tinyDSM-Based Modular Approach in Facilitating IoT Application Development and Maintenance

Przemysław Zielony, Krzysztof Piotrowski
IHP - Leibniz Institut für innovative Mikroelektronik
Frankfurt (Oder), Germany
{zielony, piotrowski}@ihp-microelectronics.com

***Abstract*—As sensor network technology advances, scalable and flexible software design is crucial. This paper presents a modular middleware based on tinyDSM, using adapters and services. This simplifies development by reusing modules, accelerating WSN implementations. Adapters facilitate communication between tinyDSM, microcontroller peripherals, and sensors, supporting new module integration. Edge computing processes data locally, reducing network traffic and enhancing resource efficiency. Effective WSN management requires robust node monitoring. tinyDSM's services and adapters meet diverse needs, ensuring precise control over node and network operations. State machines at all software levels improve node monitoring, boosting system reliability and fault tolerance. This approach scales to varied conditions, optimizing resource use across applications. User notifications and data sharing enhance communication and troubleshooting. This paper underscores modular approaches in scaling edge device applications and node monitoring's critical role in WSN reliability and efficiency.**

***Index Terms*—Wireless Sensor Networks, tinyDSM, modular, middleware**

I. Introduction

With the dynamic evolution of the Internet of Things (IoT) and the increasing demand for efficient data management, developing data-centric applications in sensor networks has become crucial. Edge devices, characterized by limited computational power, memory, and energy, necessitate a modular approach to application development. This allows for the construction of applications by combining smaller, self-contained modules, simplifying the development process and ensuring system flexibility and scalability.

In wireless sensor networks (WSNs), data is collected, processed, and transmitted in real-time, posing a challenge for efficient data management and system performance optimization. Modular development of data applications makes systems more adaptable to changing conditions. This paper presents a methodology for designing and implementing data applications for edge devices, emphasizing modularity's role in achieving these objectives.

Effective node monitoring in WSNs is essential for diagnosing failures, managing energy efficiently, and ensuring data quality. It optimizes data routing by addressing transmission issues, enhances network security by detecting threats, and helps in resource management by tracking node resources like bandwidth and processing power. Node monitoring ensures reliability, efficiency, and security in WSNs.

II. State of the Art

In wireless sensor networks, there is no single middleware that would be adapted to all microcontrollers and applications, which forces developers to choose the appropriate approach depending on the application requirements. Among the different types of middleware, different approaches are used, e.g. modular, database-based and service-oriented architecture (SOA).

The modular approach, represented by Impala[1], is ideal for dynamic topologies and mobility in WSNs. Impala, used in the ZebraNet project, is a lightweight, modular middleware that supports over-the-air software updates and dynamic adaptation of applications to device parameters and failures, optimizing energy consumption and increasing system reliability.

The database approach treats WSNs as a distributed database system, enabling structured queries on sensor data. TinyDB[2], running on the TinyOS operating system, offers an SQL-style query interface, simplifying access to data from distributed sensors. Although TinyDB is limited in handling spatiotemporal relationships and real-time data rendering, it is effective in query processing in sensor networks, providing fast, yet approximate, results.

Service-oriented architecture (SOA), exemplified by EnviroTrack[3], focuses on creating flexible and scalable service-based systems. EnviroTrack is an environmental target tracking middleware that supports distributed computing and efficient information management, enabling applications to monitor and label environmental phenomena such as vehicle movements. This system provides high scalability and flexibility in managing WSN applications.

III. tinyDSM modular concept

TinyDSM[4] is a middleware designed for wireless sensor networks (WSN) and embedded devices, focusing on data management and modularity. TinyDSM uses tuple spaces to store data, which allows for a distributed and configurable data storage architecture and provides a Data Interface for software components to access and store data. This is a more advanced approach compared to TinyDB, which offers a SQL-style query structure, but has limitations in handling time-space relationships. TinyDSM initially supports communication via a localized mesh topology for intensive data replication, offering distributed data storage with configurable reliability.

The core of tinyDSM is designed to be adaptable, featuring an adaptation layer that allows to be ported to various MCU software platforms which is more advanced than EnviroTrack, which focuses on monitoring and managing environmental data but does not offer such a wide range of hardware integration options. Each tinyDSM instance is configured with a specific list of variables tailored to the application, ensuring an optimal implementation. Applications using tinyDSM are composed of simple modules called services and adapters, which focus on data access and processing, facilitating easier testing and code reuse. Compared to Impala, which is also modular, tinyDSM offers more flexible possibilities for adapting and modifying the system.

Adapters convert other interfaces to the Data Interface, enabling data exchange with external systems or hardware. For example, adapters can be used to create software drivers for sensors or actuators, managing data import and export through specific external interfaces. Services, on the other hand, process the data within tinyDSM and generate new data. This modular architecture simplifies the development and integration of data-centric applications in WSN and embedded environments.

A. Services

Services in the context of tinyDSM are modules with simple functions that process data available in middleware. Their main task is to read data from tinyDSM, process this data, and then write the results back to the middleware. Thanks to this approach, services can perform specific data processing operations depending on the application needs.

Services act as key components in a tinyDSM-based architecture, working with adapters that translate external interfaces to a data interface, enabling data exchange with external systems or peripheral devices. Services can be configured during compilation, but can also be adjusted in real time using configuration variables, which allows for modification of the operating parameters of function modules during program execution. An example would be a variable controlling the sampling frequency of a given sensor.

Additionally, services can implement the concept of virtual sensors. This means that they can aggregate data from many physical sensors and combine them into a single value. This approach allows for effective monitoring of various parameters and provides valuable information for applications related to building automation, health management and environmental monitoring.

The services in tinyDSM provide flexibility and support in adapting applications to changing conditions, which is crucial for the effective management of sensor networks and for the implementation of various applications in WSN environments.

B. Adapters

Adapters in the context of tinyDSM are transformation modules that act as an intermediary in communication between middleware and peripherals such as sensors or other external devices. The adapters translate various interfaces into the Data Interface, enabling data exchange between external systems or hardware blocks and tinyDSM.

Each sensor in the application is typically managed by a microcontroller using a driver provided by the device manufacturer or developed by WSN developers. Due to differences in the operation and functionality of sensor drivers, they cannot be universally integrated with all microcontrollers. Adapters allow these drivers to be adapted to work with tinyDSM middleware, providing interoperability and flexibility to support different devices by making interaction data-centric.

In the proposed architecture, adapters are key elements that can be created or used to shape endpoints associated with data flow paths in applications. All different data sources can be read using their specific drivers, and the data representing the measurements is then stored in measurement variables within the tinyDSM middleware.

Adapters are defined at the compile time, but their operating parameters can also be modified during program execution using configuration variables, which allows for greater flexibility in application development and adaptation to changing conditions.

C. State Machines

State machines are a fundamental aspect of the proposed approach for maintaining Wireless Sensor Networks (WSNs) using the tinyDSM middleware. They provide a structured way to monitor and control the operations of various software components within the network, ensuring reliable and fault-tolerant performance. In the context of WSNs, state machines help monitor and control the status and behavior of sensor nodes and their components. By defining specific states and transitions, state machines enable precise tracking of node operations, facilitating error detection and system management.

In the proposed approach, state machines are implemented at every level of the software architecture. Each program block, from sensor drivers to data processing services, incorporates a state machine to monitor its status and transitions. The states of these machines are stored in tinyDSM variables, which are used for monitoring and verification purposes. The primary function of state machines in this system is to provide a detailed view of the operational status of each component. This includes monitoring data collection from sensors, tracking sensor interfaces, overseeing data processing tasks, and managing data transmission over the wireless network.

IV. Use Cases

A. WSN MAN smartRiver

Designing a Wireless Sensor Network (WSN) for city-wide coverage involves key considerations for reliable and efficient monitoring. The network must be scalable, supporting many sensor nodes and allowing easy addition of new ones. Ensuring complete coverage of all city areas, including high-traffic and critical infrastructure, is essential, with sufficient sensor density for detailed data collection. Reliable connectivity is crucial, requiring sensors to maintain strong communication links despite urban obstacles. Multi-hop communication and

mesh networking can help. Energy efficiency is key due to battery-powered sensors, necessitating power-saving protocols and possibly energy-harvesting technologies like solar panels.

Key design considerations include optimal node placement based on thorough site surveys, choosing suitable communication protocols, and possibly using repeaters to bridge network segments. Power management strategies focus on low-power hardware and efficient communication, while security measures involve encryption, authentication, and authorization.

Data processing can utilize edge computing to reduce transmitted data and employ machine learning for predictive maintenance and anomaly detection.

An example of such a network is the smartRiver project, covering the cities of Frankfurt (Oder) and Słubice. Initially designed without the tinyDSM middleware, it provided data from hydrological and meteorological measurement stations. Given the extensive area, repeaters were necessary for wireless data transmission. All devices are battery-powered and recharged by solar panels. The modular design ensures similarity, but each station features different sensors, modules, sensor read frequencies, and power management systems.

During the project, various errors were observed that middleware tinyDSM middleware could address. Correct placement of stations was problematic. Stations needed sufficient sunlight exposure, accurate placing for proper data collection, and reliable wireless network access, which was challenging in diverse terrain. Many stations failed due to insufficient energy supply. Using tinyDSM and state machines, such errors could be detected much faster.

Sensor reading errors occurred due to various communication protocols. Implementing state machines could diagnose issues by allowing log readings from individual stations. Middleware based on tinyDSM could speed up the configuration process through modularity, facilitating the addition of adapters for sensors and services responsible for data processing. Basic configurations related to energy management, node operating modes, and sensor readout frequencies would be simpler with tinyDSM middleware, accelerating deployment and enhancing efficiency.

B. WSN LAN Smart Home

Designing a Wireless Sensor Network (WSN) for a smart home environment requires distinct considerations to ensure seamless and efficient monitoring tailored to residential needs. Unlike a city-wide network, a smart home WSN requires fewer sensor nodes, focusing on localized and detailed data collection within a confined space. The nodes in a smart home network generally monitor fewer parameters, simplifying data management.

Coverage in a smart home WSN must ensure that all key areas, such as living spaces, kitchens, bedrooms, and outdoor areas, are adequately monitored. Given the proximity of sensor nodes within a home, connectivity is generally more straightforward than in a city environment. Wireless networks in a home do not need to cover large distances.

Energy efficiency remains key, but the power management strategies differ. Many smart home devices are connected to the home's power supply, reducing the dependence on battery power. For battery-powered sensors, efficient protocols and sleep modes are essential to extend battery life, though frequent recharging is often possible in a home setting. tinyDSM can help monitor and optimize energy consumption in a smart home. With services that process data in real time, it is possible to dynamically adjust the operating parameters of home devices. For example, occupancy sensors can be integrated with the heating system to automatically regulate the temperature in rooms that are currently in use. State machines can monitor the status of each device, adjusting its operation to minimize energy consumption.

Data management in a smart home WSN involves real-time data processing, aggregation, and filtering to provide actionable insights to homeowners. Security is a top priority, with encrypted data transmission and secure storage to protect against unauthorized access and ensure privacy.

State machines implemented at the level of each element of the Smart Home system allow for ongoing monitoring of the status of devices and identification of potential problems. tinyDSM allows for faster detection and diagnosis of failures, which is crucial for maintaining the continuity of security systems such as alarms or electronic locks. If a problem is detected, the system can automatically take corrective action or notify the user.

Reliability and fault tolerance in a smart home context ensure continuous operation of essential systems, such as security alarms and environmental controls. Interoperability is critical, as the network must support a variety of devices and protocols, integrating seamlessly with home automation platforms like Google Home, Amazon Alexa, or Apple HomeKit. In many cases, multi-hop wireless communication is unnecessary due to the limited range within a home.

In summary, designing a WSN for a smart home environment requires a focus on localized coverage, reliable connectivity, and enhanced security. By addressing these factors, the network can provide a seamless and intelligent living experience, improving comfort, convenience, and security for homeowners.

REFERENCES

[1] T.Liu, M. Martonosi, Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems. PPoPP'03, San Diego, Cali-fornia, USA, June 2003.

[2] S.R. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong. Tinydb: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. (TODS), 2005.

[3] T. Abdelzaher, B. Blum, Q. Cao, D. Evans, J. George, S. George, T. He, L. Luo, S. Son, R. Stoleru, J. Stankovic, A. Wood. EnviroTrack: Towards an Environmental Computing Para-digm for Distributed Sensor Networks. In Proc. of the 24th Int'l Conf. on Distributed Computing Systems (ICDCS 04) March 23-26, Tokyo, Japan, 2004.

[4] K. Piotrowski, P. Langendoerfer, and S. Peter, "tinyDSM: A highly reliable cooperative data storage for Wireless Sensor Networks," in Collaborative Technologies and Systems, International Symposium on, 2009, pp. 225-232, doi: 10.1109/CTS.2009.506