BITS Pilani presentation

BITS Pilani
Pilani Campus

Paramananda Barik
CS&IS Department

innovate achieve lead

BITS Pilani

Pilani Campus

SEZG586/SSZG586,

MEC – Multi-access Edge Computing

Lecture No.14

MEC – Multi-access Edge Computing

innovate achieve lead

Mobile operators - advantage of low latency services that 5G promises

New range of services

faster gaming experience, Augmented/Virtual Reality, connected cars

Azure, AWS, Google

building their own MEC platforms

AWS Wavelength

MEC Architecture

MEC Architecture is based on ETSI model – the defacto body for MEC standards

By understanding the MEC architecture, you:

- As service provider, will know how to evolve to a standard based MEC platform/architecture.
- As vendor/developer, will enable you to tell your customer how your solution fits the ETSI MEC model

Definition of MEC

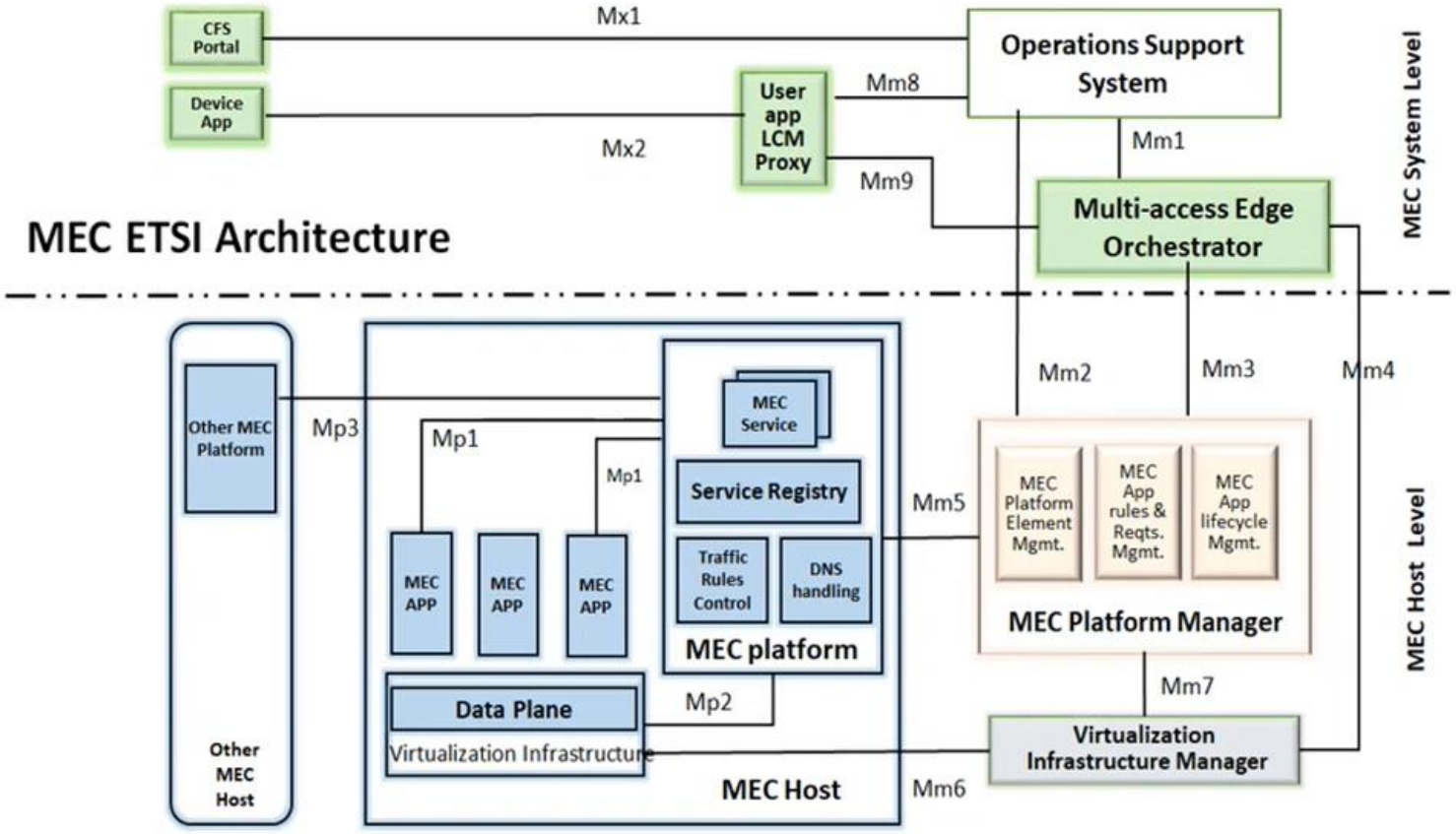
According to the ETSI, the MEC is defined as following

"Multi-access Edge Computing (MEC) offers application developers and content providers **cloud-computing capabilities** and an **IT service environment** at the **edge of the network**. This environment is characterized by **ultra-low latency** and **high bandwidth** as well as **real-time access to radio network information** that can be leveraged by applications."

ETSI's MEC standards are guided by the following principles:

- Edge technology should have a virtualization platform to be considered MEC.
- MEC can be deployed at radio nodes, aggregation points, and the edge of the core network.
- APIs in a MEC environment should be simple, controllable, and if possible, reusable for other tasks.
- Since the compute, storage, and network resources that a MEC application requires may not match what are available at a node, a MEC network needs a system-wide lifecycle management of applications to handle these variables correctly.
- MEC systems must be able to relocate a mobile edge application running in an external cloud to a MEC host and back while fulfilling all of the application's requirements

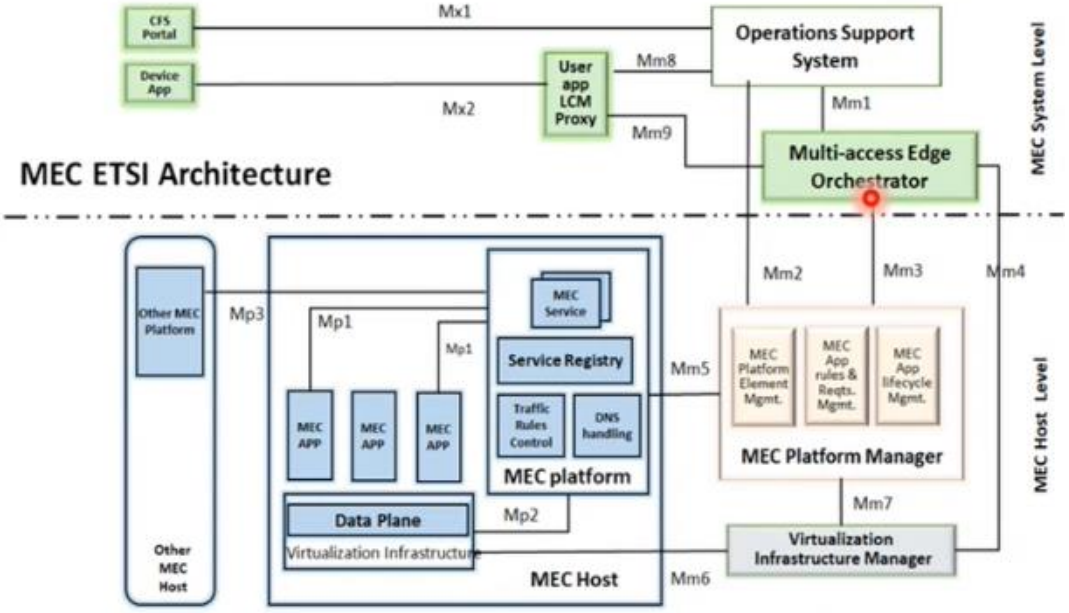
MEC ETSI architecture



MEC ETSI architecture

Three distinct blocks:

- MEC Host
- MEC Platform Manager
- MEC Orchestrator



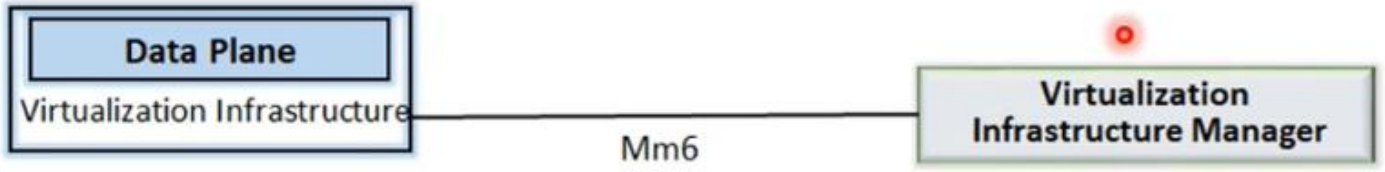
Virtualization Infrastructure Manager(VIM)

Manage the virtual machines (VMs) on top of physical infrastructure (compute, storage, and networking).

It is responsible for

- allocating
- maintaining
- releasing

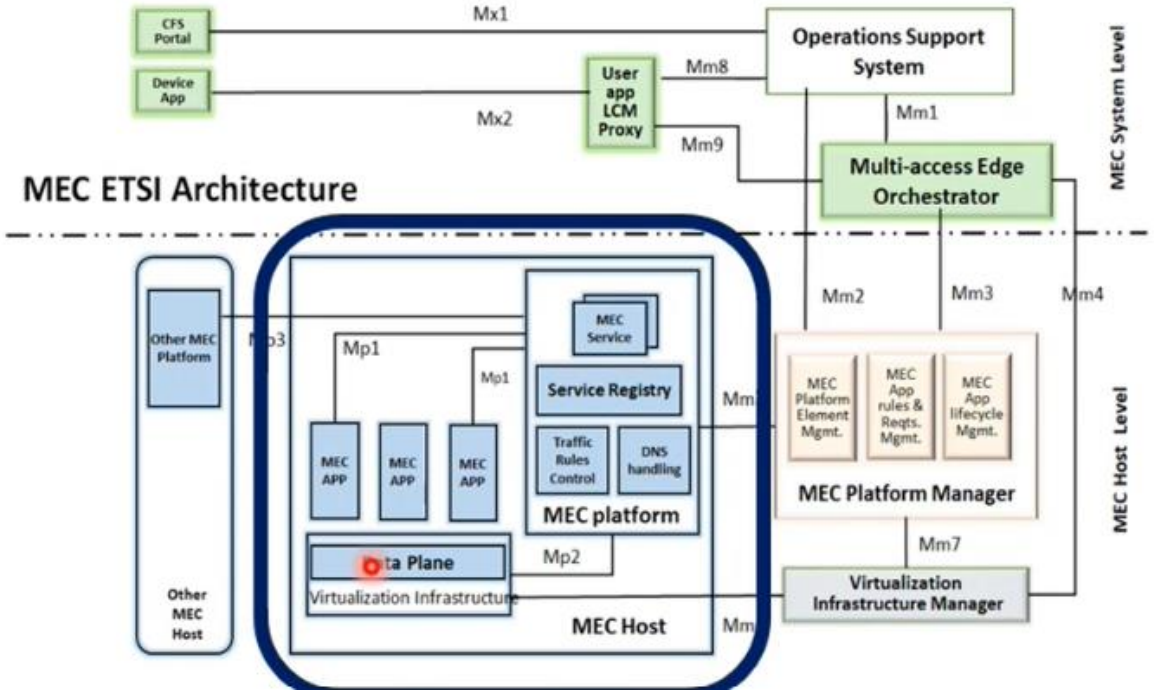
virtual resources of the “virtualization infrastructure”.



MEC Host

Virtualization Infrastructure

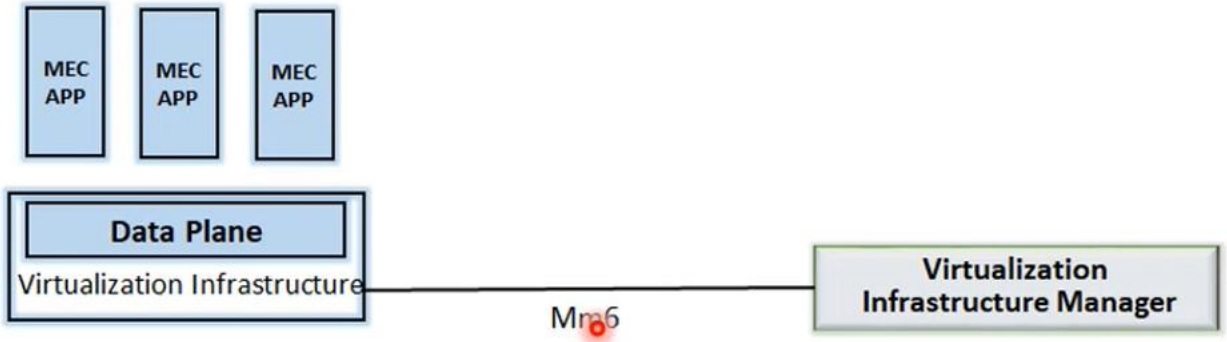
- MEC Apps
- MEC Platform



MEC Application

These are the actual applications that are run in MEC on top of VMs

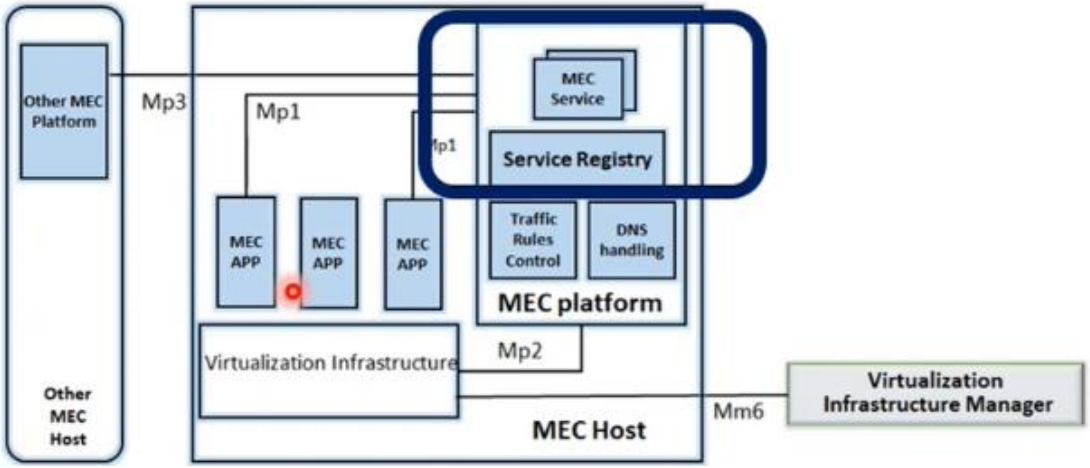
Actual apps in MEC like an application for gaming or Virtual Reality or Augmented Reality



MEC Service

"MEC Service" is an important block in MEC.

- The network-related APIs are exposed by MEC service to MEC Apps through reference point Mp1.
- Also, the MEC platform can consume these services.



MEC Service

MEC Apps are network-aware

MEC Service help by exposing the network information through APIs

At least three types of following services must be exposed by “MEC service”

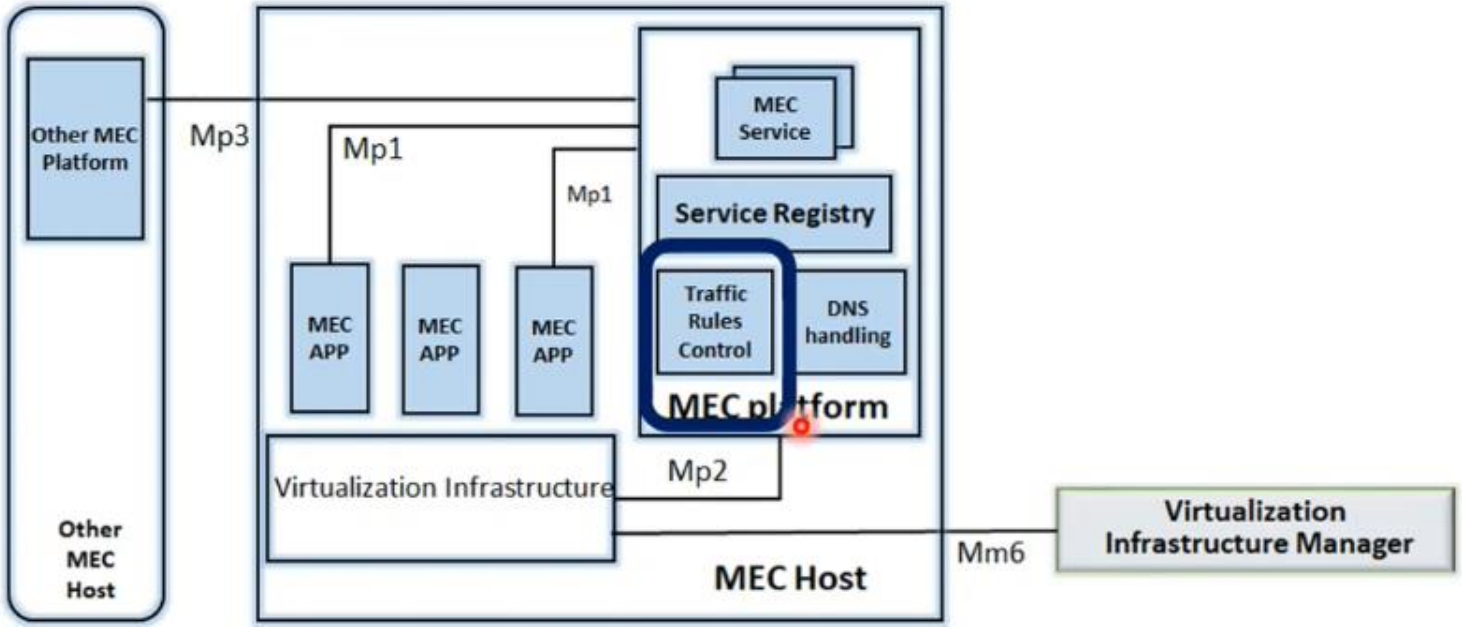
They are

- Radio Network conditions.
- Location information (for example, location of a User Equipment(UE))
- Bandwidth Manager- The Bandwidth Manager service, when available, allows allocation of bandwidth to certain traffic routed to and from MEC applications and its prioritization

They are part of service registry

Traffic Rules Control

As MEC Platform is serving multiple applications, simultaneously, It should be able to assign priorities through “Traffic rules control”

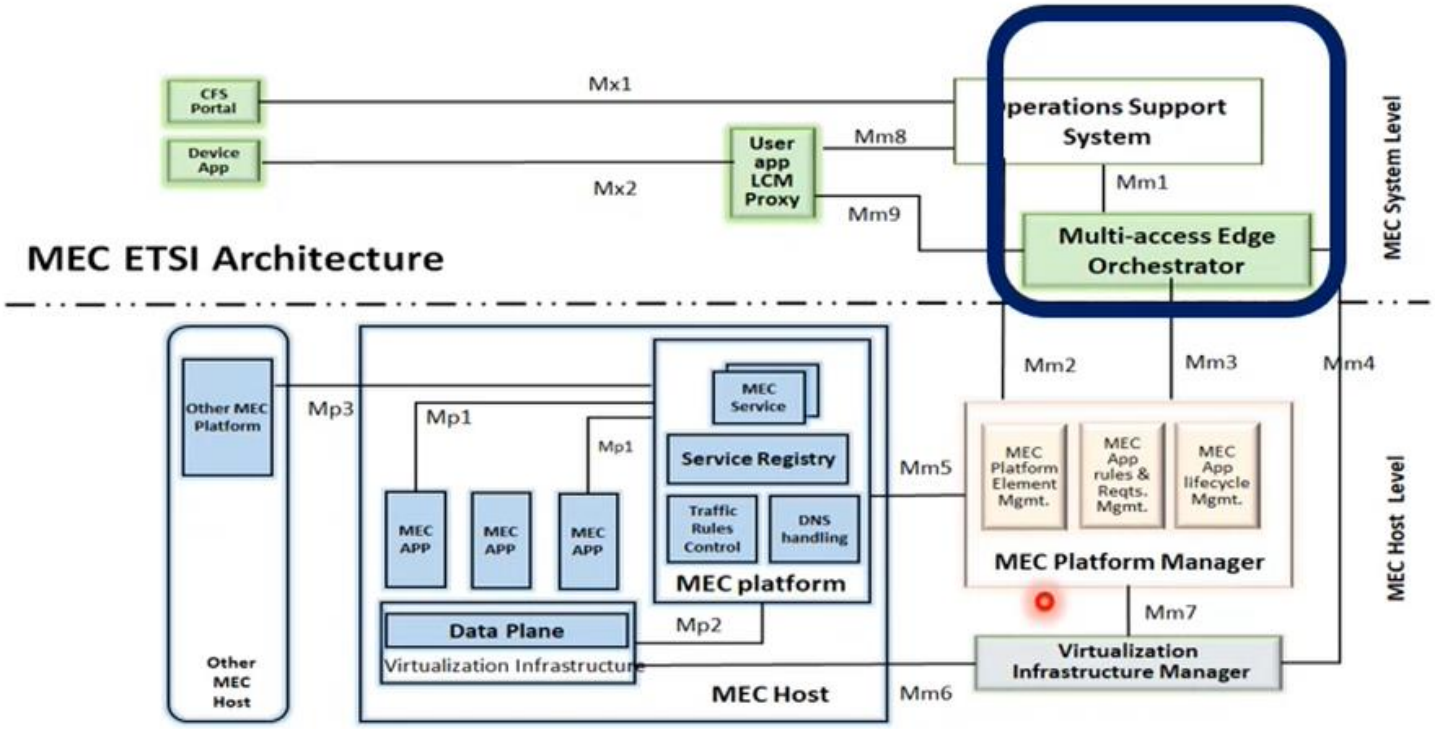


MEC Platform Manager

MEC Platform manager performs following functions:

- MEC Apps' Life-cycle management:
  - instantiating, maintaining and tearing down MEC Apps on VMs
- Element Management:
  - FCAPS (Fault, Configuration, Accounting, Performance, Security) management for the MEC platform
- Managing the application rules, traffic rules, DNS configuration

Multi-Access Edge Orchestration

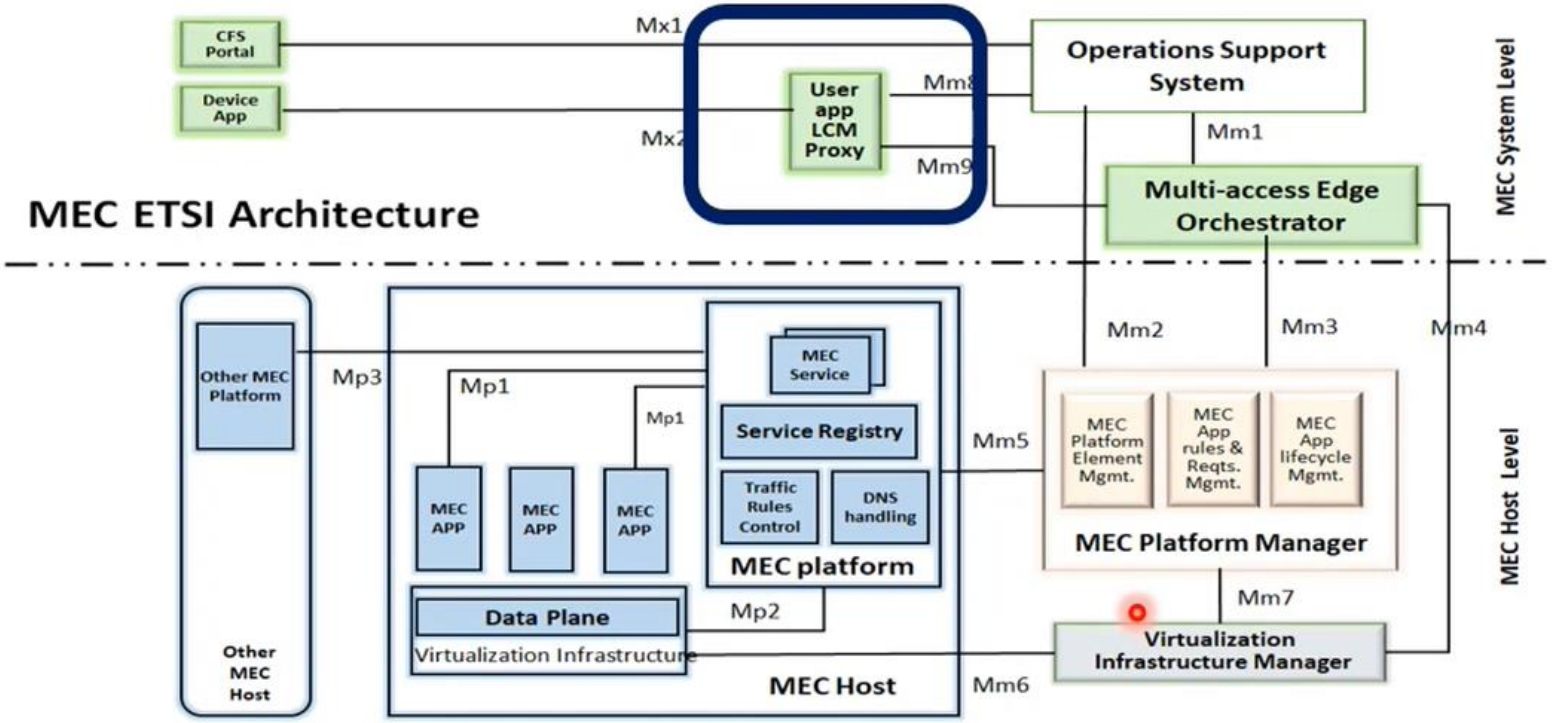


MEC Orchestration

It does the following functions

- Lifecycle management of MEC Apps (Compare this with MEC Platform manager, which can do a similar function). The Orchestrator achieves this function by talking to the application through the MEC platform manager.
- On-boarding of application packages, including checking the integrity and authenticity of the packages.
- Selecting appropriate MEC host(s) for application instantiation based on constraints, such as latency, available resources, and available services

User App LCM Proxy



Intelligent Video-Accelation

innovate achieve lead

Video Download Service

Media delivery transitioning to HTTP-based transport

uses TCP at L4

Performance issues of using TCP over wireless networks

Idea: To include a throughput guidance component at edge

Application can adapt its video codec and TCP parameters to the conditions of a wireless network

Improving the performance of video transmission from all perspectives

user experience

network load

Intelligent Video Accelation

Options:

An edge-based video codec

Or

Just a “throughput guidance” service

Reads the radio network information

Converts the information provided into a “guidance report”

Send it to a video coder/transcoder elsewhere in the cloud

Optionally - throughput guidance component may also configure traffic rules

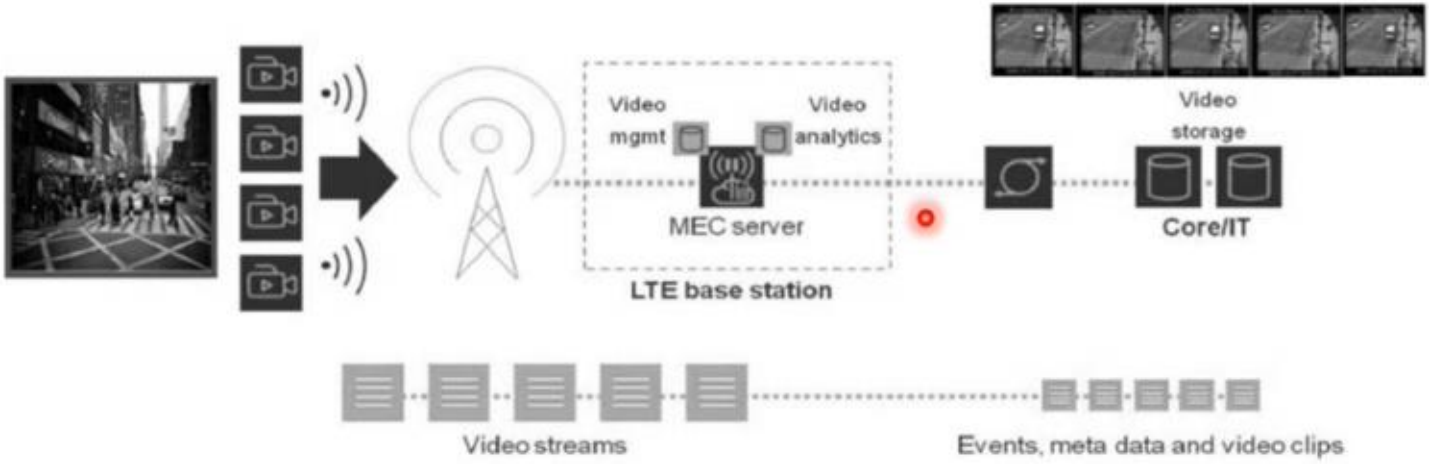
handle of the application’s traffic - it may decide between a mobile RAN- and a WLAN-based access

Video Stream Analysis

Video upload service

Surveillance system - consisting of a number of cameras with a cloud-based video-analytics system

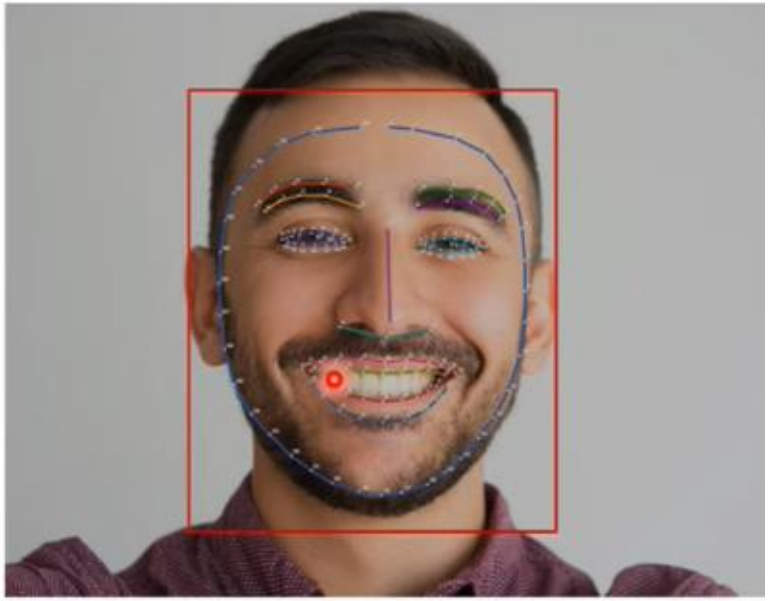
System - transmission of multiple video streams for feature extraction and analysis



Video Stream Analysis

Edge-based pre-processing

- At the edge, raw video images are processed extracting only the information containing relevant features.
- The extracted information is forwarded into the cloud, which performs further image processing for extractions and then performs analysis



Example: facial recognition system

Augmented Reality

A smart museum augmented reality application

A smart phone or a viewing device (smart glasses) is pointed at a museum object

Relevant information is augmented by local servers

Locally processing and delivery of such data makes sense

- end-user QoE perspective
- network performance perspective
- system design perspective

Connected Vehicles

MEC is vehicular automation

- strict latency requirement
- perform computation on an aggregate of multi-vehicle data

IoT Gateways

IoT Gateway is a key function – usually realized in a stand-alone device

- communication aggregation
- computational offload
- identity proxy

Public Edge Clouds – AWS Greengrass

Data Analytics in the Cloud and Edge

Analytics for the IoT segment deals with:

- **Structured data** (for example, SQL storage): A predictable format of data
- **Unstructured data** (for example, raw video data or signals): A high degree of randomness and variance
- **Semi-structured** (for example, Twitter feeds): Some degree of variance and randomness in form

Data Analytics

Data need to be interpreted and analyzed

- In real time as a streaming dataflow
- It may be archived and retrieved for deep analytics in the cloud.
- It is simply logged and dumped to a data lake like a Hadoop database.
- This is the **data ingest** phase.

Data Analytics

- Staging Phase
  - A messaging system like Kafka will route data to a
    - Stream processor
    - Batch processor, or
    - Both

Data Analytics – Staging Phase- Stream Processing

- Stream processing accepts a continuous stream of data.
- Processing is typically **constrained** and **very fast**, as the data is **processed in memory**.

  Therefore, processing must be as fast, or faster, than the rate of data entering the system.
- While stream processing provides near-real-time processing in the cloud, when we consider industrial machinery and self-driving cars, stream processing does not provide hard real-time operating characteristics.

Data Analytics

Data analytics intends to find events, usually in a streaming series of data

Analytic functions:

- Preprocessing
- Alerting
- Windowing
- Joins
- Errors
- Databases
- Temporal events and patterns
- Tracking
- Trends
- Batch queries
- Deep analytics pathway
- Models and training
- Signaling
- Control

Analytic function: Processing

Preprocessing includes

- Filtering out events of little interest
- Denaturing
- Feature extraction
- Segmentation
- Transforming data to a more suitable form (although data lakes prefer no immediate transformation), and
- Adding attributes to data such as a tag

Analytic function: Alerting

Inspect data, and if it exceeds some boundary condition, then raise an alert.

The simplest example is when the temperature rises above a set limit on a sensor.

Analytic function: Windowing

A sliding window of events is created that only draws rules upon that window.

Windows can be based on time (for example, one hour), or length (2000 sensor samples).

Sliding windows

Ex: inspect only the 10 latest sensor events and produce a result whenever a new event arises

Batch windows

Ex: produce an event only at the end of the window

Windowing is good for rules and for counting events.

For instance, you could look for the number of temperature spikes in the last hour and resolve that a defect will occur on some machine.

Analytic function: Joins

Joins combine multiple data streams into a new single stream.

A scenario where this applies is a logistics example, say a shipping company tracks their shipments with asset-tracking beacons and their fleet of trucks, planes, and facilities have geolocation information streaming, as well.

There are initially two streams of data: one for the package and one for a given truck.

The two streams **join** when a truck picks up a package

Analytic function: Errors

Millions of sensors will generate

- Missing data
- Garbled data
- Data that is out of sequence.

This is important in IoT cases with multiple streams of asynchronous and independent data.

For example, data may be lost in a cellular WAN if a vehicle enters an underground parking garage.

Analytic function: Database

The analytics package will need to interact with some data warehouse.

For example, if data is streaming in from a number of sensor tags

An example: Bluetooth asset tags tracking whether an item is stolen or lost.

A database of missing tag IDs would be referenced from all the gateways streaming in Bluetooth tag IDs to the system.

Analytic function: Temporal events and patterns

Most often used with the window pattern

Here, a series or sequence of events constitutes a pattern of interest – a state machine.

Ex: monitoring the health of a machine based on temperature, vibrations, and noise.

A temporal event sequence could be as follows:

1. Detect whether the temperature exceeds 100° C.
2. Then detect whether vibrations exceed 1 m/s.
3. Next, detect whether the machine is emitting noise at 110 dB.
4. If those events take place in that sequence, only then raise an alert.

Analytic function: Tracking

Tracking involves

when or where something exists or an event has occurred, or

when something doesn't exist where it should.

A very basic example is geolocation of service trucks

This has application in agriculture, human movement, tracking patients, tracking high-value assets, luggage systems, smart city garbage, snow removal, etc.

Analytic function: Trends

This pattern is particularly useful for predictive maintenance.

Here, a rule is designed to detect an event based on time-correlated series data.

This is similar to temporal events but differs in the sense that temporal events have no notion of time, only sequence order. This model uses time as a dimension in the process.

Ex: A running history of time-correlated data could be used to find patterns like a livestock sensor does in farming.

Analytic function: Batch Queries

Batch processing is more comprehensive and deeper than real-time stream processing.

A well-designed streaming platform can fork analysis and call into a batch processing system.

Analytic function: Deep analytics pathway

In real-time processing,

- Decisions are made on the fly that some event has occurred.
- Whether that event really should signal an alarm may require further processing that will not operate in real time.

An example is a video surveillance system in smart city.

- A smart city issues an amber alert for a lost child.
- The smart city can issue a simple feature extraction and classification model for the real-time streaming engines.
- The model would detect license plates for a vehicle the child may be in, or potentially a logo on the child's shirt.
- The first step would be to image-capture vehicles' license plate numbers or logos on pedestrians and send them to the cloud. The analysis package may identify a plate of interest or a logo out of millions of image samples as a first-level pass.

Analytic function: Models and training

The first-level model described previously may, in fact, be an inference engine for a machine learning system.

These machine learning tools are built on trained models that can be used for in-flight, real-time analysis.

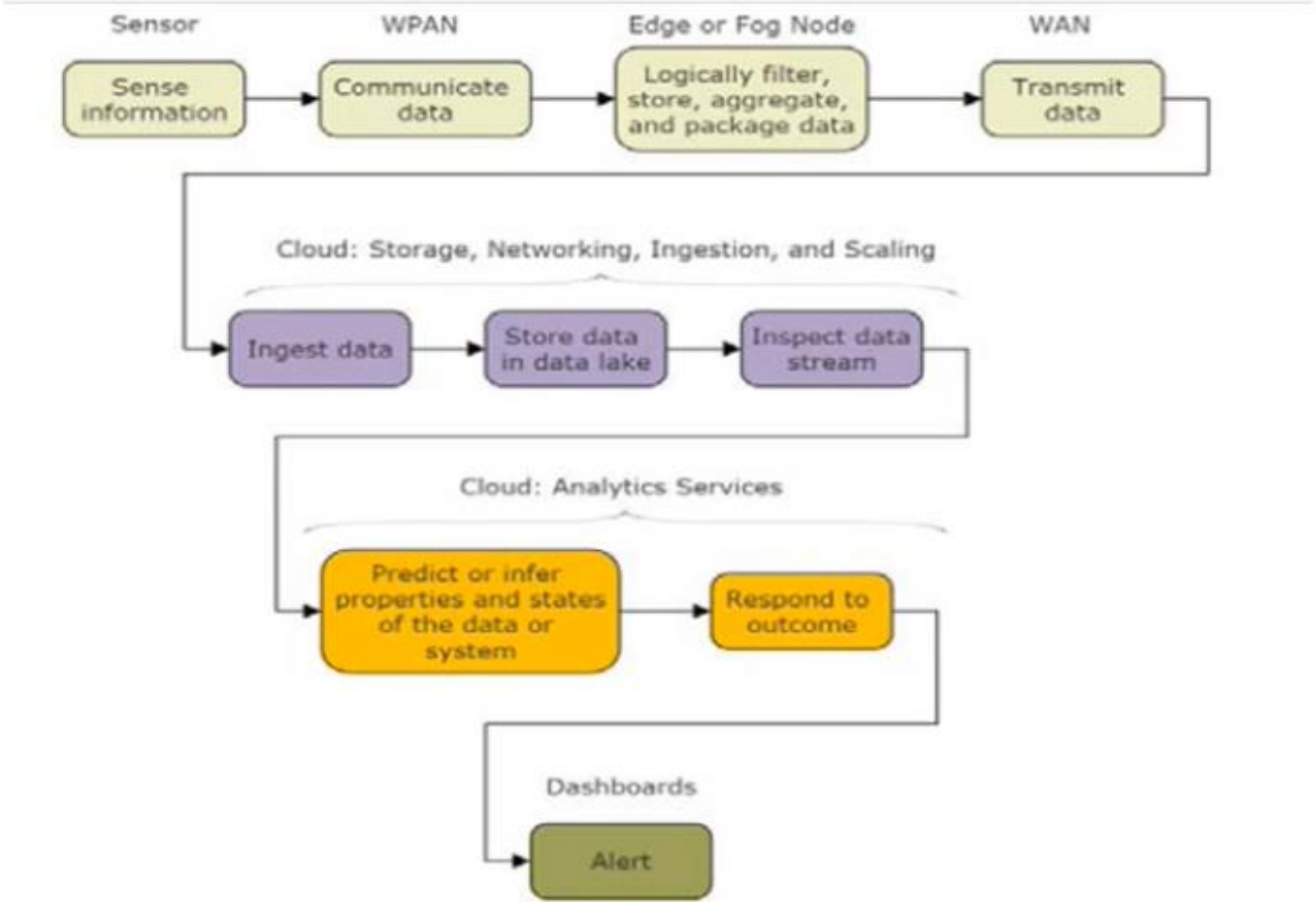
Analytic function: Control

Facilities need to be in place to manage this system

A way is needed to control these analysis tools:

- Starting
- Stopping
- Reporting
- Logging
- Debugging

Top-level cloud pipeline



Data Analytics

The analytics (predict-respond) portion of the cloud can take on several forms:

- **Rules engines**: These simply define an action and produce an outcome.
- **Stream processing**: These are where events like sensor readings are injected into the stream processor.

  -The processing path is a graph where nodes in the graph represent operators and send events to other operators.

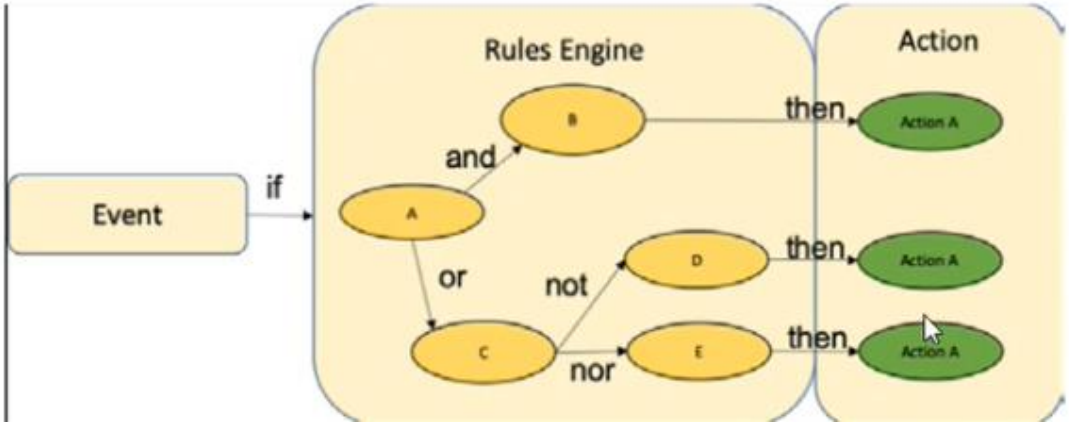
  -The nodes contain the code for that portion of the processing and a path to connect to the next node in the graph.

  -This graph can be replicated and executed in parallel on a cluster so it is amenable to scaling up to hundreds of machines.

Rule engines

A rules engine is simply a software construct that executes actions on events. For example, if the humidity in a room exceeds 50%, send an SMS message to the owner. These are also called business rule management systems (BRMSs).

Rules engines may or may not have state and be called stateful.



Rule engines

Drools can support two forms of chaining: forward and backward.

Pseudocode: Smoke Sensor = Smoke Detected Heat Sensor = Heat Detected

Forward

```
if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor == Heat_Detected)
then Fire
if (Smoke_Sensor == !Smoke_Detected) && (Heat_Sensor == Heat_Detected)
then Furnace_On
if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor == !Heat_Detected)
then Smoking
if (Fire) then Alarm
if (Furnace_On) then Log_Temperature
if (Smoking) then SMS_No_Smoking_Allowed
```

Let us assume:

```
Smoke_Sensor: Off
Heat_Sensor: On
```

Rule engines

innovate achieve lead

Backward:

1. Can we prove the temperatures are being logged? Take a look at this code:

   ```
   if (Furnace_On) then Log_Temperature
   ```

2. Since the temperatures are being logged, the antecedent (`Furnace_On`) becomes the new goal:

   ```
   if (Smoke_Sensor == !Smoke_Detected) && (Heat_Sensor == Heat_
   Detected) then Furnace_On
   ```

3. Since the furnace is proven to be on, the new antecedent comes in two parts: `Smoke_Sensor` and `Heat_Sensor`. The rules engine now breaks it up into two goals:

   ```
   Smoke_Sensor off
   Heat_Sensor on
   ```

4. The rules engine now attempts to satisfy both the subgoals. Upon doing so, the inference is complete.

Stream Processing

Ingestion – streaming, processing, and data lakes

-An IoT device is usually associated with some sensor or a device whose purpose is to measure or monitor the physical world.

-It does so asynchronously with respect to the rest of the IoT technology stack.

-That is, a sensor is always attempting to broadcast data, whether or not a cloud or fog node is listening.

Stream Processing

The IoT stream from a sensor to a cloud is assumed to be:

- Constant and never-ending
- Asynchronous
- Unstructured, or structured
- As close to real time as possible

Stream Processing

Streaming system must:

- Scale with event growth and spikes
- Provide a publish/subscribe API to interface
- Approach near-real-time latency
- Provide scaling of processing of rules
- Support data lakes and data warehousing

Ex: Apache Spark is a stream processing framework that processes data in small batches.

Stream Processing - Spark

Apache Spark is a stream processing framework that processes data in small batches.

It is particularly useful when memory size is constrained on a cluster in the cloud (for example, 1TB).

Spark is built on in-memory processing, which has the advantages of reducing filesystem dependency and latency

Use Cases

Industry	Use cases	Cloud services	Typical bandwidth	Real time	Analytics
Manufacturing	Operational technology Brownfield Asset tracking Factory automation	Dashboards Bulk storage Data lakes SDN Low latency	500 GB/day/factory part produced 2 TB/minute mining operations	Less than 1s	RNN Bayesian networks
Logistics and transport	Geolocation tracking Asset tracking Equipment sensing	Dashboards Logging Storage	Vehicles: 4 TB/day/vehicle (50 sensors) Aircraft: 2.5 to 10 TB/day (6000 sensors) Assets tracking: 1 MB/day/beacon	Less than 1s (real time) Daily (batch)	Rules engines
Healthcare	Asset tracking Patient tracking Home health monitoring Wireless health equipment	Reliability and HIPPA Private cloud option Storage and archival Load balancing	1 MB/day/sensor	Less than 1s: life critical Non-life critical: on each change	RNN Decision trees Rules engines

Use Cases

Running locally, deep learning models categorize packages,

-check if the postage matches a parcel's size, weight, and destination, and decipher barcodes, even the damaged ones.

-Edge intelligence also helps locate missing parcels: With AI, this takes a couple of people and just a few hours — instead of the previous several days and 8 to 10 people.

Kepler Night Nurse Edge Box increases patient safety

Kepler Vision, a Dutch medtech company, designed its Night Nurse Edge Box to keep elderly patients safe at night. The device runs Kepler software for detecting falls and physical distress and alerting staff when their help is required.

Instead of sending visual data to the cloud, Edge Boxes use computer vision locally and decide if nurses should intervene. This makes the system unaffected by the Internet connection breakdowns. Besides, it was estimated that replacing old sensors with Edge Boxes eliminates **99 percent** of false alarms.

Spanish connected smart tunnel offers assistance to drivers

The Cereixal tunnel in Spain's Galicia region leverages 5G and edge computing to capture and analyze data from tunnel sensors, cameras, and connected vehicles. Managers can remotely monitor what is happening inside the infrastructure via a dashboard.

Drivers who are moving through the tunnel receive notifications and alerts on the presence of pedestrians and emergency vehicles, possible delays because of traffic jams or accidents, weather conditions at the exit, and more. The project is supported by leading telecommunication companies Telefonica and Nokia.