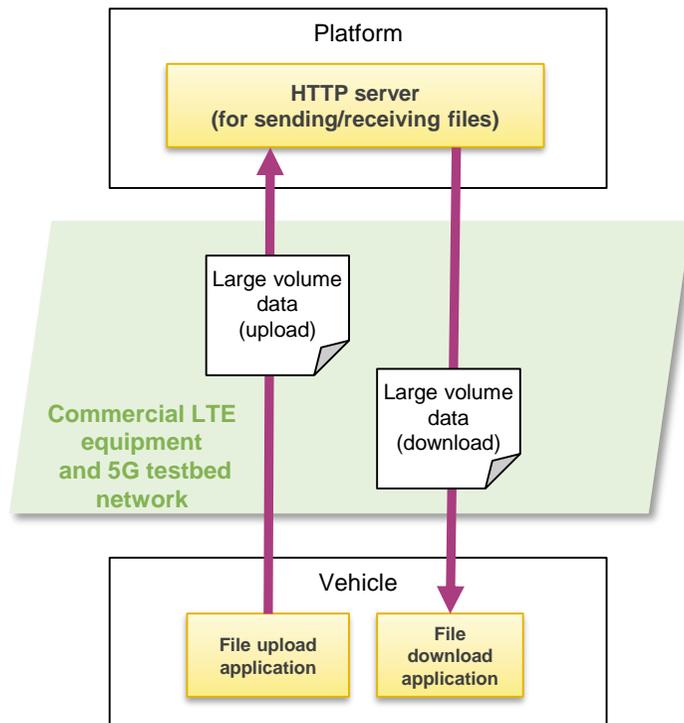
\*CPU usage for LB server 1 was calculated separately as an average value for the earlier part of the measurement when it peaked and for the latter part of measurement when it plateaued. Sections from 0 s to 8 s and beyond 74 s, where TPS is 0, have not been included in the calculation.

# Evaluation 6

We measured performance values for uploading and downloading large volumes of data between vehicles mounted with LTE/5G modems and the platform.

## Description

We established connections for communicating large volumes of data between vehicles and the platform and upload/download large files and measured the processing time and throughput until HTTP POST/GET was completed for each transmission.



## Conditions

Wireless environment

- 5G testbed network
  - Experimental equipment for handling different environments and settings than commercial models
  - 28 Ghz band 400 Mhz width
- LTE communications network
  - Commercial equipment with different execution speeds depending on the place and time of use and communication environment

Limitations

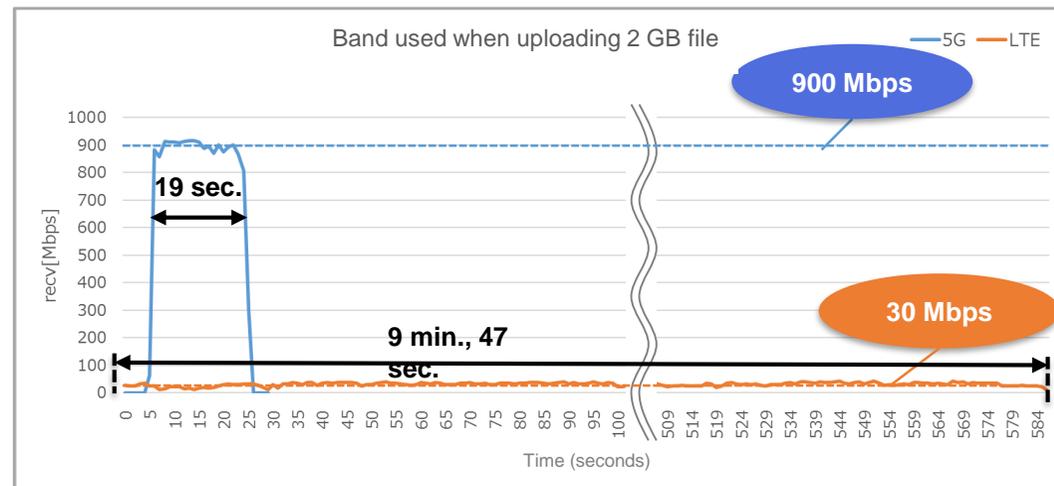
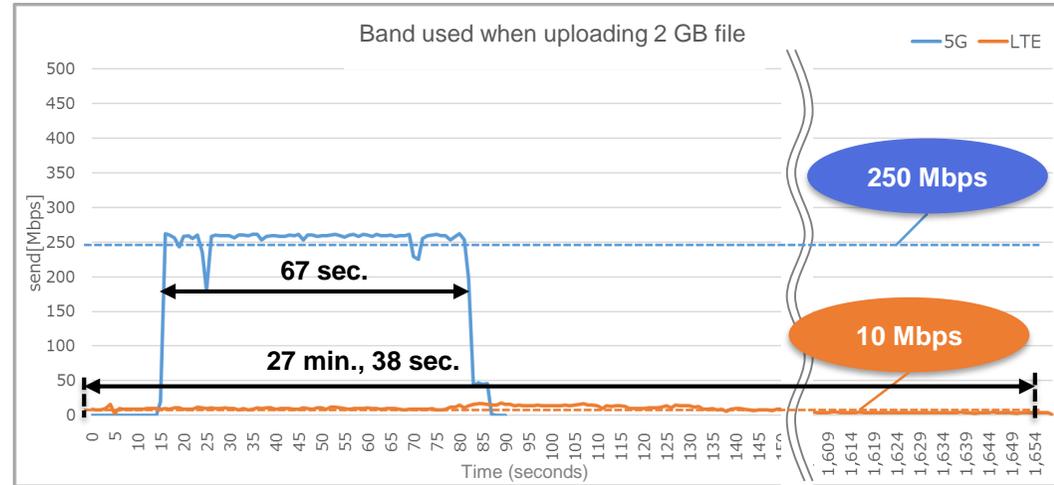
- Size of large volume files: 128 MB to 2 GB
- Assume that files were uploaded/downloaded under circumstances where CAN data is being regularly sent from vehicles
- Vehicles moved within the 5G trial site at Odaiba, where verification of LTE was also conducted
- We could not verify the communication handover due to equipment limitations

# Evaluation 6: Results

We connected the vehicles with the 5G testbed network and obtained basic performance values at regular points on the field. As a result, we confirmed that using 5G broadband significantly shortened both upload and download times compared to LTE.

## Performance result for sending 2 GB data\*1

	Upload		Download	
	5G testbed network	Reference: commercial LTE network	5G testbed network	Reference: commercial LTE network
<b>Network throughput</b>	Average value: 243.61 Mbps Maximum value: 262.25 Mbps	Average value: 9.88 Mbps Maximum value: 20.69 Mbps	Average value: 861.87 Mbps Maximum value: 914.93 Mbps	Average value: 27.91 Mbps Maximum value: 44.42 Mbps
<b>Processing time</b>	Average value: 67 s	Average value: 27 m, 38 s	Average value: 19 s	Average value: 9 m, 47 s



\*1 Results are values obtained in an experimental environment, which is different than a commercial 5G network.

# Summary

We implemented an architecture between the vehicles and application groups that consists of an arrangement of Network Edges with multiple functions (routing management, protocols, queueing, site-to-site offloading, vehicle notification management, and status management) and conducted trials to confirm its effectiveness. The results of each verification and future issues that came to light are summarized below.

Evaluation Points	Result Summary and Considerations	Future Issues
1. Optimal routing management for sending data from vehicles to center site and application servers	<p>We evaluated routing management behavior during vehicle movement and wide-area routing for DNS and LB methods and also assessed the basic performance of the two methods.</p> <ul style="list-style-type: none"> <li>• We verified that routing management can be executed using either method without issues</li> <li>• There was no major difference in switching time between DNS method and LB method</li> <li>• We verified the possibility of CPU becoming the bottleneck that lowers performance</li> </ul>	<p>It became apparent that we needed to consider methods for scaling up, scaling out, and tuning, so we addressed the issue while evaluating “2. Metrics-based routing management.”</p>
2. Metrics-based routing management	<p>We evaluated the wide-area routing behavior and processing performance of edge sites for routing management based on LB method in case of overload and site failure.</p> <ul style="list-style-type: none"> <li>• Verified that processing performance was 150,000 TPS for a single site.</li> <li>• In this verification we did not observe any performance caused by CPU bottleneck.</li> <li>• Processing was maintained in each of the node failure cases by distributing to normal nodes.</li> <li>• We observed cases where messages were resent at the time of the switch.</li> <li>• During a site failure (overload, malfunctions), we verified that allocating processing to an alternative site raised processing response.</li> </ul>	<p>Advanced metrics monitoring</p> <ul style="list-style-type: none"> <li>• In this evaluation, routing was managed based on the monitoring status and Round-Trip Time (RTT) of the target servers. We must conduct tests in real environments to verify basic technologies that will enable the use of AI to analyze accumulated data for predicting congestion and providing navigation when congestion is detected.</li> </ul>
3. Performance evaluation of TLS termination (CPU processing)		
4. Message Queue (MQ)-based data flow between edge sites	<p>We evaluated behavior and processing performance for transmitting messages and notifications, including aggregation among sites based on wide-area distributed MQ that responds to vehicle movement.</p> <ul style="list-style-type: none"> <li>• We observed site-to-site aggregation and message notification based on wide-area distributed MQ in line with our hypothesis.</li> <li>• When sending messages, we were able to process approximately 100,000 requests per server.</li> <li>• We observed a delay (50 ms) in the arrival of messages to the application caused by aggregation.</li> <li>• The effect on arrival time for message notification was limited.</li> </ul>	<p>Aggregation with priority management of application servers</p> <ul style="list-style-type: none"> <li>• We mainly evaluated routing management based on the use case of vehicle movement and equipment failure. In the future, we must consider aggregation with priority management based on data processed by application servers to assess advanced management methods that enable selecting high priority queues for higher priority data.</li> </ul>
5. Verification of TLS offloading	<p>We evaluated reduced CPU usage based on hardware offloading and operational performance.</p> <ul style="list-style-type: none"> <li>• We verified that hardware offloading can lower CPU usage and raise performance but could not confirm any advantages for ensuring maximum performance.</li> <li>• We verified that SmartNIC demonstrated superior performance for offloading.</li> </ul>	<p>Rx Offload by SmartNIC and expansion in data size</p> <ul style="list-style-type: none"> <li>• In this evaluation, there was only the Tx Offload function when testing SmartNIC. In the future, we must address Rx Offload and conduct additional evaluations. We will also need to verify reception of large volume data (from around 500 KB to 1 MB) assuming image and video data sent from vehicles.</li> </ul>
6. 5G field verification	<p>We conducted upload and download tests for large volume data via LTE and 5G networks and confirmed that 5G is capable of handling extremely large volumes of data at a high speed.</p>	—

# Future Initiatives

Since December 2018, we have been conducting verification tests to evaluate feasibility of typical services in the connected vehicle field and ICT platform technology.

As a result, we achieved the following two objectives, as well as our performance targets concerning the number of connected vehicles and real-time response.

At the time of the spread of connected vehicles and autonomous vehicles,

- Evaluation of cloud platform technology/communications infrastructure technology
- Requirements arrangement for next-generation vehicles

In preparation for further expansion of connected vehicles, we will work on followings and plan to continue developing technologies that can solve various social issues facing our society, such as traffic accidents and congestion.

- Improvement of ICT platform for connected vehicles
- Effective utilization of big data from connected vehicles

In the future, we will advance technological development in collaboration with various companies, organizations and services in order to provide communications and computational resources beyond the limits of current infrastructure.

In addition, we aim to realize a sustainable smart mobility society that brings safety and security in the coming autonomous driving era.



# Trademarks

The owners of the registered trademarks or trademarks related to company names and product names mentioned in this document are as follows.

- AMD, the AMD Arrow logo, ATI and a combination of these, as well as Radeon are trademarks of Advanced Micro Devices, Inc.
- Amazon, Amazon Web Services, AWS, Amazon EC2 and Amazon S3 are trademarks of Amazon.com, Inc. and its affiliated companies in the United States and other countries.
- Apache, Tomcat, Apache Hadoop, Apache Ignite, Apache Zookeeper, Apache Kafka and Apache Spark are registered trademarks or trademarks of Apache Software Foundation in the United States and other countries.
- Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and other countries.
- Elasticsearch is the registered trademark or trademark of Elastic NV in the United States and other countries.
- Hadoop and Spark are registered trademarks or trademarks of Apache Software Foundation in the United States and other countries.
- Java is the registered trademark of Oracle Corporation, its subsidiaries and affiliated companies in the United States and other countries.
- Kubernetes is the trademark or registered trademark of The Linux Foundation in the United States and other countries.
- Linux is the registered trademark or trademark of Linus Torvalds in Japan and other countries.
- Microsoft, Windows, Windows Aero, Internet Explorer, Office, OneNote, Outlook, Excel, PowerPoint, Windows Live, Windows Media and the Windows logo are trademarks or registered trademarks of U.S. Microsoft Corporation in the United States and other countries. Screen images are being used in accordance with guidelines published by Microsoft Corporation.
- Oracle is the registered trademark of Oracle Corporation and its affiliated companies.
- PostgreSQL is the trademark or registered trademark of PostgreSQL in the United States and other countries.
- Redis is a trademark of Salvatore Sanfilippo.
- TensorFlow is the registered trademark or trademark of Google Inc. in the United States and other countries.
- Ubuntu is the registered trademark of U.K. Canonical Ltd.
- Intel, Intel Inside, the Intel Inside logo, Intel SpeedStep, Intel Core, Intel vPro, vPro Inside, Intel Atom, Celeron and Pentium are the trademarks or registered trademarks of Intel Corporation and its subsidiaries in the United States and other countries.

Other company names and product names are the trademarks or registered trademarks of the respective company.

# Appendix

# About ICT Technologies

# About ICT Technologies (1/4)

## Appendix 1: High-performance spatial-temporal data management technology and high-performance spatial-temporal query technology<sup>(1)(2)(3)(4)</sup>

These technologies enable massive volumes of vehicle data to be stored instantaneously, allowing a high-performance query to be carried out within certain parameters (time and location). Specifically, this spatial-temporal database manages time and positioning information generated by moving objects including vehicles and smartphones as one-dimensional data using a unique method and allows querying for a vehicle within a given timeframe and location in addition to high-speed queries for roads, parking, and other complex-shaped spaces. A high-performance query for sensor information contained in a given rectangular range defined by certain time and spatial information is made possible by storing sensor data generated by massive volumes of dynamic objects in a real-world physical space along with associated time and spatial information. Furthermore, this high-performance spatial-temporal query technology can be built upon to manage more advanced spatial-temporal data, including queries for ranges that are more complex than simple rectangular shapes such as roads. For example, this enables the high-speed extraction of vehicles and smartphones (i.e., people) in locations expressed by a polygon shape such as a road, park, or school district.

## Appendix 2: Data acquisition technology with target vehicle selection algorithm

This technology prioritizes the order of data aggregation based on meta-information such as position data, time information, sensor type, resolution, observation range, and situations in which vision may be blocked by surrounding vehicles. Data aggregated from connected vehicles vary from vehicle data, video data, and image data to sensor data, all in massive volumes. These data collected from a series of vehicles adds up to a tremendous volume. Increase in data traffic raises concerns over the heavy load on network and data processing. This technology facilitates the efficient aggregation of data by minimizing the overlap of observation range as well as adjustments in the amount of data to be aggregated depending on the load condition of the telecommunications channel and analysis platform.

# About ICT Technologies (2/4)

## Appendix 3: Congestion detection by lane

This technology analyzes footage taken by vehicle-mounted cameras and driving data such as vehicle speed and position information to detect vehicle congestion per lane. By inferring the section of the lane where traffic may be congested, vehicles at the front and the back end of the queue are identified, and queue length is calculated from their positions. This technology can be applied to footage from monocular cameras such as inexpensive drive recorders. Current traffic congestion information is provided on a road-by-road basis by the Vehicle Information and Communication System (VICS) or a map app. If the technology is put to practical use or becomes widespread, drivers will be able to obtain more precise congestion information. Furthermore, the cause of road congestion can be inferred by combining the footage of the area around the vehicle at the front of the queue and its position data with the position data of landscape features such as surrounding facilities and traffic lights.

## Appendix 4: Vertical-distributed computing technology

This technology realizes high-speed processing of applications by distributing the processing functions formerly carried out at data centers to edge servers. Striving to have all devices immediately respond to inputs may from time to time hinder the realization of a high-speed response as a result of insufficient server resources. In practice, however, the necessity of a high-speed response differs depending on the vehicle's overall situation (speed, direction, time period, etc.). This technology is intended to efficiently deploy limited server resources by dynamically changing the server that performs the response process depending on vehicle circumstances while dynamically distributing the app to ensure high-speed responses to vehicles that need them. This approach will help us more efficiently use the limited resource of distributed computing capable of accommodating many vehicles and performing aggregation and responding to massive volumes of data.

## Appendix 5: Suddenness Index Calculation Method

This technology creates an index at a high speed and based on the divergence of periodicity and/or suddenness of the value aggregated in the chronological log from its stationary level. It learns the average and distribution of aggregates such as the number of vehicles on different temporal axes (all period, day of the week, time of the day, day of the week × time of the day) and creates an index that represents the level of suddenness of the newest aggregate. The suddenness index by time axis will be integrated and unified using a weighted linear combination and taking into account the number of times it was aggregated. A sudden increase in the number of aggregated vehicles will be detected as a potential place for sudden traffic congestion. To optimize traffic flow, this technology should be applied before (3) Congestion detection by lane, as communication cost cuts and server load reductions can be expected, by prioritizing the area for retrieving footage or by controlling the frequency of retrieval.

# About ICT Technologies (3/4)

## Appendix 6: Object-position-estimation technology<sup>(5)</sup>

This technology estimates the position coordinates of landscape features along the road by triangulation using footage from inexpensive, commercially available monocular drive recorders and GPS. It improves the precision of position estimation by selecting quality data from the large volume of aggregated footage and data and by statistically and automatically selecting estimation results. Accurate recognition of vertical landscape features such as road signs, signboards, and traffic lights, and high-precision estimates of their positions, helps the plotting of information onto maps. If the technology is put to practical use or becomes widespread, we can expect the creation of high-precision maps that reflect the most recent road conditions for automated driving or a cost reduction in relation to updating information about landscape features.

## Appendix 7: Video chunking transfer technology

This technology enables the transfer of footage taken by vehicle-mounted cameras in units equaling to or smaller than one second (at the extreme end, it can be transferred in a single-frame unit). Since the minimum length of video data transferred using a conventional technology is one second, waiting times of one second or more is inevitable at the platform before it can start processing. Applying this technology can accelerate the process by the shortened waiting time. Additionally, note that CAN data linked to video data can be generated by a unit of one second each and can be sent as it is generated by transmitting it in parallel.

# About ICT Technologies (4/4)

## Bibliography

- (1) A. Isomura, “Real-time spatiotemporal data utilization for future mobility services,” RedisConf19, June 2019
- (2) RedisConf19 presentation  
(<https://www.slideshare.net/RedisLabs/realtime-spatiotemporal-data-utilization-for-future-mobility-services-atsushi-isomura>)
- (3) RedisConf19 video (<https://www.youtube.com/watch?v=jVnlkwpcL3U>)
- (4) NTT GIJUTU Journal ([https://www.ntt.co.jp/journal/1911/JN20191118\\_h.html](https://www.ntt.co.jp/journal/1911/JN20191118_h.html))
- (5) Aki Hayashi, Yuki Yokohata, Takahiro Hata, Kouhei Mori, and Kazuaki Obana, The Road Environment Measurement for Automatic Road Map Generation, 2021, Volume 52, Issue 2, pp. 419–424, published in 2021, released on February 19, 2021

# Trial Routes for Verification

# Trial Routes for Verification

Routes were set in the Odaiba area for use case verification. The route shown in red below was set as the basic route, and the trial was extended to the routes shown by broken gray lines as necessary.

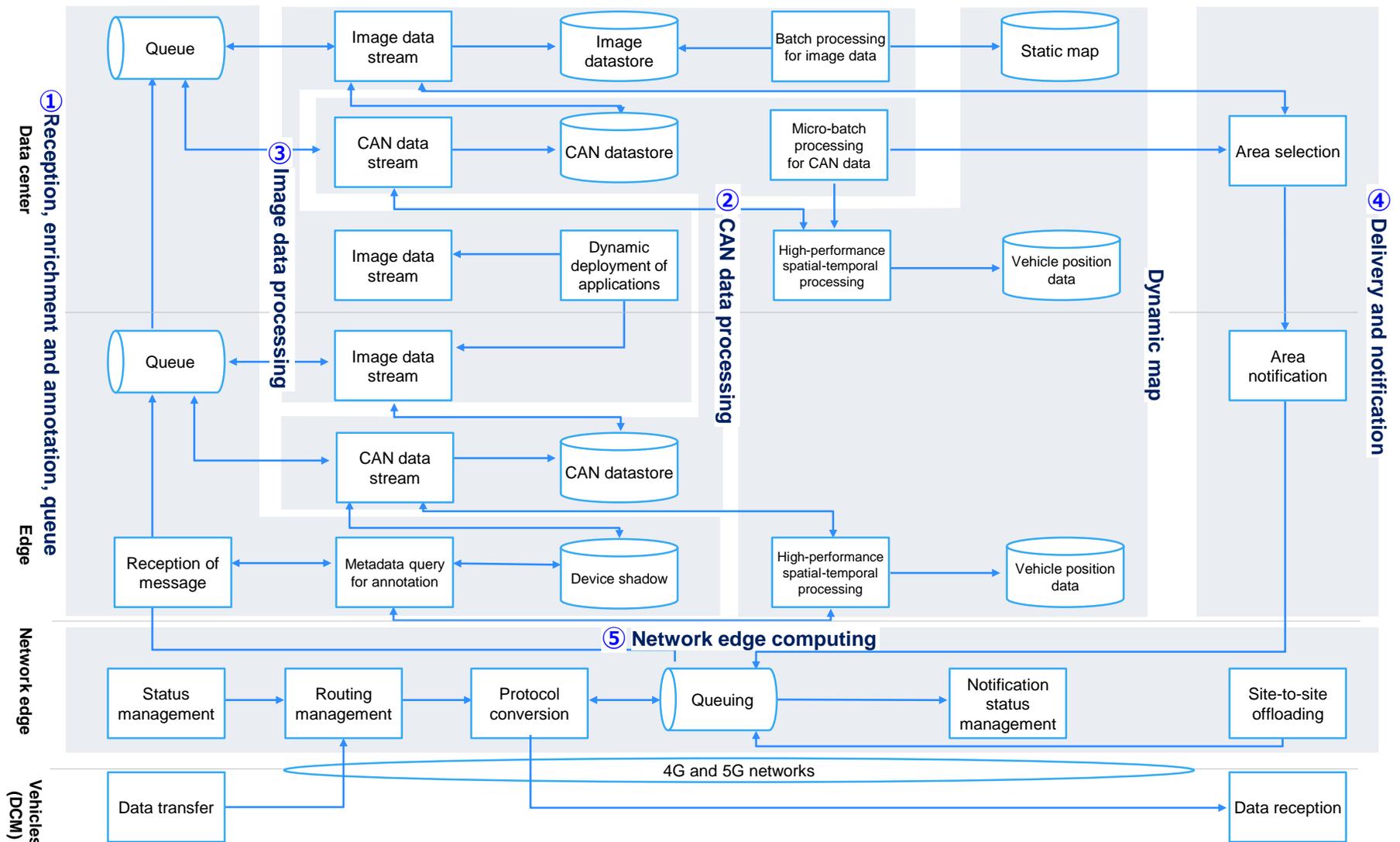


© OpenStreetMap contributors

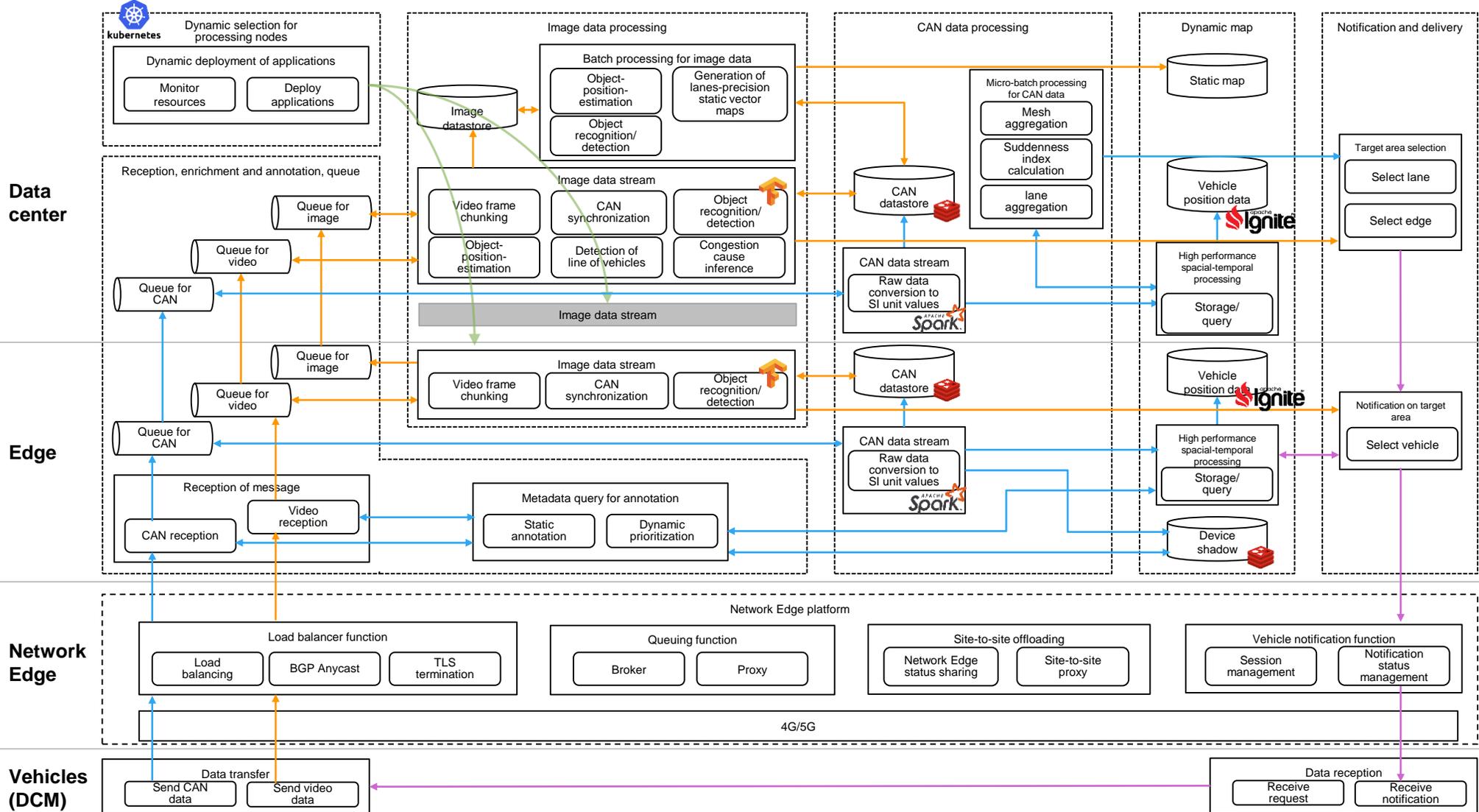
# Platform Verification System

# Architecture

Platform verification was performed for each platform of ① to ⑤ shown below.



# Architecture (Detailed Version)



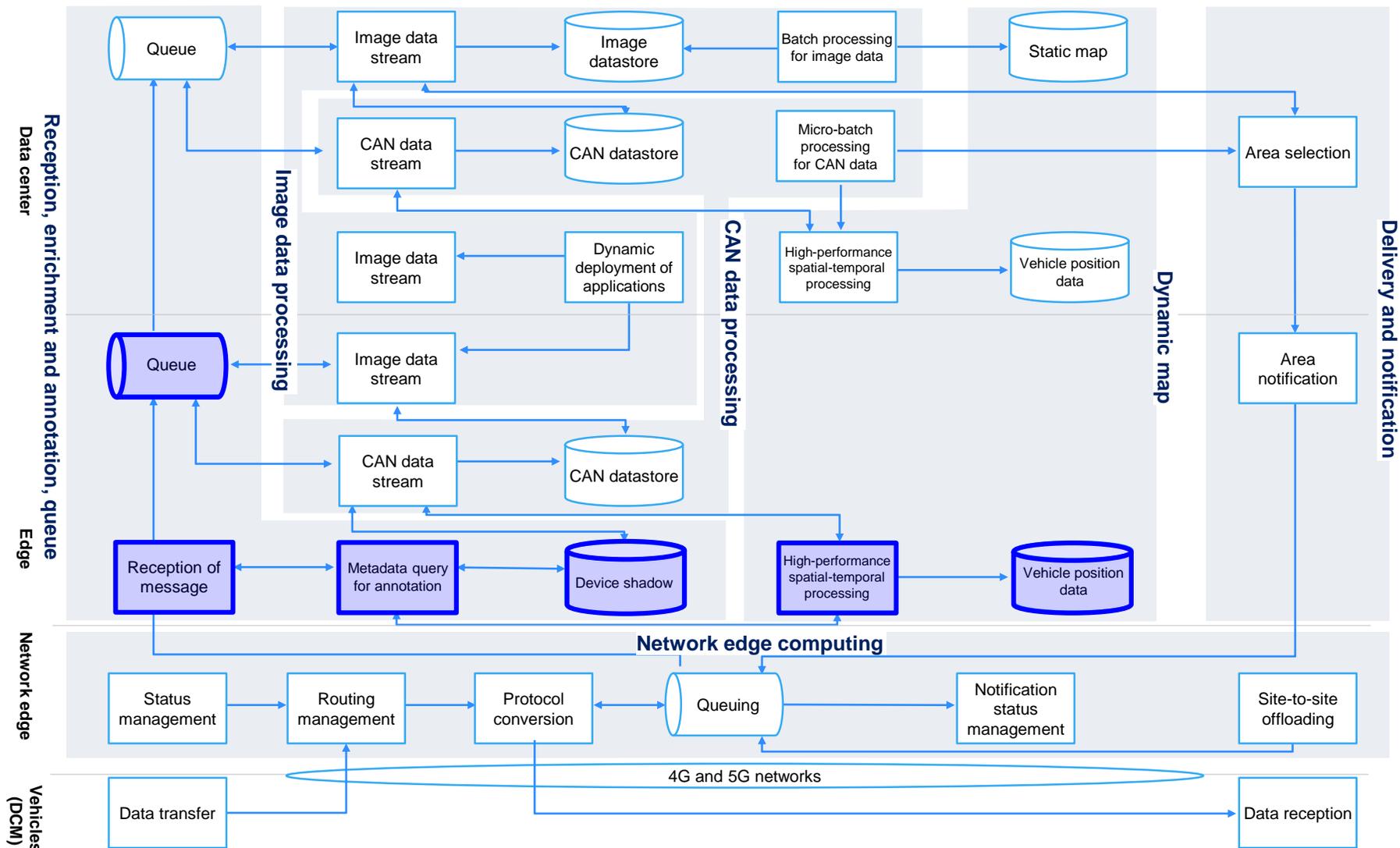
# List of Software

List of software used in the platform verification.

No.	Name of software	Description	Reception, enrichment and annotation, queue	CAN data processing	Image data processing	Dynamic map	Notification/delivery	Network edge computing
1	CentOS	OS	○	○	○	○	○	—
2	OpenJDK	Java operating environment	○	○	○	○	○	—
3	Netty	Framework wrapping Java NIO	○	—	○	—	○	—
4	Apache Kafka	Middleware for processing distributed messaging	○	○	○	—	—	—
5	NATS	Middleware for processing distributed messaging	—	—	—	—	—	○
6	Apache HDFS	Distributed file system	—	○	—	—	—	—
7	Apache YARN	Distributed resource management mechanism	—	○	—	—	—	—
8	Apache Spark	Parallel distribution processing engine	—	○	—	—	—	—
9	Redis	Distributed in-memory KVS	—	○	—	—	○	—
10	Apache Ignite	Distributed in-memory database	—	—	—	○	—	—
11	Elasticsearch	Distributed query/analysis engine	—	—	—	—	○	—
12	UltraMonkey	Software load balancer	○	—	○	—	—	—
13	Docker	Platform providing container-type virtualization environment	—	—	○	—	○	—
14	Kubernetes	Middleware for integrated management of containers running on multiple nodes	—	—	○	—	○	—
15	Hitch	TLS decoding middleware	○	—	○	—	○	—
16	PostgreSQL	Relational Database Management System (RDBMS)	—	—	—	○	—	—
17	PostGIS	Extension to handle geospatial information using a PostgreSQL database	—	—	—	○	—	—
18	Nginx	Used as a load balancer for web server software	—	—	—	—	—	○
19	OpenSSL	SSL/TLS communication library	—	—	—	—	—	○
20	FRRouting	Routing protocol management software	—	—	—	—	—	○

# Architecture

Reception, enrichment and annotation, and queue platform verification components are highlighted in blue.



# List of Functions

Functions of the reception, enrichment and annotation, and queue component

No.	Category	Name of function	Description
1	Reception of message	Receive CAN and video data	Receives message by HTTP protocol
2	Metadata query for annotation	Query fixed metadata for annotation, fixed/dynamic method selection	<ul style="list-style-type: none"> <li>(1) Refer to the annotation master and retrieve the initial value of annotation information</li> <li>(2) Refer to the data type of the message and process area detection if it is a video message of an event</li> <li>(3) Return the retrieved annotation information if prioritization will not be performed</li> </ul>
3		Area detection	<p>Prioritizes event video messages in the following sequence.</p> <ul style="list-style-type: none"> <li>(1) Retrieves a list of vehicles present in the surrounding mesh</li> <li>(2) Narrows down the data by calculating the average moving speed, etc.</li> <li>(3) Detects coverage</li> <li>(4) Detects the area threshold</li> </ul>
4		Output data queue	Sends messages to queue for CAN and queue for video based on annotation information
5	Queue	Queue for CAN, queue for video	Temporarily stores an annotated message

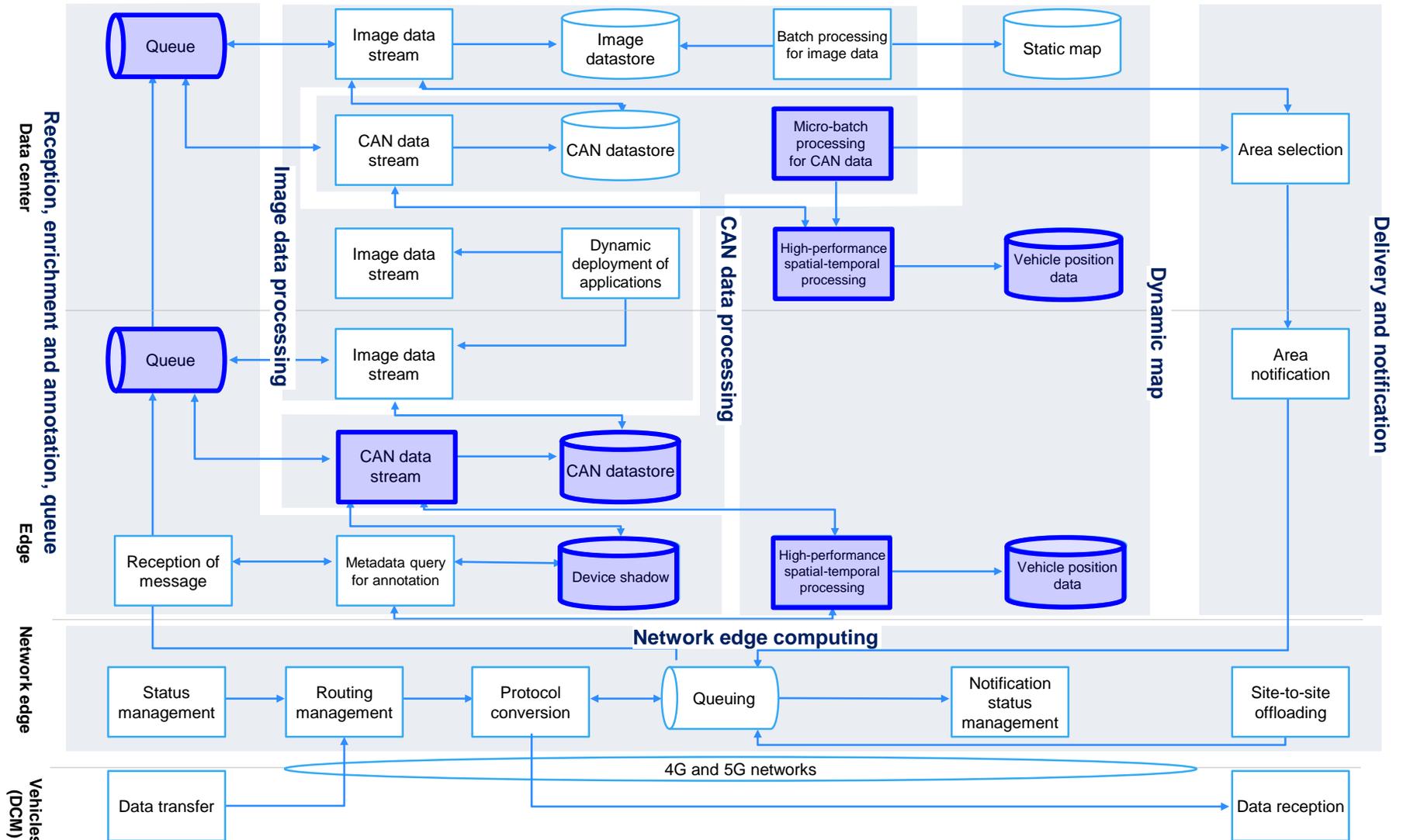
# List of Hardware

List of hardware used in the reception, enrichment and annotation, and queue component.

No.	Name of server	Name of function	Number of servers	Server specs
1	Generator server	Send CAN and video data	25	CPU (Xeon Gold 5118): 2 P, 24 C, 48 T Memory: 192 GB Disk (HDD): 1,200 GB
2	Reception, enrichment and annotation server	Receive CAN and video data, query fixed metadata for annotation, fixed/dynamic method selection, area detection, and output data queue	6	CPU (Xeon Gold 5118): 2 P, 24 C, 48 T Memory: 192 GB Disk (HDD): 1,200 GB
3	Device shadow	—	1	CPU (Xeon Gold 5118): 2 P, 24 C, 48 T Memory: 192 GB Disk (HDD): 1,200 GB
4	Vehicle position data	—	3	CPU (Xeon Gold 5118): 2 P, 24 C, 48 T Memory: 192 GB Disk (HDD): 1,200 GB
5	Queue server	Queue for CAN, queue for video	9	CPU (Xeon Gold 5118): 2 P, 24 C, 48 T Memory: 288 GB Disk (SSD): 1,600 GB

# Architecture

Components used in the CAN data processing platform verification are highlighted in blue.



# List of Functions

## Functions of the CAN data processing component

No.	Category	Name of function	Description
1	CAN data stream	Raw data conversion to SI unit values	(1) Retrieves message stored in data queue (2) Filters (3) Interprets binary data contained in the message and converts it into data with high readability
2		Vehicle position registration	Output data to the vehicle position data and device shadow after performing raw data conversion to SI unit values
3		Output CAN datastore	Stores in the event store the data sent from CAN at the time of the event and after performing raw data conversion to SI unit values
4	Micro-batch processing for CAN data (aggregation process)	Mesh aggregation	Retrieves vehicle information in the designated mesh and calculates the number of vehicles
5		Suddenness index calculation	Determines if the number of vehicles calculated by mesh aggregation has suddenly increased compared to the usual state within the mesh
6		Lane aggregation	Calculates the average moving speed of vehicles in the lanes within the mesh by using a suddenness index calculation and picks up those that were below the threshold as a potentially congested lane
7	Queue	Queue for CAN, queue for potential congestions	Temporarily stores annotated CAN message and the results of congestion detection

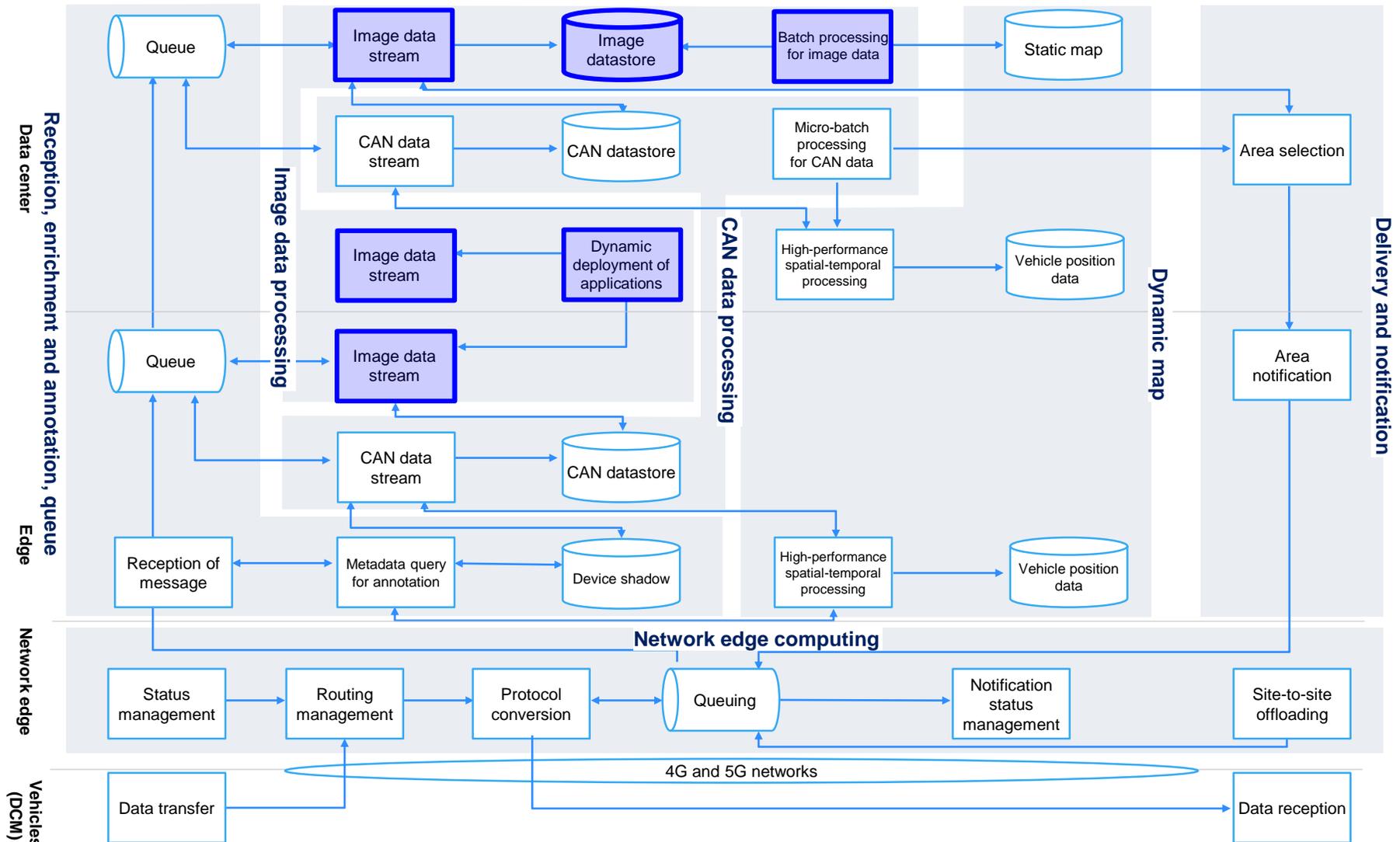
# List of Hardware

List of hardware used in the CAN data processing component.

No.	Name of server	Name of function	Number of servers	Server specs
1	Queue server	Queue for CAN, queue for potential congestions	8	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 288 GB Disk (SSD): 1,600 GB
2	CAN data stream server	Raw data conversion to SI unit values, vehicle position registration, output CAN datastore	a: 5 b: 15	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 192 GB Disk: 1,200 GB
3	Device shadow	—	1	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 192 GB Disk (HDD): 1,200 GB
4	Vehicle position data	—	3	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 192 GB Disk (HDD): 1,200 GB
5	CAN datastore	—	1	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 192 GB Disk: 1,200 GB
6	Micro-batch processing for CAN data server	Mesh aggregation, lane aggregation	3	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 192 GB Disk: 1,200 GB
7	Suddenness index calculation server	Suddenness index calculation	1	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 288 GB Disk: 1,600 GB

# Architecture

Components used in the verification of the image data processing platform are highlighted in blue.



# List of Functions

Functions of image data processing component:

No.	Category	Name of function	Description
1	Image data stream	Obstacle detection and notification	<ul style="list-style-type: none"> <li>(1) Processes video data into chunks of image data</li> <li>(2) Retrieves CAN data linked to image data from the CAN datastore</li> <li>(3) Sends image data to obstacle inference containers and identifies objects that appear repeatedly</li> <li>(4) Stores obstacle information in a queue for an obstacle and forwards obstacle information to the notification process</li> </ul>
2		Obstacle inference	Detects obstacles appearing in the image data
3		Call for an object position estimation and notification of obstacle details	<ul style="list-style-type: none"> <li>(1) Retrieves CAN data linked to image data from the CAN datastore</li> <li>(2) Sends image and CAN data to an object-position-estimation container and receives the estimated object position results</li> <li>(3) Forwards obstacle information to the notification process</li> </ul>
4		Estimated object position	Performs an object position estimation and outputs the object label and position coordinates (lat/long)
5		Detection of line of vehicles in adjacent lanes and congestion cause analysis	<ul style="list-style-type: none"> <li>(1) Checks the image data showing the picture of the vehicle to analyze the object detection method and circumstances in regard to the footage taken and detects the occurrence of congestion in the adjacent lanes</li> <li>(2) Infers the cause of congestion based on information of features around the place where the occurrence of congestion was detected</li> </ul>
6	Dynamic deployment of applications	Aggregation of monitoring metrics, aggregation, and threshold assessment	<ul style="list-style-type: none"> <li>(1) Aggregates the monitoring metrics information of the GPU node deployed to each site</li> <li>(2) Calculates the average GPU usage rate per unit of time</li> <li>(3) Assesses whether the average calculated for the aggregation of monitoring metrics exceeds the threshold</li> </ul>
7		Deployment planning and site selection	<ul style="list-style-type: none"> <li>(1) Preplans a deployment site when a bottleneck is detected within a site</li> <li>(2) When a bottleneck is detected within a site, decides on the final site for deployment based on the situation of the planned deployment site and then proceeds with deployment</li> </ul>
8	Batch processing for image data	Recognition of traffic light/road sign and estimated object position	<ul style="list-style-type: none"> <li>(1) Detects traffic lights and road signs appearing in image data</li> <li>(2) Estimates the object position and output object label and position coordinates (lat/long)</li> </ul>
9		Generation of a road network map including lanes	Synchronizes video and CAN data to DC in other sites and generates a road network map including lanes
10		Static map generation	Combines a road network including lanes and the estimated object position results and records them on a static map

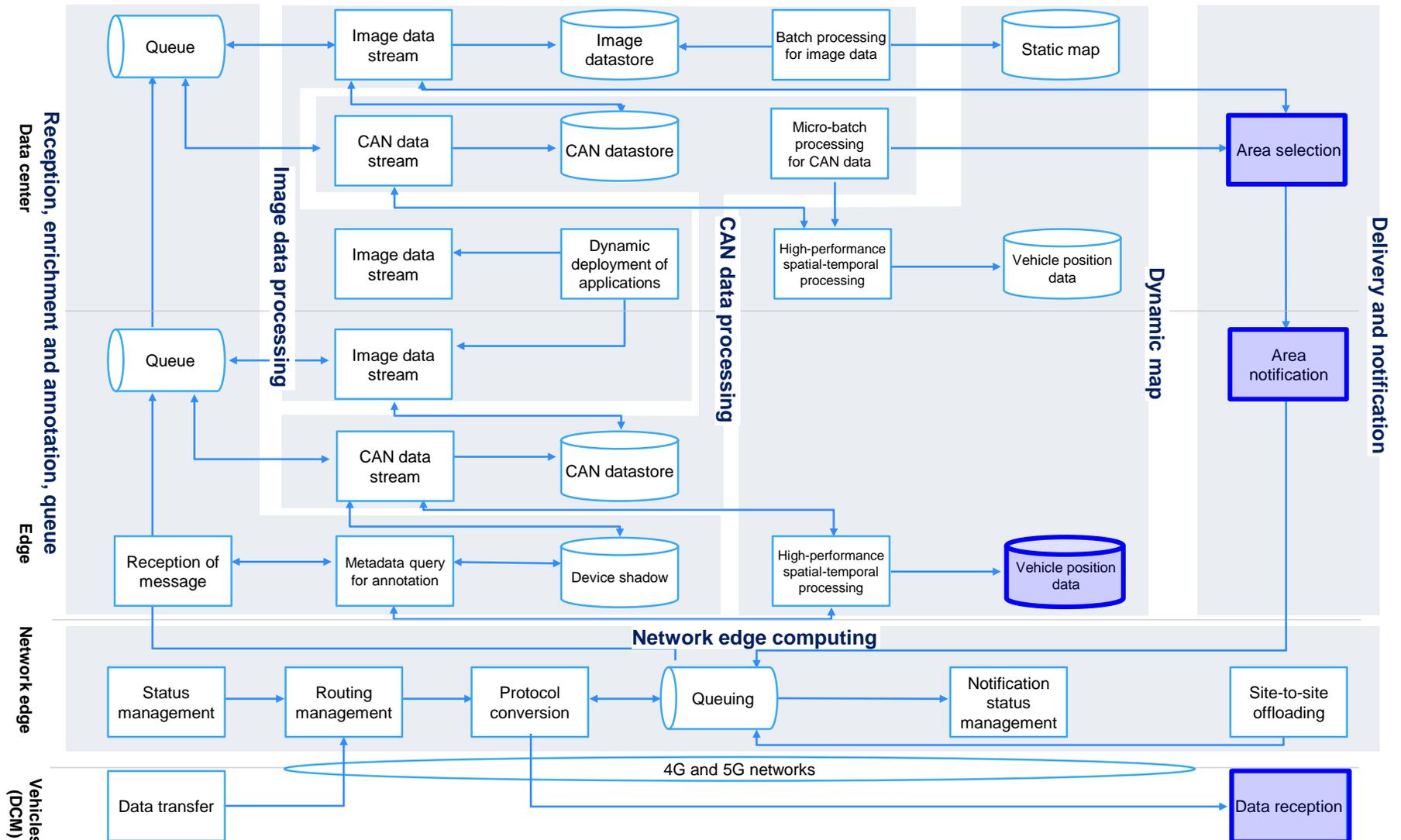
# List of Hardware

List of hardware used in the image data processing component.

No.	Name of server	Location	Name of function	Number of servers	Server specs
1	Queue server	Edge node	Queue for video	1	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 288 GB Disk (SSD): 1,600 GB
2	Image data stream server (with GPU)	Edge node	Obstacle detection and notification, obstacle inference	2	GPU: Nvidia V100-PCIe CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 288 GB Disk (SSD): 1,600 GB
3		Data center	Detection of line of vehicles in adjacent lanes and congestion cause analysis Recognition of a traffic light/road sign and estimated object position, generation of a road network including lanes, and generation of a static map	2 4	
4	Image data stream server (without GPU)	Data center	Call for an estimated object position and notification of obstacle details, and an estimated object position	9	CPU (Xeon Silver 4110): 2P, 16C, 32T Memory: 192 GB Disk (HDD): 1,200 GB
5	Image data stream LB server	Data center	Call for an estimated object position	1	CPU (Xeon E5-2630v4): 2P, 20C, 40T Memory: 128 GB Disk (HDD): 300 GB
6	Container management server	Data center	Aggregation of monitoring metrics and a threshold assessment, and deployment planning and site selection	1	CPU (Xeon E5-2630v4): 2P, 20C, 40T Memory: 128 GB Disk (HDD): 300 GB
7	CAN datastore	Edge node and data center	—	1 each	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 192 GB Disk (HDD): 1,200 GB
8	Image datastore	Data center	Mesh aggregation, lane aggregation	4	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 288 GB Disk (SSD): 1,600 GB

# Architecture

Components used in the notification and delivery platform verification are highlighted in blue.



# List of Functions

Functions of notification and delivery process component:

No.	Category	Name of function	Description
1	Target area selection	Query for target edge server	Identifies the edge node for transferring the notification request.
2	Notification on target area	Query for target vehicle	Narrows down target vehicles to transmit image data send request. The following two patterns of logic are assumed to be used for narrowing down. (1) Query for target vehicle using only vehicle position data (2) More precise query taking into account the specs and angle of the vehicle-mounted camera
3		Request for notification	Sends a list of vehicles in mesh identified through a query for a target vehicle to notification process

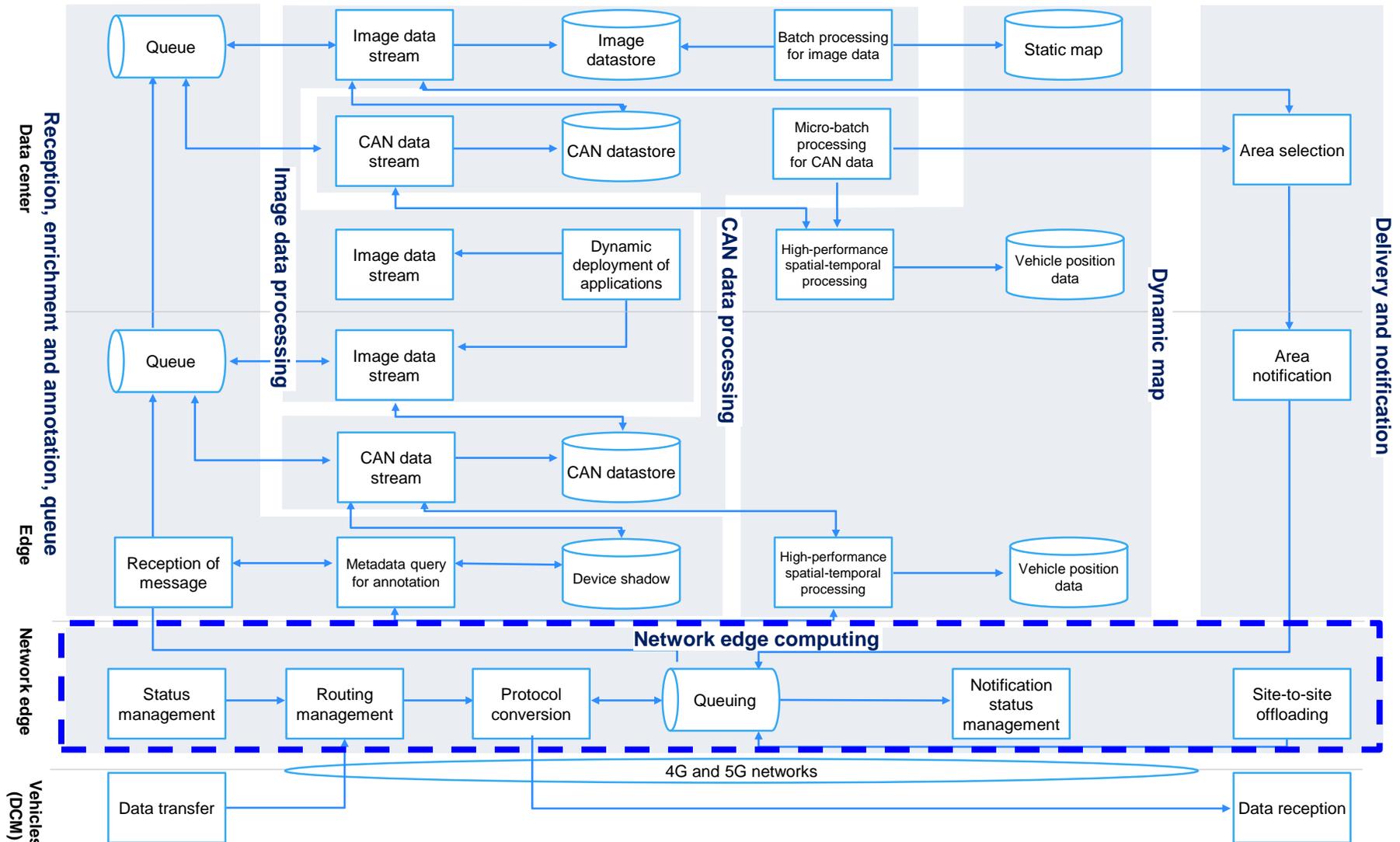
# List of Hardware

List of hardware used in the notification and delivery processing component.

No.	Name of server	Name of function	Number of servers	Server specs
1	Target area selection server	Query for the target edge server	1	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 192 GB Disk (HDD): 1,200 GB
2	Notification on the target area server	Query for the target vehicle, request for notification	1	CPU (Xeon E5-2630 v4): 2P, 40C, 80T Memory: 128 GB Disk (HDD): 1 TB
3	Vehicle position data	—	3	CPU (Xeon Gold 5118): 2P, 24C, 48T Memory: 192 GB Disk (HDD): 1,200 GB

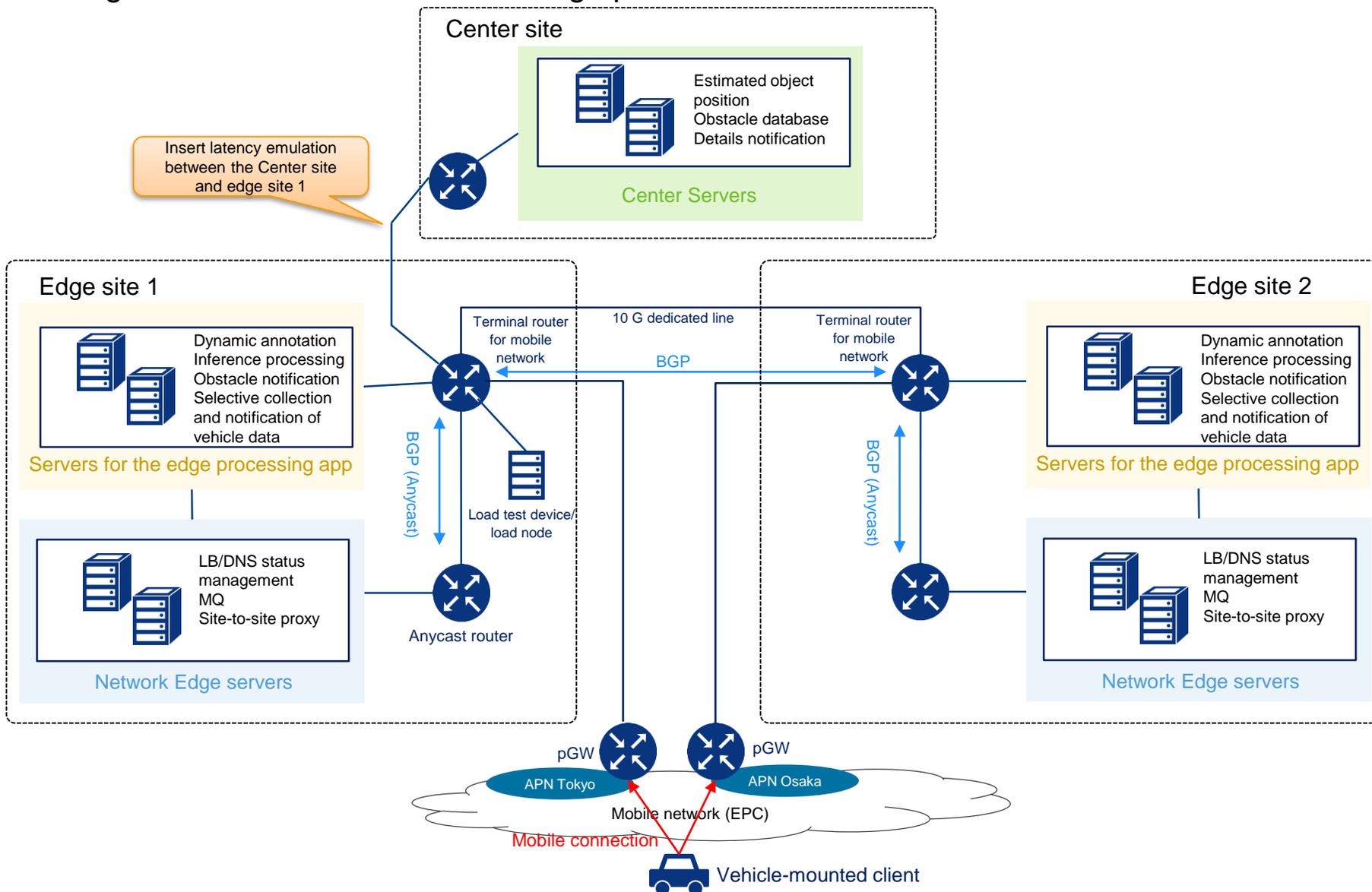
# Architecture

Sections shown inside the broken blue line have been verified in the network edge platform verification.



# System Configuration

System configuration used for the network edge platform verification.



# List of Functions

The below table shows the functions of the Network Edge platform. In the verification, these functions were equipped mainly to an IA server, and scaling out was enabled by increasing the number of servers.

No.	Name of function		Description of function
1	Routing management	BGP Anycast	Guides data from vehicles to the nearest Network Edge by propagating the IP address where the message should be sent through BGP Anycast
		Load balancing	Sorts messages and distributes load based on the established policy
		TLS termination	Authentication and decrypts TLS traffic
2	Protocol conversion	—	Converts the communication protocol (HTTP, MQTT, WebSocket, etc.)
3	Queuing	—	Holds the received message over a certain period
4	Site-to-site offloading	Network Edge status management and sharing	Manages the status of other Network Edges and selects a site for sharing messages
		Site-to-site proxy	Sends/receives a message between edge sites
5	Notification status management	Session management	Manages the connection status between the vehicle and edge site
		Notification status management	Manages the notification status of the notification messages to vehicles
6	Status management	Monitoring metrics	Monitors metrics information of the Network Edge
		Visualize	Visualizes the aggregated monitoring metrics information

# Hardware and Software Configurations **TOYOTA**

Hardware and software configurations of the Network Edge server for verifications 1 through 4

Verification 1 DNS, LB, annotation server	
Hypervisor specs	
CPU	Intel Xeon E5-2650 2.00 GHz 8 C/16 T
RAM	64 GB
SAS	1.67 TB (300 G × 8 RAID50)
Hypervisor	ESXi-6.5.20180502001-standard
Virtual machine specs	
CPU	8vCPUs
RAM	32 GB
HDD	64 GB
OS	Ubuntu 18.04.1.0 LTS amd64
Kernel	Linux Kernel 4.15.0-45-generic
Nginx (LB)	1.15.6
dnsdist (DNS)	1.3.3 (Lua 5.1.4)
Knot DNS (DNS)	2.6.9

Verifications 2 and 3 of the Network Edge, annotation server	
Hardware specs	
CPU	Intel Xeon Gold 6140 2.3 GHz 18C/36T ×2
RAM	384 GB
SSD	480 GB (480 G ×2 RAID1)
Software specs	
OS	Ubuntu 18.04.3.0 LTS
Kernel	Linux Kernel 4.15.0-135-generic
Nginx(LB)	1.14.0

Verification 4 of the Network Edge, annotation server	
Hardware specs	
CPU	Intel Xeon Gold 6140 2.3 GHz 18 C/36 T x2
RAM	384 GB
SSD	480 GB (480 G ×2 RAID1)
Software specs	
OS	Ubuntu 20.04.1 LTS
Kernel	Linux Kernel 5.4.0-65-generic
Nginx(LB)	1.19.3

# Hardware and Software Configurations **TOYOTA**

Hardware and software configurations of the Network Edge server for verification 5.

Verification 5 of all Network Edges	
Hardware specs	
CPU	Intel Xeon Gold 6140 (2.3 GHz, 18 C) ×2
RAM	384 GB DDR4
SSD	480 GB (RAID1)
Software specs	
OS	Ubuntu 20.04.1 LTS
nginx (LB)	1.18.0
TLS (Cipher)	v1.2 (ECDHE-RSA-AES128-SHA256)

Verification 5 of Network Edge 2 (TLS Accelerator, QuickAssist Adapter)	
Hardware specs	
CPU w/ HT	Yes
NIC	Intel X710
QAT board	QuickAssist Adapter 8950 ×3
Software specs	
Kernel	Linux Kernel 5.4.0-65-generic
OpenSSL	1.1.1f

Verification 5 of Network Edge 1 (CPU)	
Hardware specs	
CPU w/ HT	Yes
NIC	Intel X710 ×2
Software specs	
Kernel	Linux Kernel 5.4.0-42-generic
OpenSSL	1.1.1f

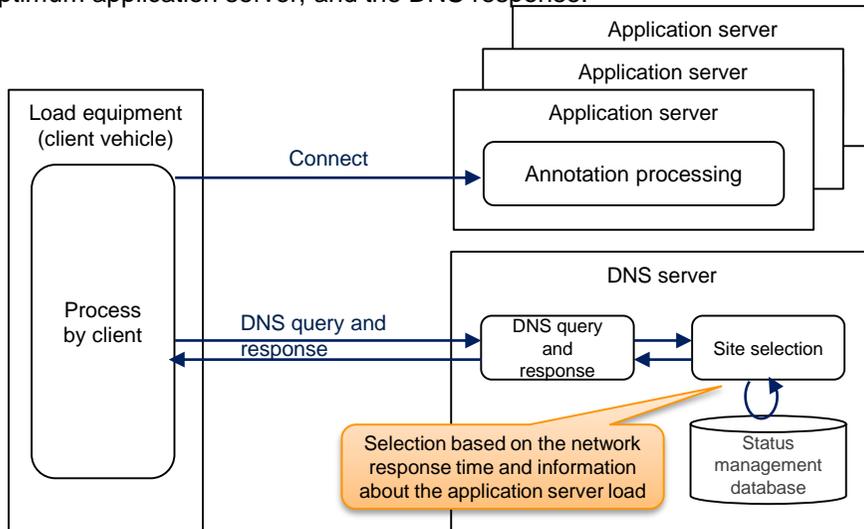
Verification 5 of Network Edge 3 (ASIC based SmartNIC)	
Hardware specs	
CPU w/ HT	No
NIC	Intel X710
SmartNIC	Mellanox ConnectX-6 Dx
Software specs	
Kernel	Linux Kernel 5.4.0-65-generic (ktls enabled)
OpenSSL	3.0.0-alpha13

# Evaluation 1: Evaluation Architecture

To realize for connecting the closest NW edge to the vehicle and the wide-area routing due to site failure or increased load, the routing management of the vehicle data traffic must be administered properly. This evaluation tested the basic functions and performance of the DNS method as well as the LB method, which we designed as a routing management method. Each architecture of the two methods is illustrated below.\*

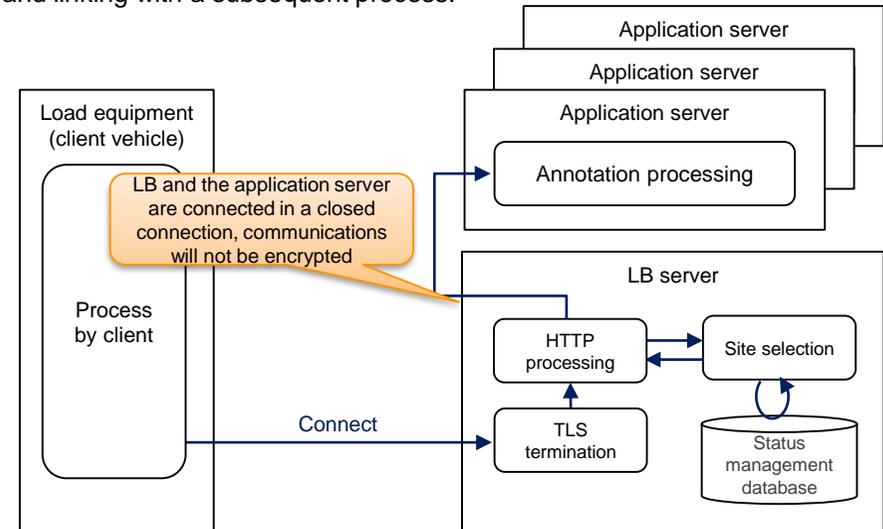
## DNS method

Load information of the application server, network, etc., is kept in a status management database. Routing management is done by referring to information in the database through the process at DNS, selecting the optimum application server, and the DNS response.



## LB method

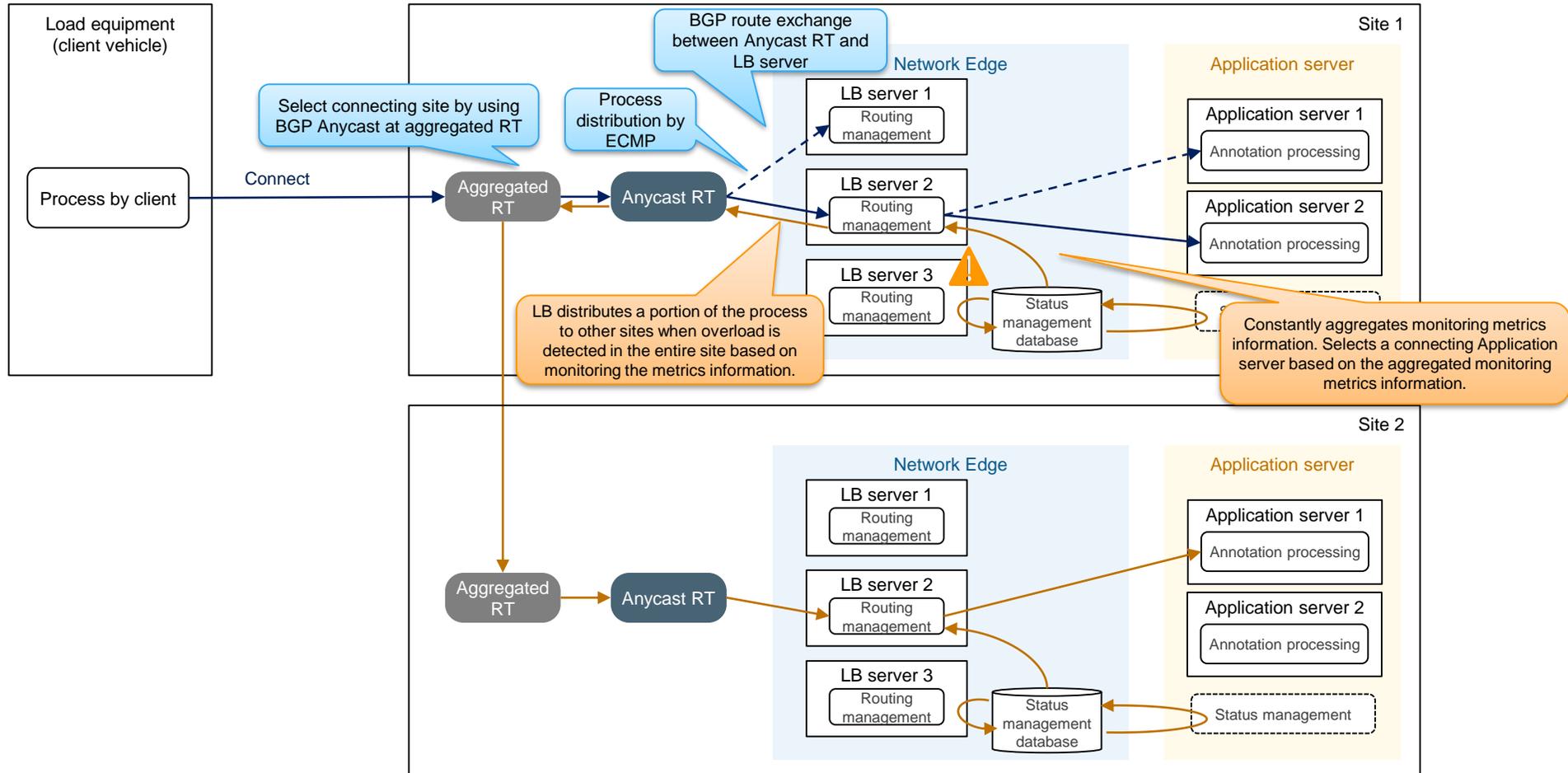
Load information of the application server, network, etc., is kept in a status management database. Routing management is done by referring to the database when processing at LB, selecting the optimum application server, and linking with a subsequent process.



\* Use of BGP Anycast is assumed for the selection of connecting edge site (DNS server, LB server). BGP Anycast selects a connecting site based on the advertised route information between gateways. It is introduced to realize reconnection when pGW changes within the connecting mobile network along with the vehicle's motion.

# Evaluations 2 and 3: Evaluation Architecture

Routing management based on monitoring metrics information is required to realize wide-area routing that is responsive to the failure or load status at each edge site. We test the routing management based on monitoring metrics information under LB method in Evaluations 2 and 3. A basic performance evaluation of an edge site was also carried out. The architecture is illustrated below.

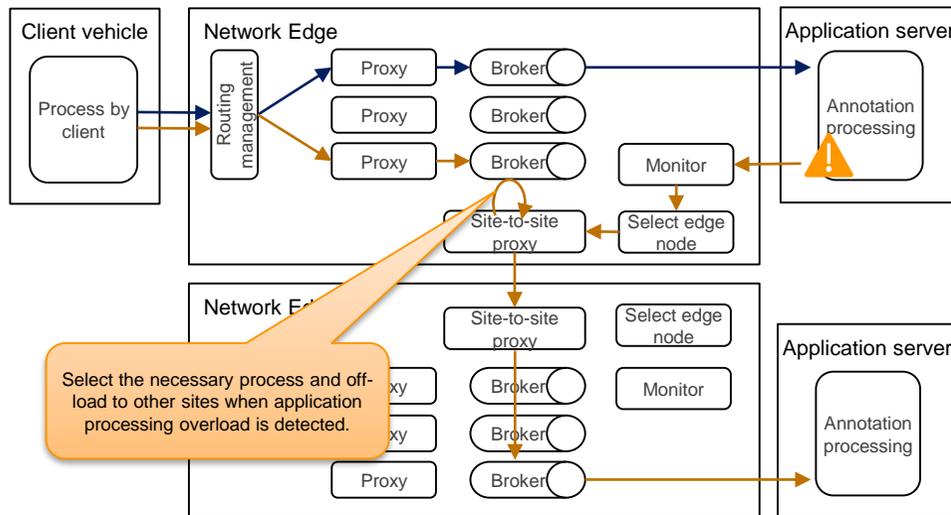


# Evaluation 4: Evaluation Architecture

In an environment where a Network Edge is dispersed to multiple sites, synchronizing data between sites becomes important because communication with vehicles must continue even when availability is lost due to failure at the Network Edge site. We will address this by introducing a Message Queue (MQ) to a Network Edge to improve data synchronization and availability across the entire Network Edge. This verification tested data synchronization between edge sites using wide-area-distributed MQ. The verification architecture is illustrated below.

## Data transmission from vehicles

Off-loading processes among edge sites is necessary as application processing may become overloaded when connections from vehicles are concentrated to a certain site. When overload is detected by load status monitoring of application processing, only the necessary data is off-loaded through site-to-site proxy, to be processed at other sites.



## Messages notification to vehicles

Connected edge node may shift from one to the next along with the notification target vehicle's motion. Therefore, notification status management becomes necessary by synchronizing a notification message between edge nodes. MQ logic that covers multiple sites is created to synchronize data between sites. Metadata of a notification message is shared at each site, and the message is retrieved from MQ logic and vehicles receive a notification upon triggering events such as a connection request from vehicles.

