INTERNATIONAL TELECOMMUNICATION UNION

**TELECOMMUNICATION STANDARDIZATION SECTOR**

**Team:** *Hyperion*

**Title:** *PS1 Report on ITU WTSA Hackathon 2024*

**Contact:** Gideon Ndabuye E-mail: sndabuye@gmail.com

**Abstract:** This document contains the submission report from team Hyperion towards ITU WTSA Hackathon 2024 for the use case "Real-time Network reliability prediction".

# 1. Use case introduction: Real-time Network reliability prediction

Ensuring reliable and efficient communication in diverse network environments, including industrial, vehicular, and general-purpose networks, is essential for a wide range of applications. To address the challenge of maintaining network reliability, a real-time machine learning model can be developed to predict network Quality of Service (QoS) based on factors such as traffic patterns, resource utilization, and environmental conditions. By accurately predicting QoS in real-time, the model can optimize resource allocation, enhance adaptive routing, prevent potential service disruptions, and ultimately improve the overall Quality of Experience (QoE).

This approach aims to optimize network performance across various dynamic and demanding environments, ensuring seamless and reliable communication while preventing failures and improving user satisfaction.

## Key Challenges

- Predicting network QoS in dynamic and complex environments.
- Ensuring the model can process data and make predictions in real-time.
- Integrating the model with existing network management systems.

## Proposed Solution

A real-time ML model can be developed to predict network QoS based on various factors, including traffic patterns, resource utilization, and environmental conditions. The model can then be used to attain the below objectives.

- **Optimize resource allocation**
  Allocate resources to critical applications based on predicted QoS.
- **Adaptive routing**
  Select optimal routes for data packets based on predicted network conditions.

- **Predictive maintenance**
  Identify potential network failures or bottlenecks and take proactive measures to prevent service disruptions.
- **Quality of Experience (QoE) improvement**
  Tailor network services to individual users' needs, ensuring that critical applications receive the required QoS.

### Sample scenario

**Phase 1 -** A vehicular network is experiencing increasing congestion due to a large influx of vehicles.

**Phase 2 -** The ML model analyses real-time network data and predicts a significant drop in QoS within the next unit time

**Phase 3 -** The network management system is alerted to the predicted QoS degradation.

**Phase 4 -** The system automatically adjusts resource allocation to prioritize critical vehicular applications, such as emergency vehicle communications.

**Phase 5 -** The QoS degradation is mitigated, and the network continues to operate efficiently.

## 2. Use case requirements

1. It is critical that the ML model should accurately predict network reliability/QoS with minimal error.
2. It is critical that the model should be able to process data and make predictions in real-time, with extremely low latency.
3. It is critical that the model should be able to handle large volumes of data and adapt to changing network conditions.
4. It is critical that the model should be easily integrated with existing network management systems.

## 3. Pipeline design

### Core activities

- **Time series analysis –** analyse the time patterns from the network traffic to get general trends
- **Regression analysis –** making predictions on network reliability using QoS metrics based on input features.
- **Anomaly detection –** detecting unusual network behavior that may indicate potential issues.

## Pipeline

The figure below shows an ideal setup of the pipeline into a dynamic environment. The explanations on the components of the diagram follow.

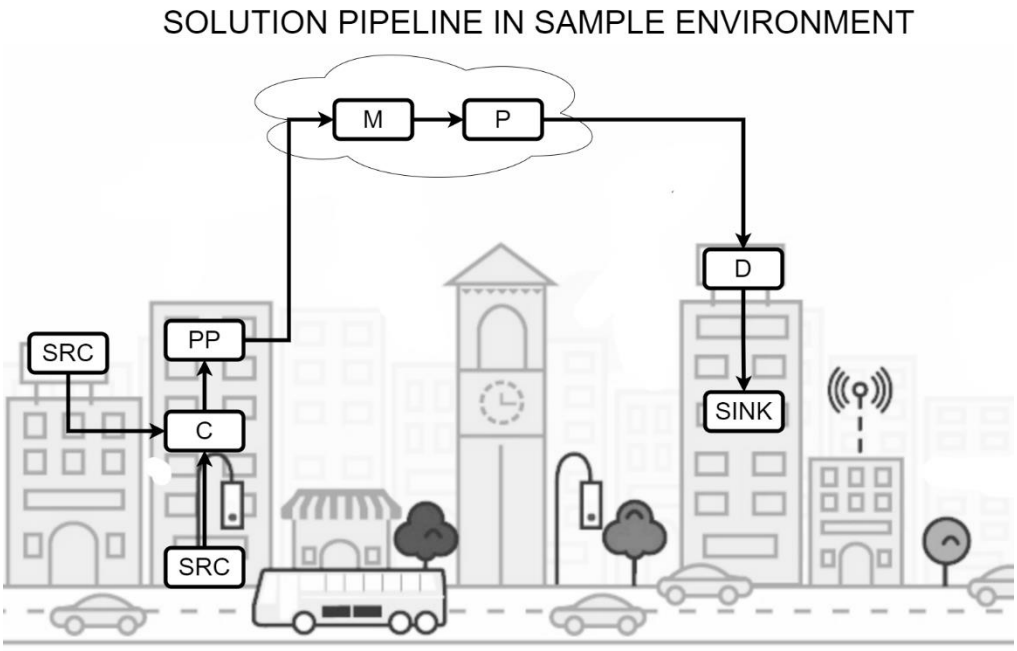


*Figure 1: Solution pipeline in a sample environment (Dynamic network environment)*

1. **SRC (Source)**
   Road-side sensors and base stations collect real-time data on traffic conditions and network performance metrics like packet loss and latency.

2. **C (Collector)**
   A central server gathers this data from all sources, ensuring it reflects the current state of the network accurately.

3. **PP (Preprocessor)**
   The data is cleaned and normalized. Outliers (e.g., faulty sensor readings) are removed, and missing data is imputed. The system then computes rolling averages for features like traffic density and latency over the past unit time.

4. **M (Model)**
   A pre-trained time series ML model predicts a QoS drop within the next unit time based on real-time network conditions and historical patterns. For instance, if traffic continues to grow, packet loss will increase, and latency will worsen, leading to degraded network performance. This can be run both in cloud and on-premises.

5. **P (Policy)**
   The network applies policies to ensure critical services like emergency vehicle communications are prioritized. Policies dictate that traffic for non-essential services is deprioritized when the network congestion is predicted. This contains predefines set of actions that the system should take based on the results output from the model.

6. **D (Distributor)**
   The system distributes the prediction to the network management system and relevant devices in the network (e.g., routers, base stations).

7. **SINK**
   Routers and base stations reallocate bandwidth, prioritizing vehicular communications and adjusting routes for packets to mitigate congestion. Emergency communications remain unaffected, while non-critical traffic is delayed or rerouted. This can be a device or software, mostly an xApp configured to take actions within the network.

# 4. Relation to Standards

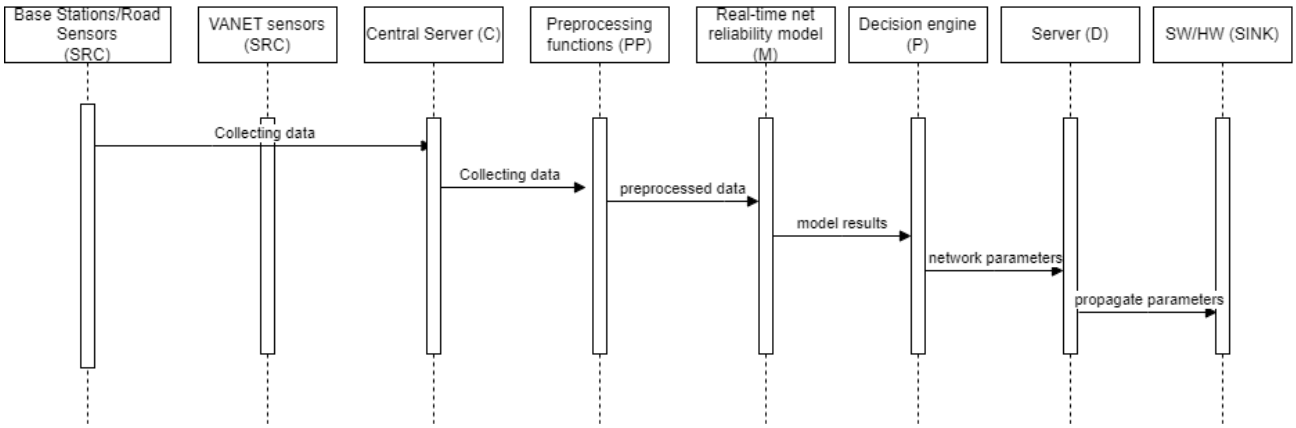


*Figure 2: Sequence diagram for the proposed pipeline*

This ML pipeline is structured across several logical nodes, each representing a distinct functionality in the process of collecting, processing, modeling, and distributing data for reliable network operations as specified in the ITU-T Y.3172 standards.

**SRC (Source) – Base Stations/Road Sensors and VANET Sensors**

The first step in the pipeline involves data collection from various network sources, including base stations, road sensors, and vehicular ad hoc network (VANET) sensors. These components serve as the source (SRC) of data, providing real-time input on network performance indicators such as traffic volume, latency, and environmental conditions. In a real-life scenario, these sensors monitor traffic conditions and resource usage, which are essential inputs for predicting network reliability.

**C (Collector) – Central Server**

The collected data from the various sensors is sent to a collector node, in this case, the central server. This server aggregates data from multiple sources, ensuring that the system has comprehensive network performance information. In real-time vehicular networks, the central server collects data from all connected sources, such as vehicles and infrastructure, and prepares it for preprocessing.

**PP (Preprocessing Functions) – Preprocessing Layer**

The data collected by the central server undergoes preprocessing, such as cleaning, filtering, and aggregation. This stage is crucial to ensure that the data is formatted and normalized before being input into the model. Preprocessing may include handling missing data, smoothing out noise, or normalizing traffic patterns. For instance, before feeding the data into the reliability model, the preprocessing functions may aggregate traffic data across various regions or time intervals.

**M (Model) – Real-time Network Reliability Model**

At this stage, the pre-processed data is passed to the model node, where a real-time ML model predicts the network's reliability based on the input features. The ML model uses time series analysis and regression techniques to predict upcoming network performance, identifying potential degradation in Quality of Service (QoS) or bottlenecks. In the vehicular network, the model predicts a drop in network reliability, which could negatively impact vehicle-to-vehicle communication or other critical services.

**P (Policy) – Decision Engine**

The output of the model is then passed to a policy node, responsible for applying network-specific policies that help mitigate any predicted network degradation. The decision engine ensures that the output of the model aligns with the network's operational requirements, such as minimizing impact on live traffic and prioritizing critical vehicular applications. Policies might include prioritizing emergency vehicle communication or adjusting resource allocations to avoid service disruptions.

**D (Distributor) – Server**

Once the decision engine has applied the relevant policies, the output is distributed to the necessary nodes, in this case, the server. The server ensures that the decisions, such as resource adjustments, are propagated throughout the network infrastructure. In a vehicular network scenario, this may involve adjusting communication protocols or rerouting traffic to ensure QoS levels are maintained.

**SINK – Software (e.g. xApp)/Hardware**

Finally, the SINK node represents the target of the ML output, such as software or hardware components that apply the parameters provided by the distributor. These components act on the instructions, making real-time adjustments to network configurations, such as adjusting bandwidth for critical services or rerouting traffic. For example, in the vehicular network, the SINK might adjust the frequency of channel measurements or resource allocations to ensure smooth communication.

# 5. Code submission details

- **Programming language:** Python
- **Libraries:** Scikit-learn, LightGBM, Multi-Layer Perceptron
- **Code repository:** GitHub

# 6. Self-Testing results

**R-squared (R2) (0.9446)**

The model explains 94.46% of the variance in network reliability data, indicating a strong fit.
The model's predictions have a Mean Absolute Percentage Error (MAPE) of approximately 35.69% on average.
Thus, the model is promising, although MAPE suggests room for improvement. Further analysis, including domain-specific metrics, data exploration, hyperparameter tuning, and ensemble methods, might enhance its performance.