Distributed Predictive QoS in Automotive Environments under Concept Drift

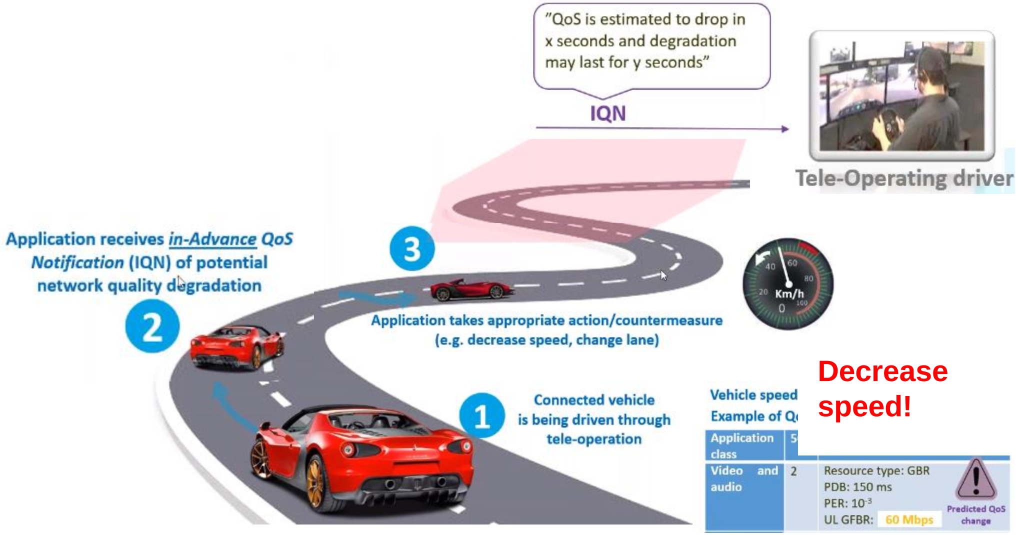
Georgios Drainakis, Panagiotis Pantazopoulos, Konstantinos V. Katsaros, Vasilis Sourlas, Angelos Amditis & Dimitra I. Kaklamani

Institute of Communications and Computer Systems (ICCS)
National Technical University of Athens (NTUA)

Making 5G proactive for the automotive industry

The concept of Predictive QoS (pQoS) by 5G Automotive Association (5GAA)

- Cooperative, connected and automated mobility (CCAM) services rely on mobile network connectivity.
- No QoS guarantees - Unexpected network conditions
- Service degradation threats **safety** & **user-experience**.



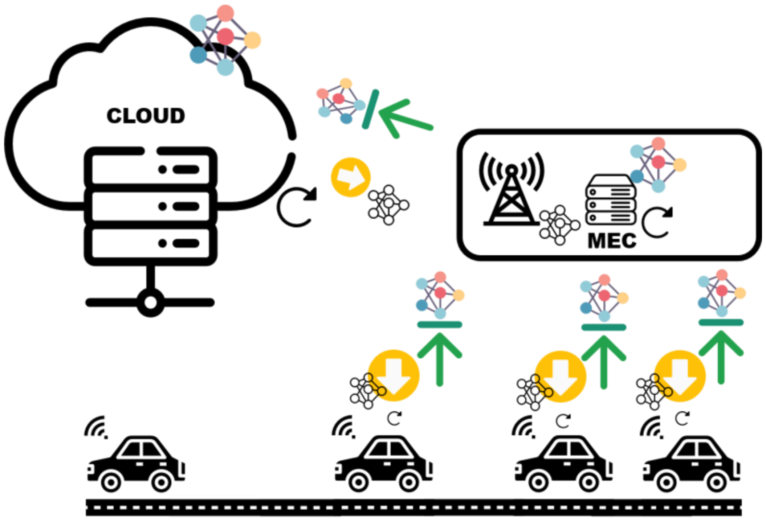
QoS Prediction → **In-advance Notification** → **App Adaptation**

Ref: https://5gaa.org/5gaa-releases-white-paper-on-making-5g-proactive-and-predictive-for-the-automotive-industry/

Towards distributed QoS prediction

The case of Federated Learning (FL)

- Volatility of cellular QoS parameters
  - AI/ML for efficient pQoS
- (Traditional) Centralized pQoS
  - Central server collects large volumes of client data
  - Centralized training of pQoS models
- (A trend towards) Distributed pQoS
  - Collaborative training by the vehicle-clients
  - Google's Federated Learning Framework (FL)
    - Data remains at the clients at all time – **PRIVACY**!
    - Communication cost reduction
    - Scalability & security

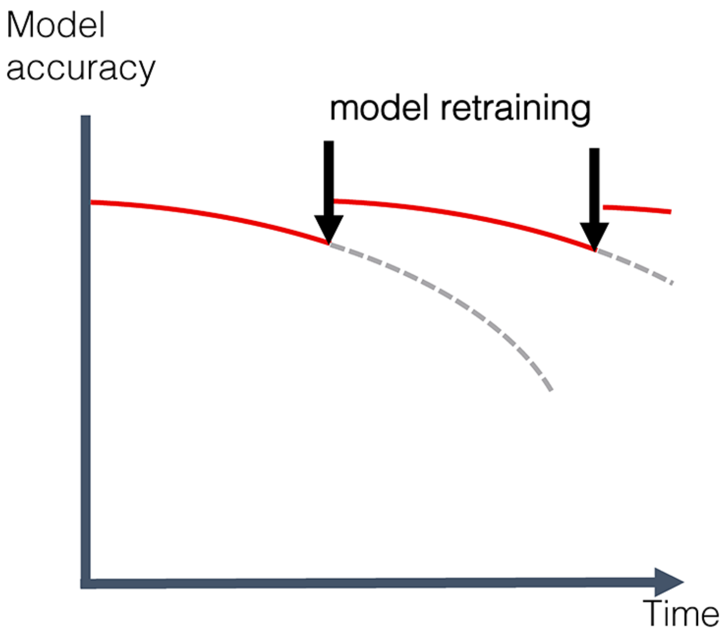


Ref: https://federated.withgoogle.com/

Training in the course of time

The problem of Concept Drift

- **Concept drift**: client data distribution changes due to seasonality, trends, user habit variations, etc.
- **Model drift**: degradation of ML model’s accuracy due to concept drift
- pQoS is shown to experience **frequent** and **severe** drifts as a result of:
  - Network/HW changes/upgrades
  - Changes of active users - population
  - User mobility patterns
  - Environmental changes



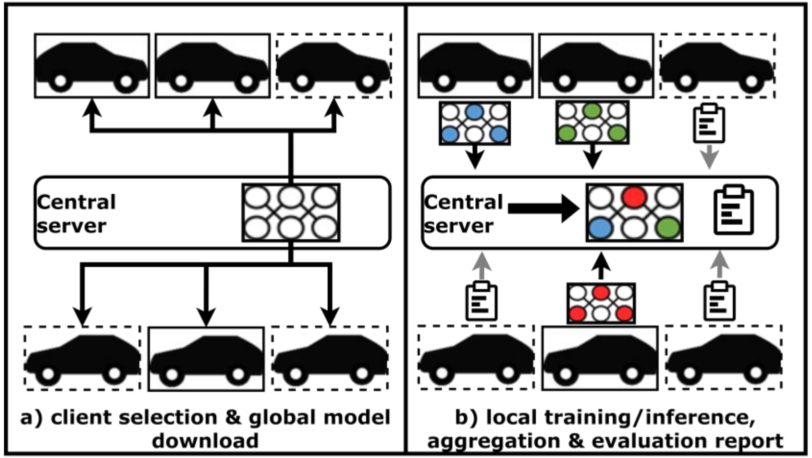
SotA: Managing concept drift

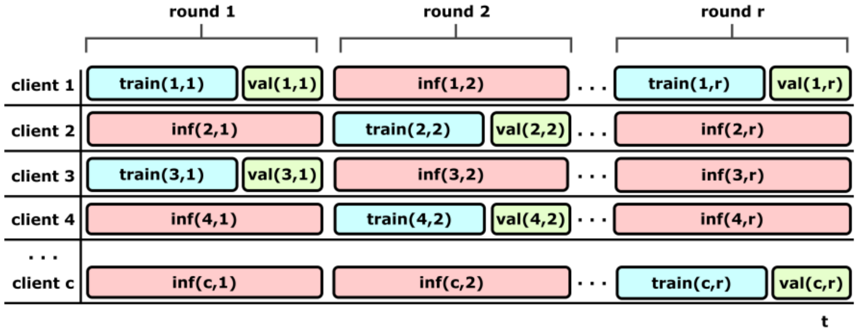
Concept Drift Management	Centralized AI/ML	Federated AI/ML	
		Techniques	**Challenges**
Detection	Access to raw data: statistical tests	Data sharing	**Privacy** violation
		Client-level detection	**Resource-constraint** clients
		Server-level detection	Low detection **accuracy**
Mitigation	Re-training, model tuning, etc.	Personalized Learning	Multiple model **maintenance**
		Async FL	Extra layer of **complexity**
		Continuous FL	Waste of network **resources**

- Research Questions
  - How to detect drift in FL, subject to FL deployment restrictions/privacy?
  - How to effectively mitigate drift in FL, w.r.t. the induced resource consumption?

Vanilla FL framework for pQoS

- FL training in equally timed rounds ***R***
- In each round random selection of
  - ***K*** clients for training (**trainers**)
  - ***M*** clients for inference/testing (**testers**).
- Round termination
  - Server collects local models from testers for aggregation
  - Server collects inference results from trainers for evaluation report
- FL termination criteria
  - convergence/accuracy threshold
  - total number of rounds
  - timeout
- <u>**No drift management mechanism!**</u>





DareFL: Main concepts

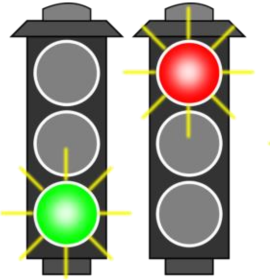
Drift-aware resource-efficient algorithm for FL (DareFL)

Stop training upon convergence

- Timely halt of training upon convergence
- Reduce resource consumption waste

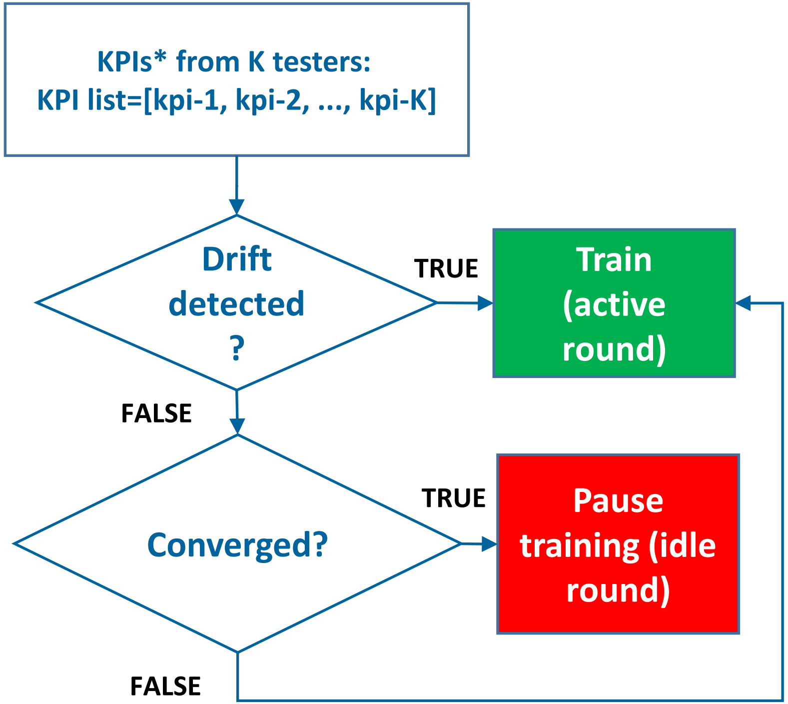
Restart training upon drift

- Accurate drift detection
- Orchestration of retraining for mitigation



- **Active rounds**: training and testing
- **Idle rounds**: testing only – clients save on their resources without sacrificing ML accuracy

DareFL: Algorithm description



*kpi stands for the % improvement of the ML model vs. a naïve predictor

Algorithm 1 $DD(\{kpi\})$

```
define DDM list: {ddm}
for each element e ∈ {kpi} do
    if e >= β1 then
        append 0 to {ddm}
    else
        append 1 to {ddm}
    end if
    return DDM({ddm}, β2, β3)
```

Drift-Detection (DD) algorithm

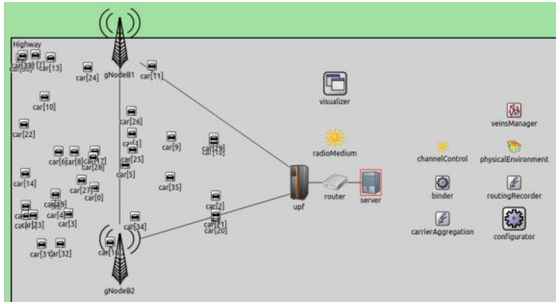
Algorithm 2 $CD(\{kpi\})$

```
define ckpi list: {ckpi}
ckpi=mean({kpi})
append ckpi to {ckpi}
if {ckpi}
    not increase(β4) then
    return boolean=True
else
    return boolean=False
```

Convergence-Detection (DD) algorithm

Simulation Environment

- Synthetic pQoS datasets with concept drift
  - Network and traffic co-simulation
  - 2x (public*) distinct drift scenarios/datasets inspired by Ericsson's Mobility report 2022
    - Network-driven (Sc1)
    - User behavior-driven (Sc2)
- (Open-source**) Distributed ML simulator
  - Pytorch-based implementation
    - Training and inference
  - Resource consumption modelling based on commercial product specs and benchmarking
    - CPU/GPU processing speed & energy consumption
    - 5G modem transmission speed & cost for uplink/downlink



Network simulator: Simu5G/OMNET

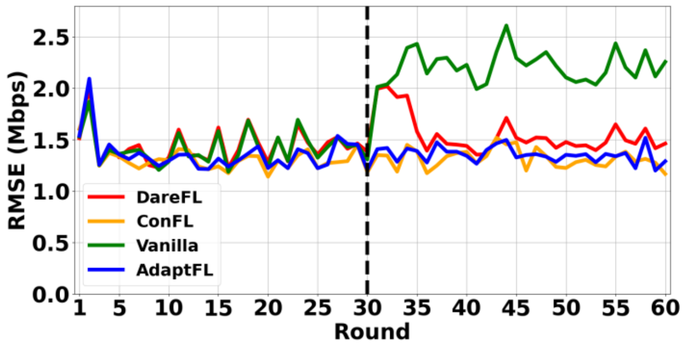
Traffic mobility simulator: SUMO/OSM

* https://zenodo.org/records/11084689

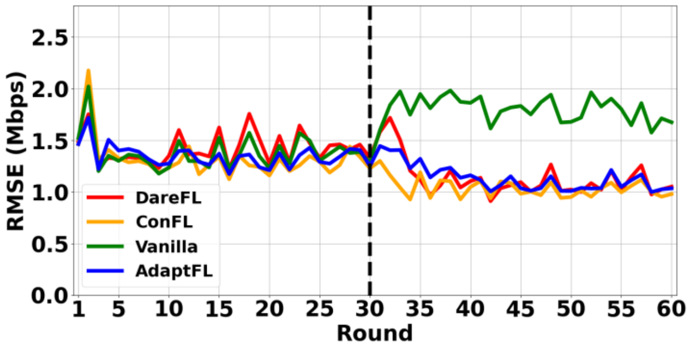
** https://github.com/gdrainakis/distributed_pqos

Results – Accuracy Metrics vs. SotA

- pQoS AI/ML task: Throughput Prediction (LSTM predictor)
- Drift: round 30 (half-time)
- Against SotA
  - Vanilla FL: 50% RMSE increase – cannot adapt to drift
  - Continuous FL (always-on): Optimal performance due to continuous training
  - AdaptFL*: Similar to ConFL
  - DareFL:
    - Sc1 - Max diff<10%
    - Sc2 - Max diff<5%



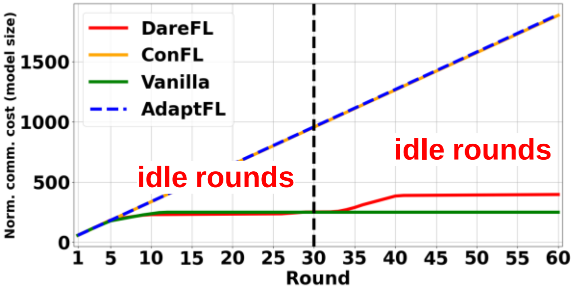
Sc1: RMSE comparison



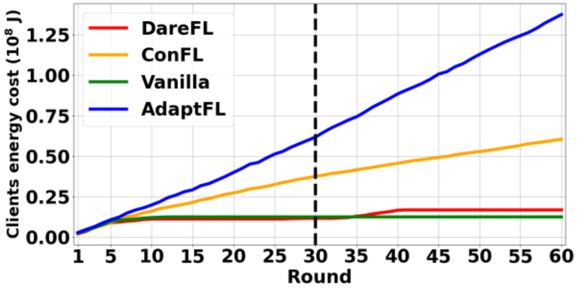
Sc2: RMSE comparison

* [Canonaco et al. Adaptive federated learning in presence of concept drift]

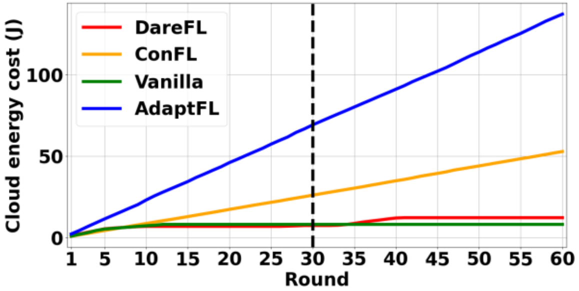
Results – Resource Consumption vs. SotA



76% lower communication costs



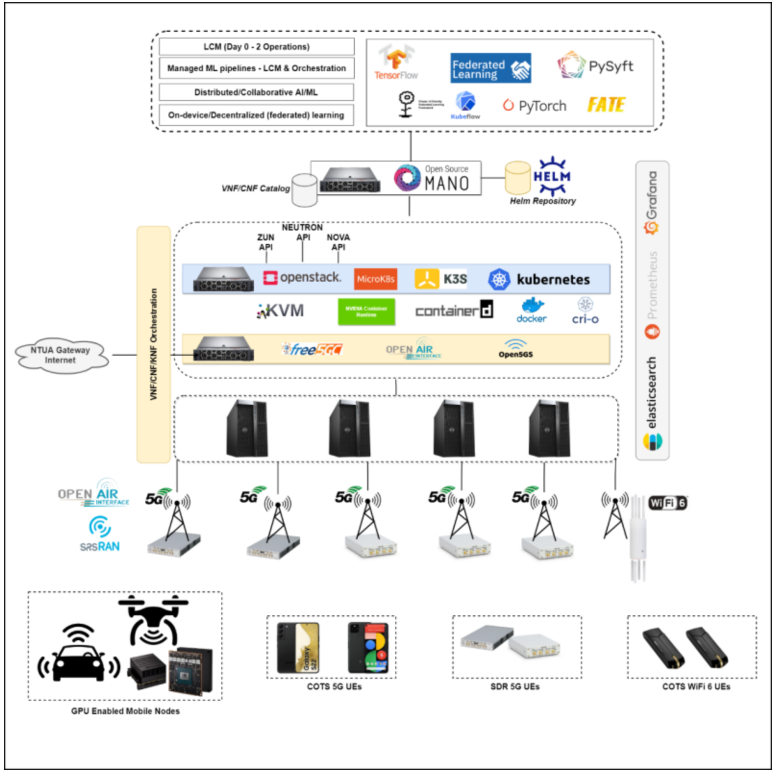
68% lower energy costs in the clients



74% lower energy costs in the server

Conclusion & Future Work

- Our contributions:
  - DareFL – concept drift management algorithm for distributed pQoS
    - Fully-aligned to FL principles
  - Similar accuracy to SotA – save up to **70%** on the network resources
  - Open-source FL simulator
  - Public synthetic datasets for pQoS under drift
- Next steps:
  - Generalize results on multiple drift scenarios
  - Cross-validation measurements on a real 5G-testbed



ICCS 5G-Testbed

Thank you!

Drainakis Georgios, Software Engineer

giorgos.drainakis@iccs.gr

Institute of Communication & Computer Systems (ICCS)

Iroon Politechniou str. 9, NTUA Polytechnic Campus

15773 Zografou, Athens, GR

www.iccs.gr

IEEE
Advancing Technology for Humanity