ForestEdge: Unobtrusive Mechanism Interception in Environmental Monitoring

Patrick Lampe, Markus Sommer, Artur Sterz, Jonas Höchst, Christian Uhl, Bernd Freisleben
Department of Mathematics & Computer Science, University of Marburg, Germany
{lampep, msommer, sterz, hoechst, uhlc, freisleb}@informatik.uni-marburg.de

***Abstract*—A network for environmental monitoring typically requires a large number of sensors. If a longer service life is intended, it is essential that the deployed sensor systems can be upgraded without modifying hardware. Often, these networks rely on proprietary hardware/software components tailored to the desired functionality, but these could technically also be used for other applications. We present a demo of *mechanism interception*, a novel approach to unobtrusively add or modify the functionality of an existing networked system, in our case a *TreeTalker*, without touching any proprietary components. We demonstrate how a cloud infrastructure can be unobtrusively replaced by an edge infrastructure in a wireless sensor network. Our results indicate that mechanism interception is a compelling approach for our scenario to provide previously unavailable functionality without modifying existing components.**

***Index Terms*—Network Protocols, Embedded Devices, Sensor Networks, Edge Computing, Mechanism Interception**

I. Introduction

Networked systems for environmental monitoring are often based on highly specialized integrated hardware and software components with a limited feature set. Many of these systems are created for exactly one use case, with a particular focus on cost. If such systems are operated for a long time, several components might not be state of the art anymore and might lack functionality that has become relevant over time. Furthermore, lengthy and costly standardization and deployment processes may also hinder or slow down updates, which is not only valid for proprietary components.

Sensor networks for environmental monitoring were previously criticized for being too costly to operate [1]. Advances in other fields, i.e., technologies borrowed from the Internet of Things (IoT) and machine learning, including denoising methods and suitable standards for communication, allow environmental monitoring applications to become smarter [2]. Using new technology, environmental monitoring can be realized in a more cost-efficient manner. An example is the *TreeTalker* platform. It is a low-cost networked system for monitoring the health of trees in a natural ecosystem [3], which is currently in use in many areas around the world[1]. The platform consists of sensor devices called *TreeTalkers (TT)* and a base station called *TTCloud*, which can be used to relay data received via LoRa and upload it to a cloud storage provider. Our university operates 100 TreeTalkers as part of the *Nature 4.0* research project [4]. The TreeTalker platform is proprietary and does not give buyers access to the software's source code. There is also no intention from the vendor that these devices can be upgraded after purchase. The original implementation is limited in functionality and restricted to pure data acquisition. Hence, modern advances and technologies, such as machine learning methods, anomaly detection algorithms, or dynamical measurements, are not utilizable in the TreeTalker platform.

In this demo, we replace TTCloud with a versatile gateway based on open source software and components-off-the-shelf (COTS) hardware to allow dynamic reconfiguration of all TTs and to have a common basis for the automated analysis of collected data based on the novel concept of *unobtrusive mechanism interception* [5].

[1]https://www.nature4shop.com/our-vision/

II. Background

To replace *TTCloud* with our gateway, we performed a black-box analysis of a *TT's* behavior and network communication to understand the communication protocol. TTCloud's command message contains four significant values influencing a TT's behavior. *Sleep* is the time the device will sit idle between measurements (default is one hour). *heating time* is the time the TT will run its internal heater before taking the second heat-probe measurement. Finally, *time slot* and *slot length* govern the time the device will wait between measurements and sending measurement data; these values are multiplied to get the time to wait before transmitting. The waiting time between measurements is user-configurable during deployment. The time slot length will also be the same for all nodes, while the gateway will assign each TT a unique time slot. The only way for a user to access the collected data is by downloading a CSV file containing the measurement data from the vendor's web server or via a serial interface on the device. Neither a TT nor the cloud backend implements an interactive interface or an evaluation/error check of the data.

TTCloud is controlled via SMS. Collected data is sent to the cloud backend via a mobile data connection using a LoRa-based call-response protocol initiated by a TT. LoRa-certified hardware provides a dedicated network layer protocol, *LoRaWAN*, which handles collisions, addressing, and other issues resulting from the shared-medium characteristics of LoRa. However, the TT vendor decided to forgo LoRaWAN in favor of using the LoRa PHY layer directly. Thus, communication with each TT must be scheduled manually to avoid collisions. Especially in congested frequency bands, operating a TT network without collision avoidance is problematic, often resulting in poor reliability.

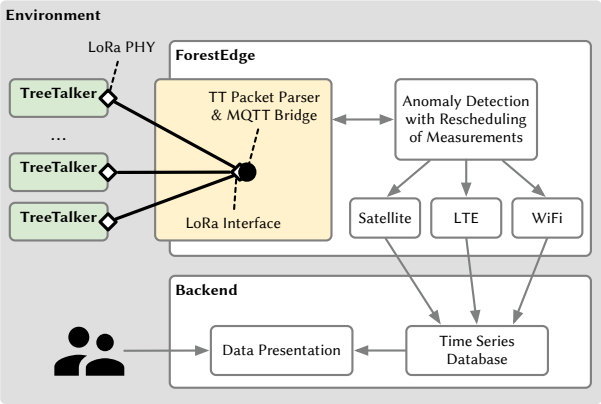

Fig. 1. TT scenario with system, environment, and interceptor

To add or modify the functionality of an existing networked system, we rely on the novel approach of *unobtrusive mechanism interception* [5]. Therefore, we distinguish between *system*, i.e., components containing proprietary or other non-changeable parts, and *environment*, i.e., components containing modifiable parts under our control. Behavioral changes are achieved by functionality-enhancing yet unobtrusive *interceptors*, i.e., hardware/software components introduced between the environment and the system to add or update mechanisms to represent some functionality. To showcase the use of mechanism interception, we illustrate our approach on a specific proprietary hardware system, namely the *TT*. We call the resulting interceptor *ForestEdge*.

III. ForestEdge

With the insights gathered from our black-box analysis, we can fully replace TTCloud with our open-source alternative, which allows us to bring mechanism interception to the TT network. By intercepting and redirecting TT messages, we can increase both the functionality and the reliability of the TT network: (a) we can make the system interactive, (b) we can perform statistical analyses on measurement data to find and rectify anomalies by triggering repeated measurements, and (c) we can maximize the battery lifetime by adjusting the measurement frequency to the remaining battery power.

According to the approach of unobtrusive mechanism interception, the following components are present: (a) The TT is the *system*; its goal is the collection of accurate environmental data and to make it available to users. (b) The *environment* is a forest-wide LoRa network that serves as the gateway to connect the set of TTs to the Internet. It receives the TTs' data via LoRa and can process and relay it to other systems in the Internet. ForestEdge is modular and allows the use of arbitrary communication technologies, such as WiFi, LTE, 5G, or a satellite uplink for data transmission. (c) The *mechanism* that allows the set of TTs to talk to the system is LoRa. (d) *ForestEdge* is the *interceptor*. It replaces TTCloud, communicates with the set of TTs via LoRa, and ultimately allows users to access the data in a more convenient format. Furthermore, it enhances the system's functionality by introducing anomaly detection algorithms, improving the data quality. This approach is unobtrusive, since the TTs cannot distinguish between TTCloud and ForestEdge.

Fig. 1 shows the architecture of ForestEdge. It is based on two separate decision-making entities. The local decision engine is located on the physical device, in this case, a Raspberry Pi, in the forest and communicates directly with a set of TTs. It only has data from its directly connected sensor nodes. This component is labeled *ForestEdge* in the figure; it consists of the actual interceptor, shown in the yellow area, which includes the LoRa transceiver and a translation layer that re-marshals the packets and sends them on via MQTT. The anomaly detection algorithm checks if a measurement seems reasonable, and if successful, the packet is transmitted onward via different communication technologies. ForestEdge decodes LoRa packets from a TT and submits the measurement data via MQTT. When the local decision engine receives data via MQTT, it stores it locally and forwards it to a global aggregator, which periodically aggregates data from all the nodes and computes relevant benchmark values. When generating a reply packet, the local decision engine performs a statistical evaluation of the corresponding TT's previous measurement data and how it relates to the aggregator-supplied benchmark values. For simplicity, we decided to use a statistical analysis where a value is labeled as an "anomaly" if it deviates from the expected value by more than two standard deviations. The expected value is computed from the ForestEdge node's local data and aggregated values supplied by the backend. However, employing a more sophisticated analysis method (e.g., based on machine learning) is possible with little to no modification of the architecture. If ForestEdge detects an anomaly, a TT is required to rerun its measurements by sending a command packet with an updated shorter sleep time.

The second decision-making entity, labeled *Backend* in Fig. 1, is a global decision engine, i.e., a centralized aggregator in the backend. It has global knowledge of all nodes in the network. The backend receives data from all ForestEdge instances and saves them in a time-series database to aggregate and redistribute them to aid stations in making statistical evaluations. It also contains a presentation layer through which users can view the data.

IV. Anomaly Detection at the ForestEdge

A disadvantage of the TT devices is the fluctuating quality of on-device measurements. To compensate for this shortcoming, we implemented local anomaly detection, which can trigger a new measurement. To make meaningful decisions, we compare a long-running and a short-running time window of the data. If we assume the data is normally distributed, the calculation of the twofold standard deviation for short-running windows is already sufficient to make decisions, e.g., whether a new measurement is necessary or these new values can be appointed as the new standard. In general, the preferred decision is to reduce the upcoming measurement interval to verify and increase the accuracy of the collected data. If no anomaly is found, the sleep time until the subsequent measurement