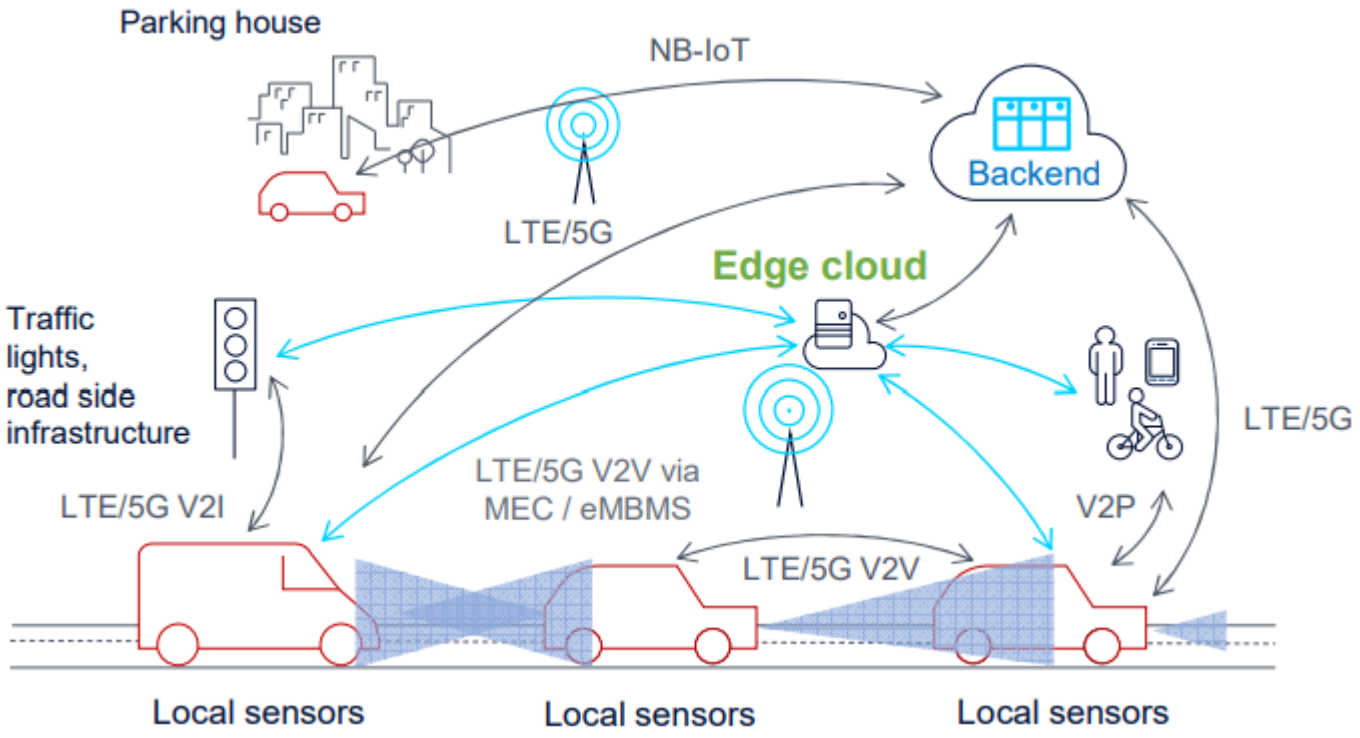
Self-organized systems: Data, learning, decisions, edge computing

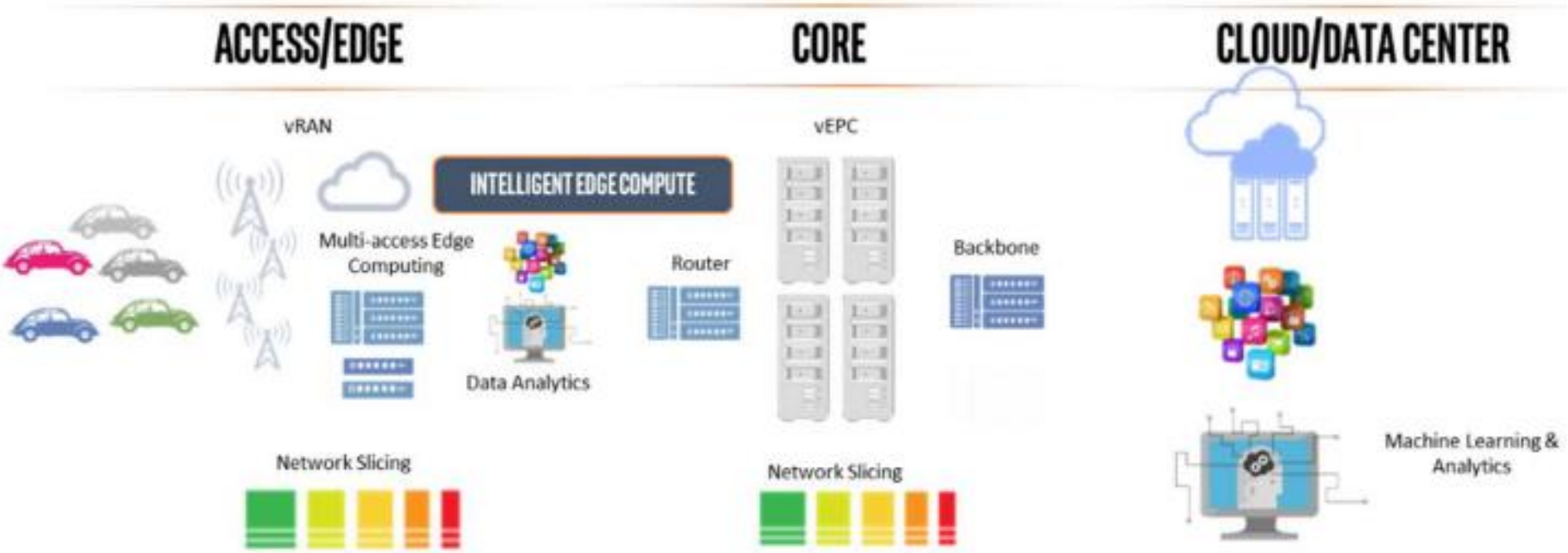
Mestrado em Engenharia de Computadores e Telemática

2021/2022

Use cases and data



Where to process the data?

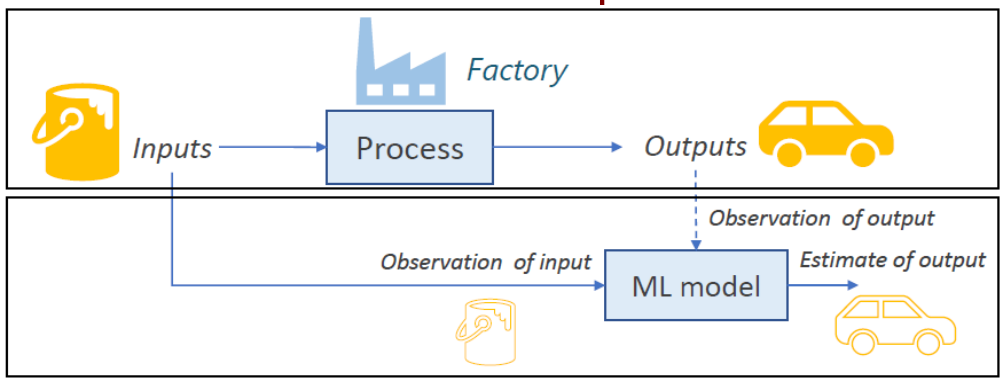


Data Analytics

- Processing of the data
- Get some decision with the data
- Network decisions with network and services data
  - Give more bandwidth?
  - Assign some special queue to reduce the delay?
- User decisions or element decisions
  - Obstacle in place?
  - Robot kicks the ball to the right?
- Network decisions with users' data
  - Predict handovers with to location and velocity
  - Move the CDN content to the users' most near access point
  - Ambulance is on the way with network requirements
- User decisions with network data
  - Chose a path with great connectivity for gaming or video
  - Chose a place for remote augmented and virtual reality

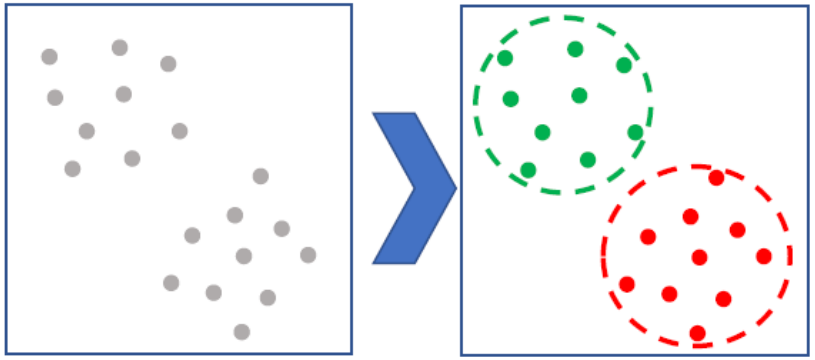
Machine Learning

- A pragmatic definition:
  - Collection of algorithms and statistical models (methods) for machines to carry out automated tasks ***based on the observation of inputs and/or outputs of a process***
- The goal of Machine Learning is to produce an estimate or a classification given a set of input values.
- We often distinguish:
  - ML method: the mechanism to train a model (neural network, support vector machine, etc.)
  - ML model: an instance of the method trained to replicate the behavior of the target process



Types of Learning

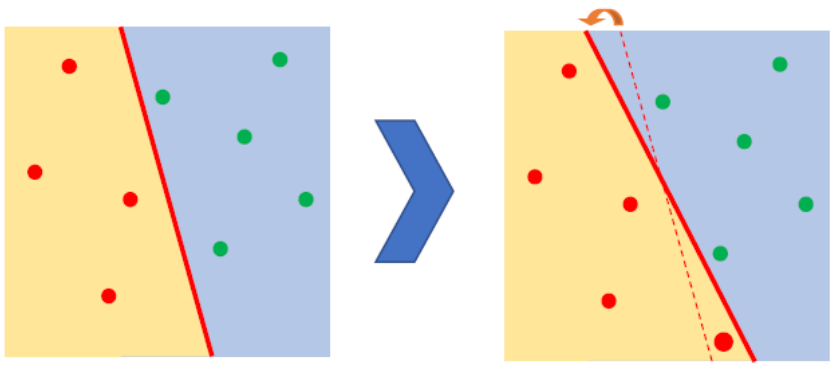
- Supervised: model is trained with a dataset of the target process
  - When training for a classification task, the historical dataset should contain the Ground Truth - the actual class of a given sample
- Unsupervised: classification or regression does not depend on prior knowledge



Unsupervised (classes are created by, e.g., finding clusters of similar data points)

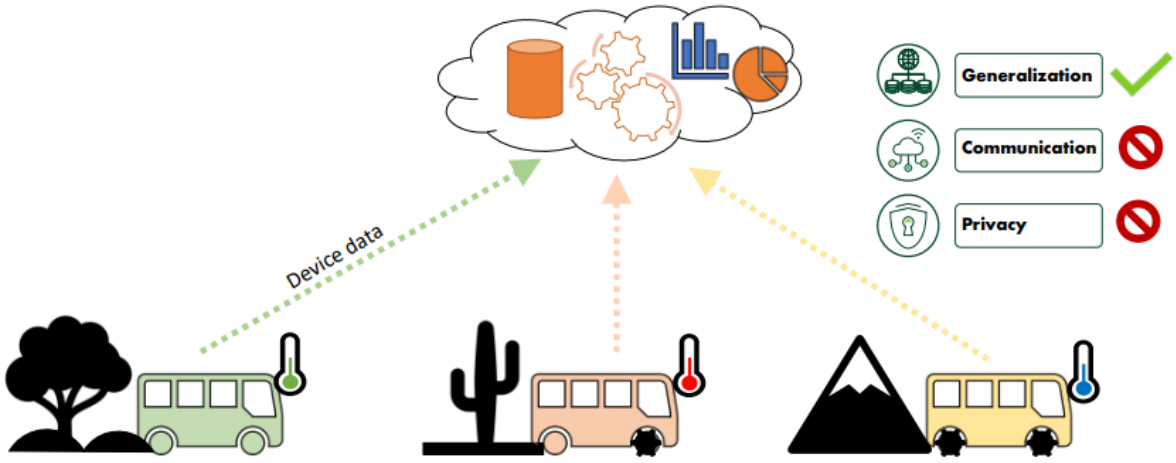
Ground Truth
necessary for training

Historical Dataset		
Feature 1	Feature 2	Class
A	1	X
B	1	X
C	2	Y



Supervised (classification depends on historical inputs)

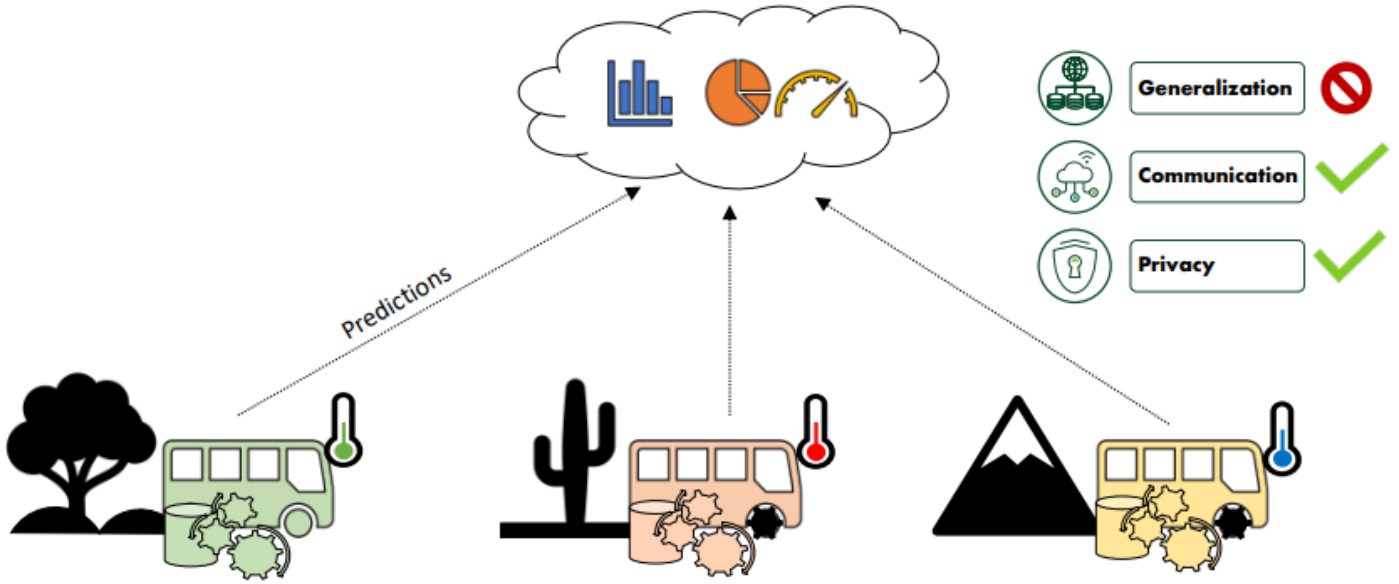
Learning: centralized



Traditional centralized learning – ML runs in the cloud, gathering info from all connected devices and sending back a model.

- The model can generalize based on data from a group of devices and thus instantly work with other compatible devices
- Data can explain all variations in the devices and their environment
- Connectivity - data must be transmitted over a stable connection
- Bandwidth – e.g. a new electrical substation could generate 5 GB/s
- Latency - real-time applications, e.g. automation, requires very low latency
- Privacy - sensitive operational data must remain on site

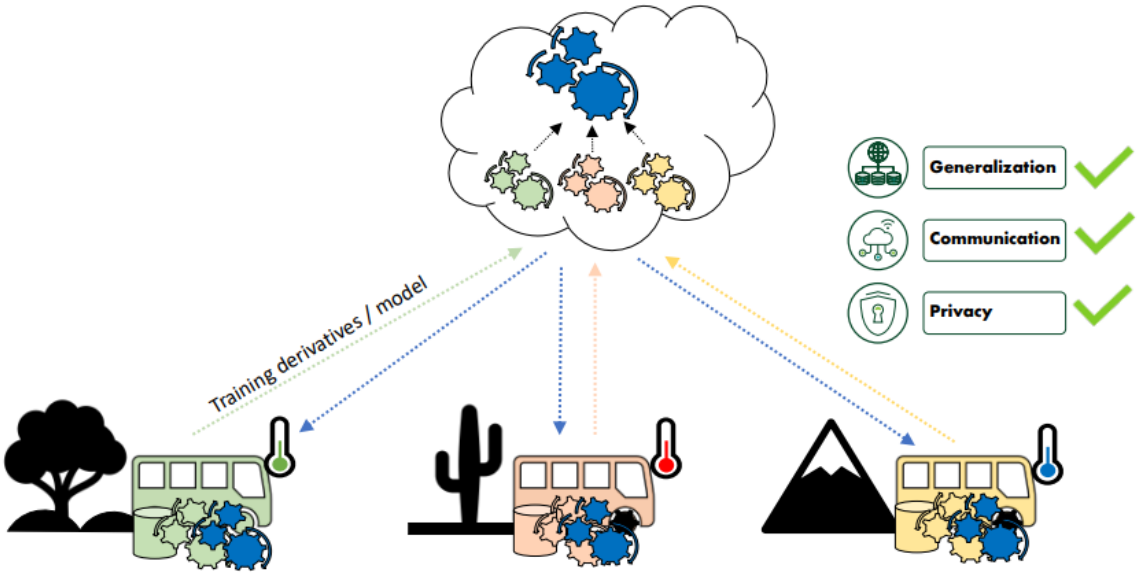
Learning: de-centralized



Edge/decentralized learning: ML continuously onboard each device at the edge of the network.

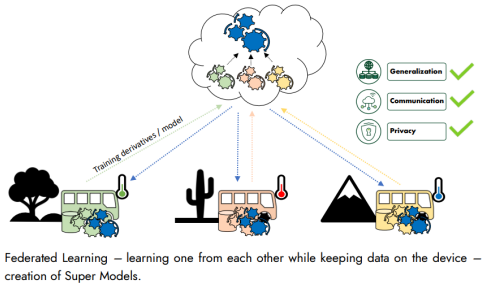
- ML that runs on site, onboard each connected device, by continuously training the ML model on streaming data, the devices learn an individual model for their environment.
- Each model only needs to be able to explain what is normal for itself and not how it varies compared to all other devices.
- Models adapt to changes over time, learning is not constrained by the internet connection and that no confidential information needs to be transferred to the cloud.
- It is not possible to get an overall view and learning

Learning: federated



Federated Learning – learning one from each other while keeping data on the device – creation of Super Models.

- ML technique to train algorithms across decentralized edge devices while holding data samples locally
- Google is the main player
- Aim to train ML models on billions of mobile phones while respecting the privacy of the users
  - Only send fractions of training results, i.e. training derivatives, to the cloud
  - Never store anything on the device



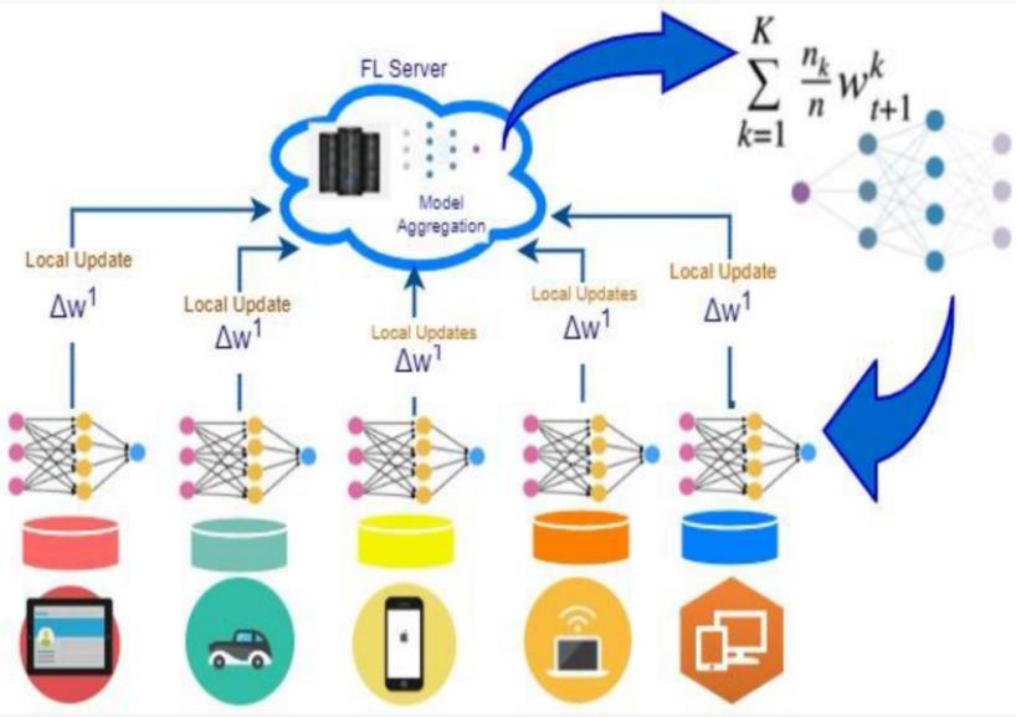
Federated Learning – learning one from each other while keeping data on the device – creation of Super Models.

Learning: federated

- When collected in the cloud, the partial training results can be assembled to a new supermodel that, in the next step, can be sent back to the devices
  - **Goolgle open source framework TensorFlow Federated**
- Model inspection – evaluation of device behavior through its model
- Model comparison – comparing models in the cloud to find outliers, super models
- Robust learning - learning can continue even if connection to the cloud is lost
- Tailored initialization – new devices can start with a model from a similar device, instead of a general super model

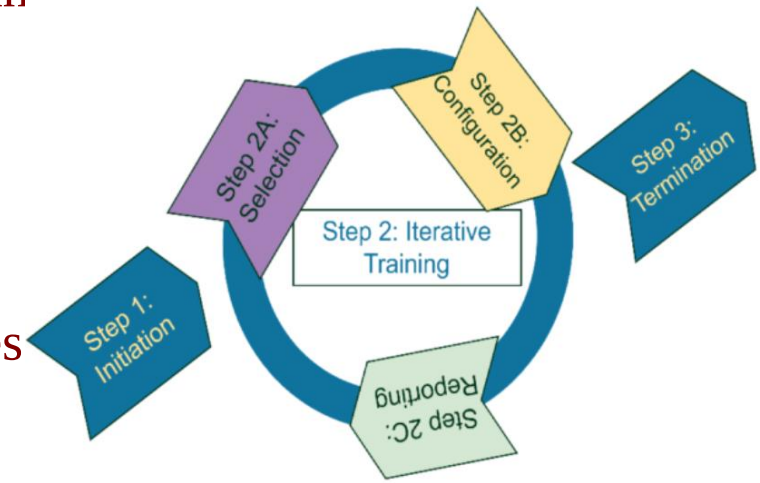
Learning: federated

- Federated learning distributes deep learning by eliminating the necessity of pooling the data into a single place
- In FL, the model is trained at different sites in numerous iterations



Federated learning: iterative

- FL employs an iterative method containing multiple client-server exchanges: federated learning round
  - Diffuse the current/updated global model state to the contributing nodes (participants)
  - Train the local models on those nodes to yield certain potential model updates from the nodes
  - Process and aggregate the updates from local nodes into an aggregated global update so that the central model can be updated accordingly
- FL server is used for this processing and aggregation of local updates to global updates
  - Local training is performed by local nodes with respect to the commands of FL server



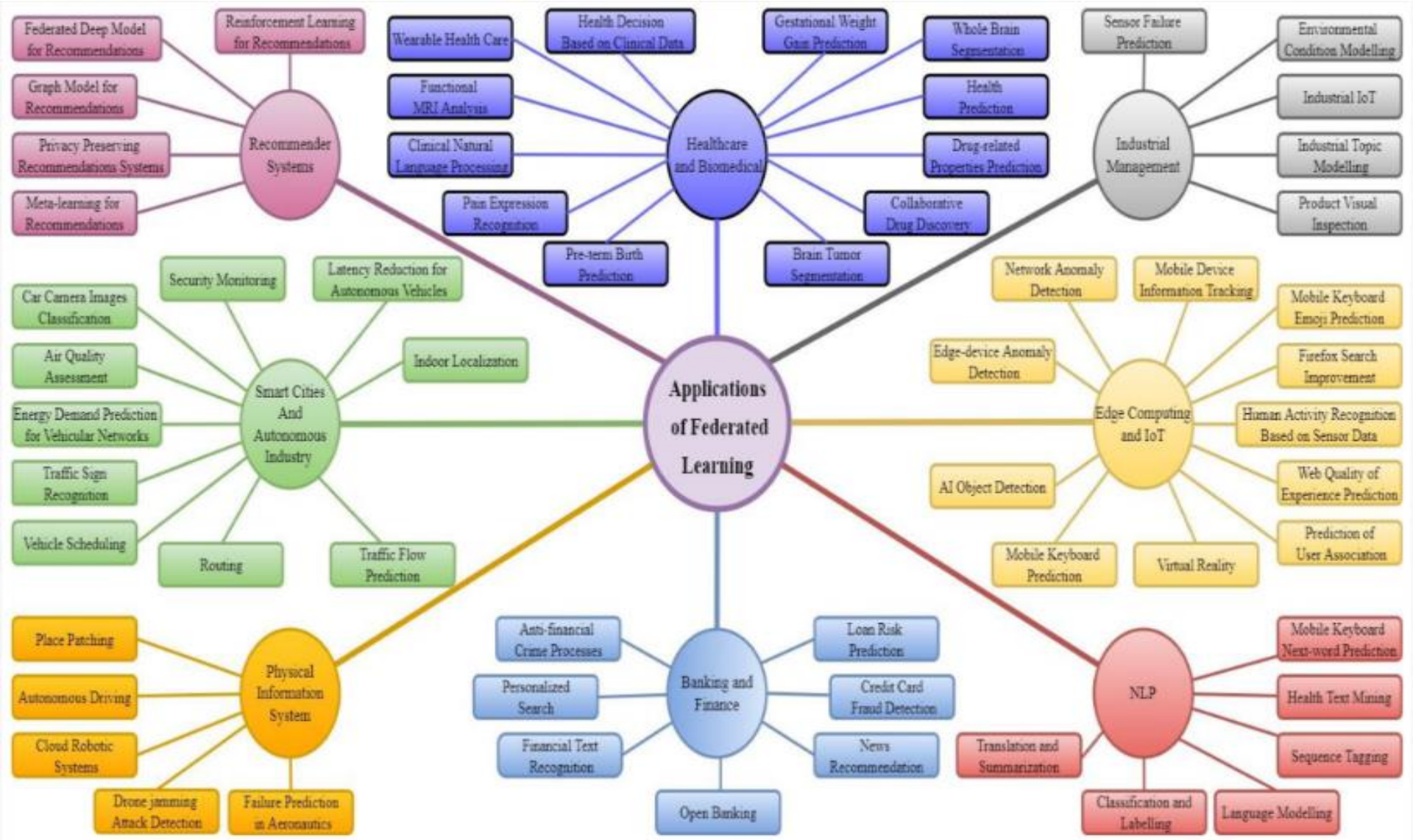
Federated learning: FedAvg

- FL is based on the “FedAvg” federated averaging method: first vanilla federated learning algorithm
- Clients update their local model with a metric learning loss that minimizes intra-class variance and maximizes inter-class variance
- The goal of each round of FedAvg is to reduce the global model’s objective ‘w’, which is just the total of the weighted average of the local device loss.

$$\min f(w) = \sum_{k=1}^{N} p_k F_k(w) \qquad where\ F_k(w)\ shows\ the\ loss\ on\ device\ k$$

- A random selection of clients/devices is taken.
- Each client receives the server’s global model.
- Clients execute SGD (stochastic gradient descent) on their loss function, in parallel, and direct the learned model to the FL server for model aggregation.
- The server uses the average of these local models to update its global model: repeated for n more rounds of communication.

Applications for Federated



Applications for Federated

Domain	Applications
Edge computing	FL is implemented in edge systems using the MEC (mobile edge computing) and DRL (deep reinforcement learning) frameworks for anomaly and intrusion detection.
Recommender systems	To learn the matrix, federated collaborative filter methods are built utilizing a stochastic gradient approach and secured matrix factorization using federated SGD.
NLP	FL is applied in next-word prediction in mobile keyboards by adopting the FedAvg algorithm to learn CIFG [93].
IoT	FL could be one way to handle data privacy concerns while still providing a reliable learning model
Mobile service	The predicting services are based on the training data coming from edge devices of the users, such as mobile devices.
Biomedical	The volume of biomedical data is continually increasing. However, due to privacy and regulatory considerations, the capacity to evaluate these data is limited. By collectively building a global model for the prediction of brain age, the FL paradigm in the neuroimaging domain works effectively.
Healthcare	Owkin [31] and Intel [32] are researching how FL could be leveraged to protect patients‘ data privacy while also using the data for better diagnosis.
Autonomous industry	Another important reason to use FL is that it can potentially minimize latency. Federated learning may enable autonomous vehicles to behave more quickly and correctly, minimizing accidents and increasing safety. Furthermore, it can be used to predict traffic flow.
Banking and finance	The FL is applied in open banking and in finance for anti-financial crime processes, loan risk prediction, and the detection of financial crimes.

TensorFlow Federated

- Open source framework for experimenting with machine learning and other computations on decentralized data.
- Locally simulating decentralized computations into the hands of all TensorFlow users.
  - ML model architecture of our choice
  - Train it locally across data by all users
- Version of the NIST dataset that has been processed by the Leaf project to separate the digits written by each volunteer.

```python
# Load simulation data.
source, _ = tff.simulation.datasets.emnist.load_data()
def client_data(n):
  dataset = source.create_tf_dataset_for_client(source.client_ids[n])
  return mnist.keras_dataset_from_emnist(dataset).repeat(10).batch(20)

# Wrap a Keras model for use with TFF.
def model_fn():
  return tff.learning.from_compiled_keras_model(
      mnist.create_simple_keras_model(), sample_batch)

# Simulate a few rounds of training with the selected client devices.
trainer = tff.learning.build_federated_averaging_process(model_fn)
state = trainer.initialize()
for _ in range(5):
  state, metrics = trainer.next(state, train_data)
  print (metrics.loss)
```

federated_learning_for_image_classification.py hosted with ❤ by GitHub

TensorFlow Federated

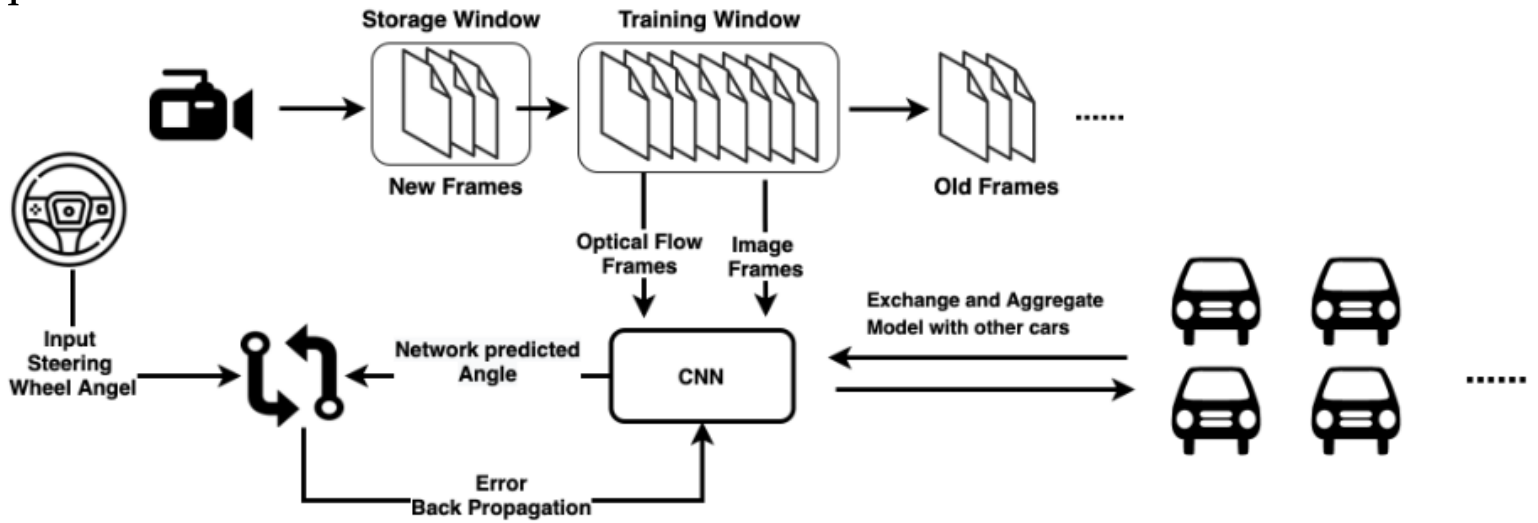
- Training an ML model with federated learning is one example of a federated computation
- Evaluating it over decentralized data is another
  - Array of sensors capturing temperature readings
  - Compute the average temperature across these sensors
- Each client computes its local contribution
- Centralized coordinator aggregates all the contributions.

```
1  @tff.federated_computation(READINGS_TYPE)
2  def get_average_temperature(sensor_readings):
3    return tff.federated_average(sensor_readings)
```

get_average_temperature.py hosted with ❤ by GitHub

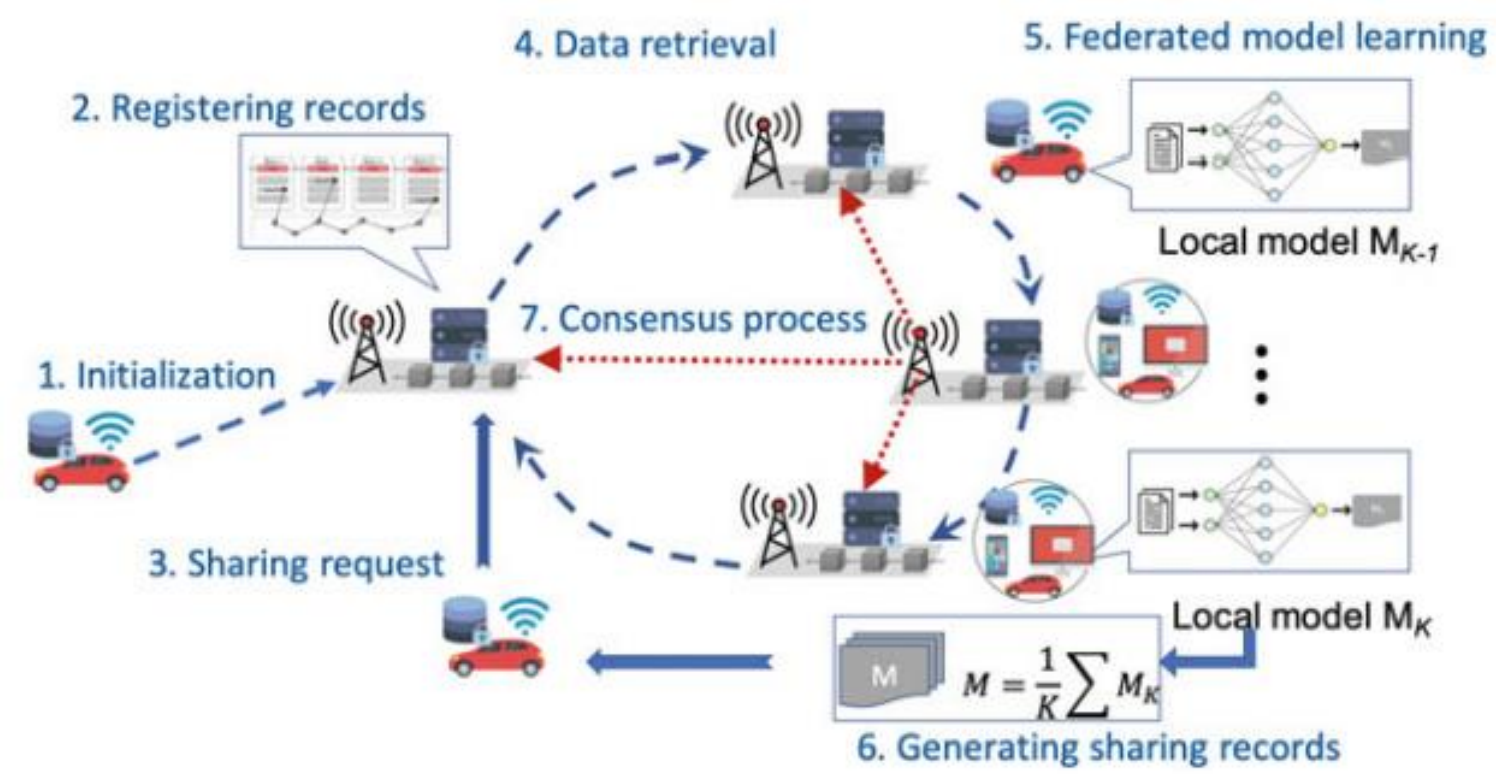
Federated learning in self-driving

- Edge vehicles compute the model locally; after completing each local training epoch, they retrieve the global model version and compare it to their local version.
- In order to form a global awareness of all local models, the central server performs aggregation based on the ratio determined by the global and local model versions.
- The aggregation server returns the aggregated result to the edge vehicles that request the most recent model.



Edge-based Federated

- MEC-empowered model sharing
  - **Edge intelligence to wireless edge networks and enhances the connected intelligence among end devices in 6G networks.**



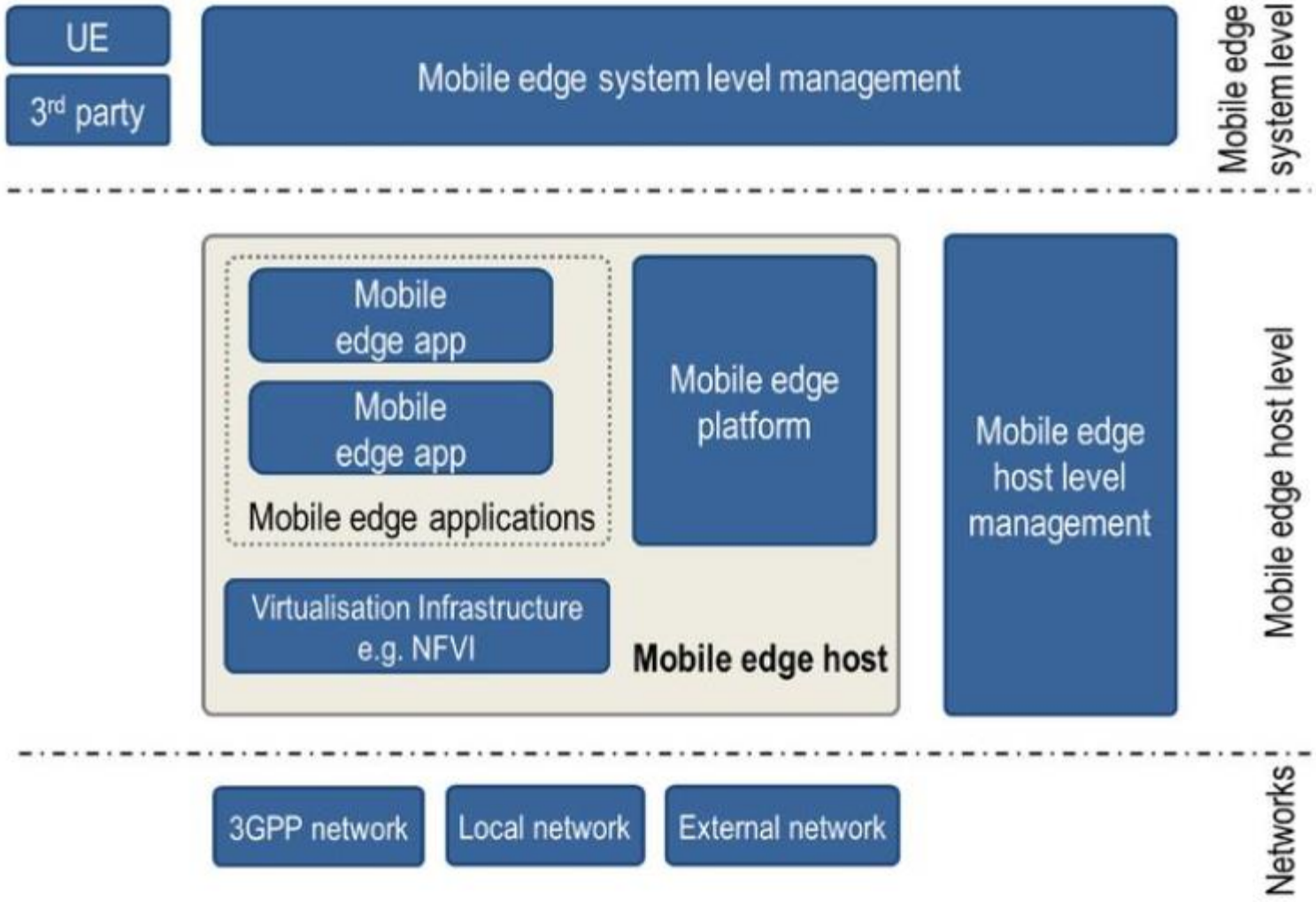
- https://www.pdl.cmu.edu/SDI/2019/slides/2019-09-05Federated%20Learning.pdf
- https://wires.onlinelibrary.wiley.com/doi/epdf/10.1002/widm.1443
- https://medium.com/tensorflow/introducing-tensorflow-federated-a4147aa20041

Mobile Edge Computing (MEC)

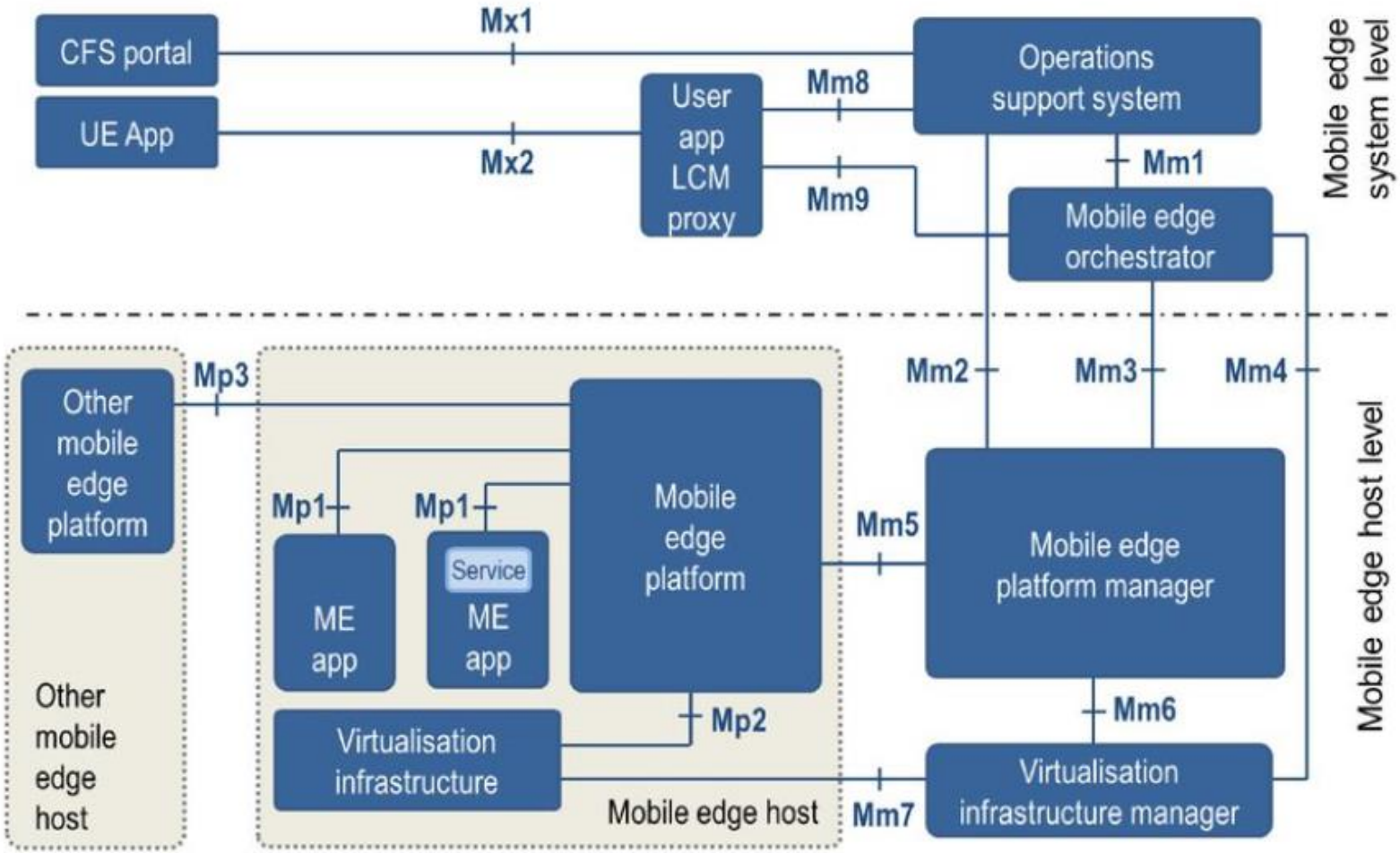
- Concept in 5G
- Brings the cloud closer to the network edge
- Opens the edge for applications from 3rd parties
- Provides services to enhance applications with context information (network information, location)
- Traffic redirection
- Ultra-low latency

- Facilitates running applications at the right location and at the right time

Edge Computing: ETSI MEC Reference Architecture

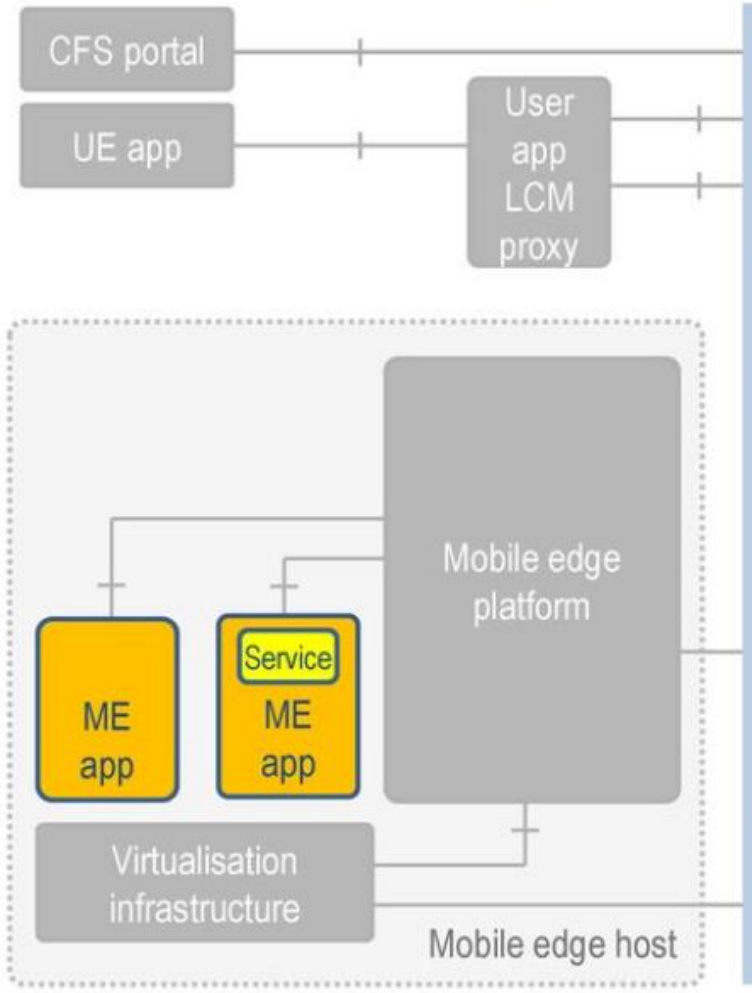


ETSI MEC Reference Architecture



* UE = User Equipment (Smartphone, Device) | CFS = Customer Facing Service | LCM = LifeCycle Management | ME = Mobile Edge

MEC: Mobile Edge Apps



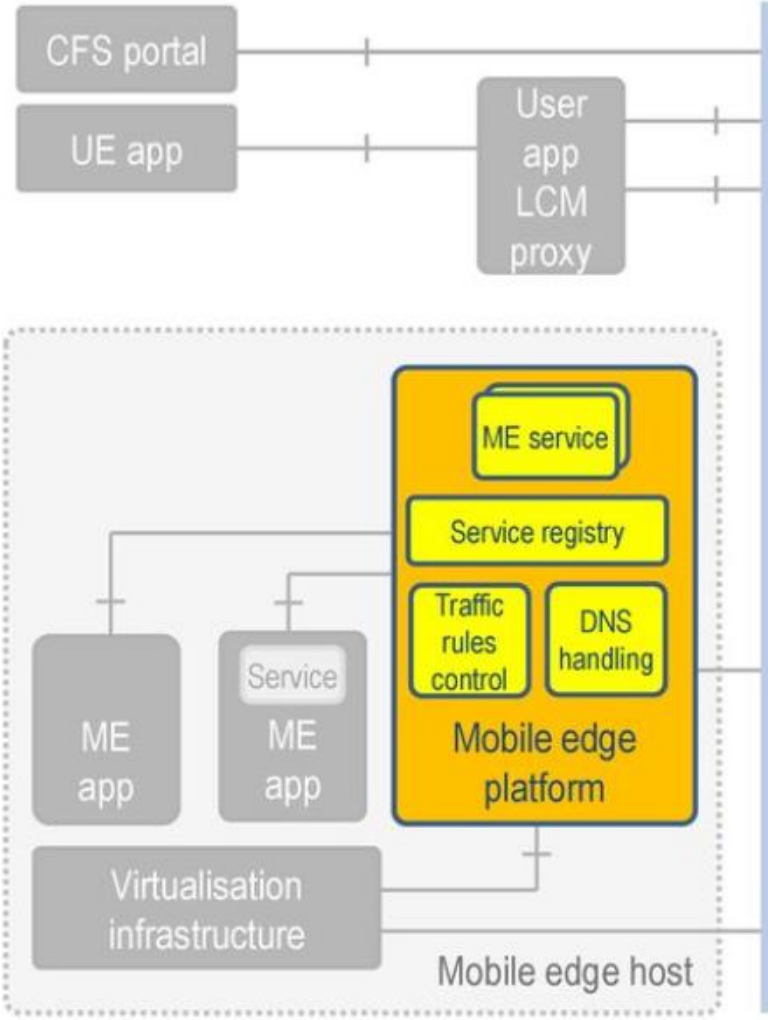
Mobile edge applications:

- Run as virtual machines
- Typically consume, but may also provide mobile edge services
- May provide information to aid the lifecycle management (e.g. indication of availability)
- Have associated rules and requirements and rules regarding:
  - DNS configuration and traffic redirection (originate/inspect/modify)
  - Compute/ Storage/Networking resources, maximum latency, required services

  The rules and requirements are validated by mobile edge system level management
- Assisted by mobility information may relocate user state
- If supported, may be relocated to another mobile edge host

Examples: handover, caching, ML processing, etc.

MEC: Mobile Edge Platform



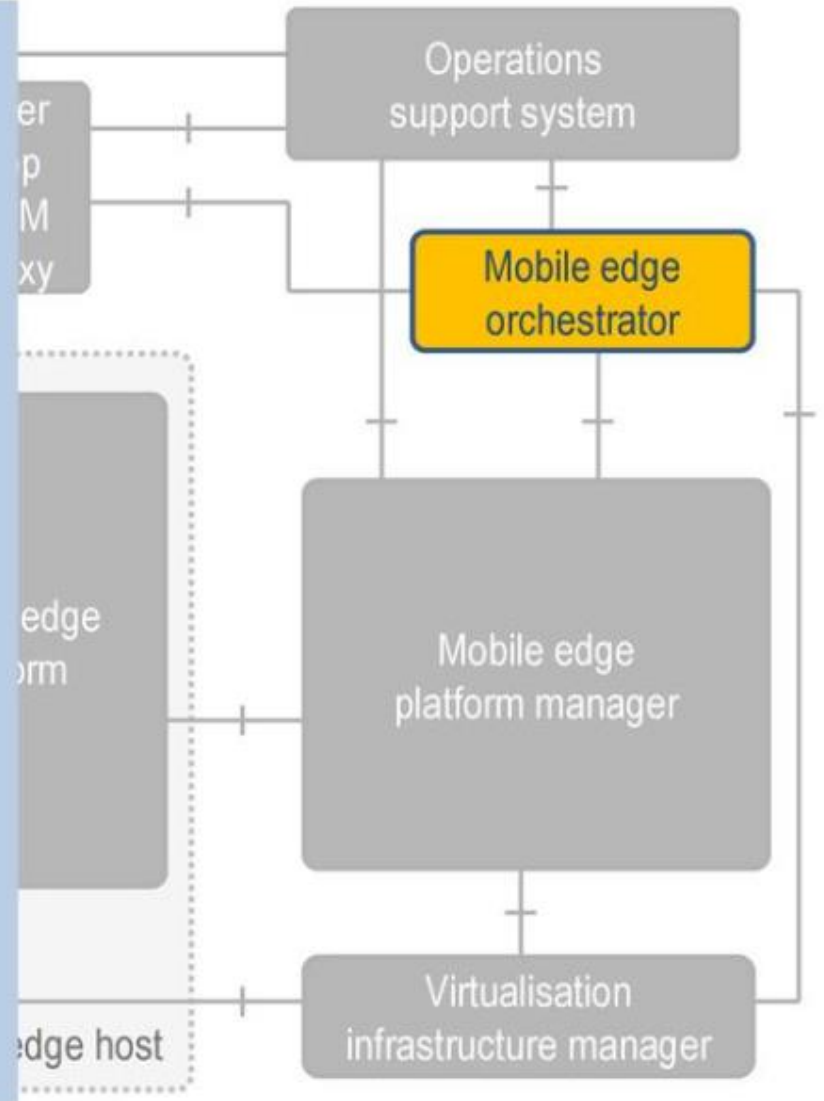
Mobile edge platform:

- An environment where applications can discover, advertise, consume and offer mobile edge services
- Controls data-plane in the virtualization infrastructure ("SDN") based on traffic rules
- Configures DNS proxy/server based on DNS records from the mobile edge platform manager
- Provides mobile edge services. May also consume mobile edge services provided by the applications
- Provides access to persistent storage and time of day information

MEC: Mobile Edge Orchestrator

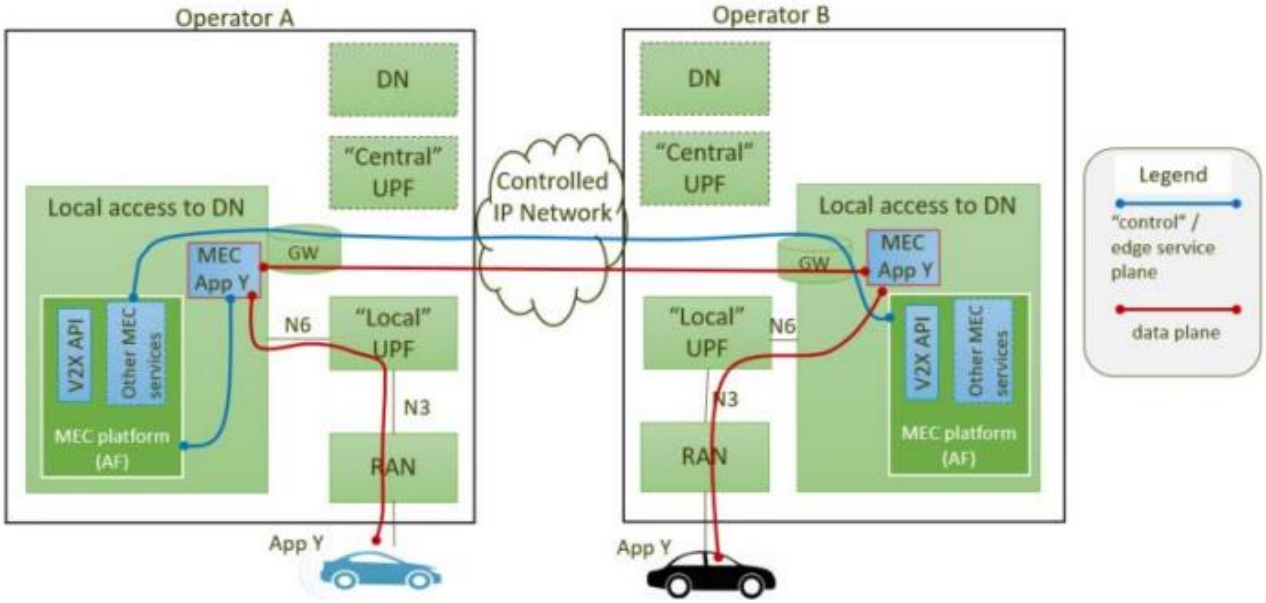
Orchestrator:

- Maintains an overall view of the system and mobile edge hosts, available resources, available services and topology
- On-boards application packages, including
  - an integrity check and authenticity
  - validation of the application rules and requirements; if necessary adjusts them to comply with operator policies
  - Maintenance of a record of on-boarded packages
  - Preparation of the virtualization infrastructure manager(s) to handle the apps
  - Selection of the appropriate host for the application, satisfying its rules and requirements
- Triggers application instantiation and termination
- Optionally, triggers application relocation



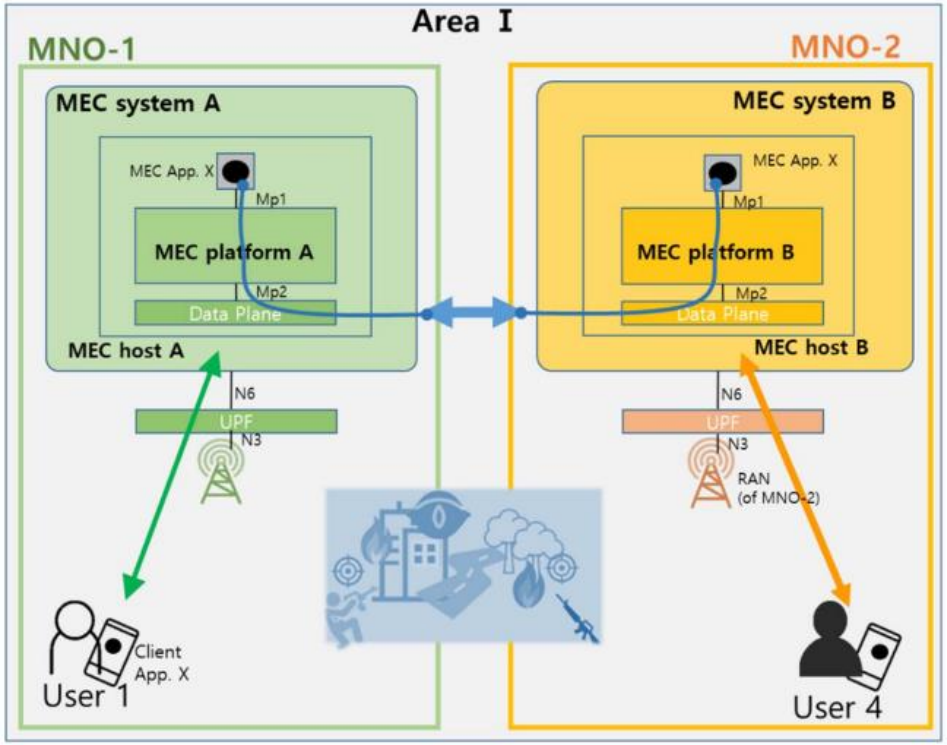
Examples using Edge Computing

- A V2X application instance may be running on a car connected to MNO 1 which is equipped with a MEC system from vendor 1, and communicating with another V2X application instance, running on a second car connected to MNO 2, which, in its turn, is equipped with a MEC system from vendor 2.
- V2X service is implemented with two instances of the "MEC App Y", each of which communicates with its corresponding Client App, i.e. "App Y", and is also connected with a MEC platform in each respective MEC system (domain).



Examples using Edge Computing

- MEC is envisioned as a promising means to deliver better Quality of Experience (QoE) for immersive AR applications
  - Reduces the delay
  - Addresses computation-intensive and battery-consuming tasks offloaded from the mobile devices.



2 MEC application Xs, instantiated on MEC hosts of MEC system A and MEC system B respectively, communicate and coordinate together for synchronizing the game scenario. Information to be exchanged between the two MEC applications for coordination mostly include users' game play actions such as players' position, movement, direction, game control and the status of game contents virtually created.

MEC future

- MEC is identified as a key enabler for IoT and mission critical vertical solutions
- Key architectural concept and technology for 5G
- Enables applications to be deployed and run in a virtualized environment
  - Services orchestrated and instantiated on demand where needed
- Myriad of different scenarios: health, industry, IoT, automotive, environment, etc
  - Change the application deployment
- Enabled autonomous networks and systems!

- https://www.etsi.org/deliver/etsi_gr/MEC/001_099/035/03.01.01_60/gr_mec035v030101p.pdf
- https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.02.01_60/gs_MEC003v020201p.pdf
- https://5gaa.org/wp-content/uploads/2017/12/5GAA_T-170219-whitepaper-EdgeComputing_5GAA.pdf
- https://link.springer.com/content/pdf/10.1007/978-3-030-83944-4.pdf