Krzysztof Turchan, Krzysztof Piotrowski
IHP- Innovations for High Performance Microelectronics,
Frankfurt (Oder), Germany

DISTRIBUTED DATA-CENTRIC APPLICATION LOGIC LAYER

EDGElevel architecture has been proposed in response to the attempt to move cloud computing to distributed computing on Internet of Things (IoT) devices. A key element of this architecture is the modular tinyDSM/SmartDSM middleware, which enables efficient data exchange across heterogeneous sensor networks. However, the Distributed Data-Centric Application Logic Layer plays a key role, enabling data aggregation, pattern recognition, parallel processing and efficient real-time processing.

WARSTWA ROZPROSZONEJ LOGIKI APLIKACJI SKONCENTROWANEJ NA DANYCH

W odpowiedzi na próbę przeniesienia przetwarzania w chmurze do rozproszonego przetwarzania na urządzeniach Internetu Rzeczy (IoT), zaproponowano architekturę EDGElevel. Kluczowym elementem tej architektury jest modułowy pośrednik tinyDSM/SmartDSM, umożliwiający efektywną wymianę danych w heterogenicznych sieciach czujników. Jednak to warstwa rozproszonej logiki aplikacji skoncentrowanej na danych (ang. Distributed Data-Centric Application Logic Layer), pełni kluczową rolę, umożliwiając agregację danych, rozpoznawanie wzorców, przetwarzanie równoległe oraz efektywne przetwarzanie w czasie rzeczywistym.

1. INTRODUCTION

The European Union is facing a major competitive challenge in the cloud computing sector. Rapidly establishing numerous data centres with cloud computing capabilities to catch up with the competition seems to be a difficult task. As a result, moving the central point of computing from the cloud to Internet of Things (IoT) edge devices is being considered. Creating a system that enables computing on edge devices becomes crucial. Thus, an EDGElevel architecture concept has been proposed to move cloud computing to edge devices, using a generic solution available for most IoT devices.

The architecture involves connecting to a collaborative ecosystem based on distributed shared memory (DSM) solutions from the IoT device level [1]. However, these differ in terms of the computing power that they are equipped with. Devices in the microcomputer class, such as Raspberry Pi, BeagleBone, or NVIDIA Jetson, have relatively high computing power and are capable of processing more complex operations. On the other side of the scale there are low-power microcontrollers, which often have fractions of what microcomputers have at their disposal. EDGElevel offers solutions regardless of which group the chosen IoT device belongs to.

These solutions are SmartDSM [2] and tinyDSM [3]. Their method of operation is similar while, as this paper focuses on low-power embedded devices, the tinyDSM solution will be described and further considerations will be made from its perspective.

TinyDSM is a pivotal component of the architecture proposed in the EDGElevel architecture, enabling efficient data processing in heterogeneous sensor networks at the edge level. It is a modular intermediary software that focuses on data exchange through tuple space implementation. This allows data to be shared as variables, enabling communication based on grid topology and intensive data replication, thereby increasing system reliability.

The data exchange in tinyDSM takes place via the Data Interface. This can either be connected to Services or Adapters. Services are software modules connected to the Data Interface to perform a specific data processing task. A Service is a generic term, and its application depends mainly on the project assumptions in which tinyDSM is used.

Another software module class is an Adapter. This class includes software modules that aim to translate data into the format that is understandable by the Data Interface, and consequently by the tinyDSM [3]. Any other data source or sink can reside on the other side of an adapter. Data can flow both ways, and this depends mainly on the purpose of the adapter. These can include an adapter for receiving data from sensors, or an adapter for exchanging data between tinyDSM and the communication protocol. The communication protocol is thus a special example of adapter (see Fig. 1).

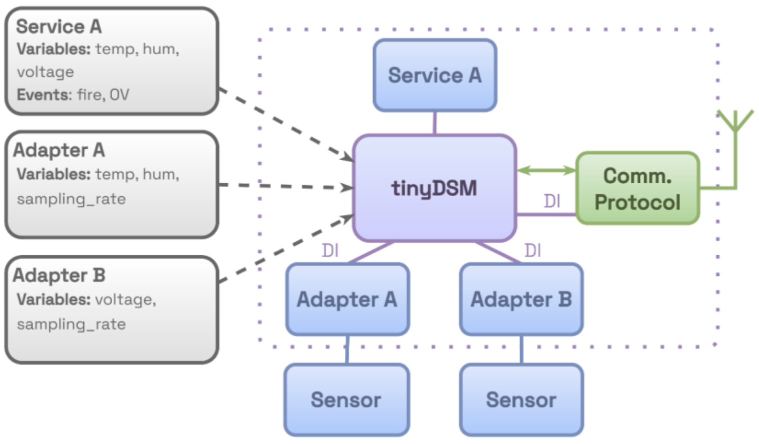

Fig. 3 Structure of tinyDSM

2. TECHNICAL ANALYSIS OF DISTRIBUTED DATA-CENTRIC APPLICATION LOGIC LAYER

This work describes the features of the Distributed Data-Centric Application Logic Layer. This is a distributed software responsible for executing algorithms that affect data across the network for overall application benefits. This layer focuses on the distributed application data processing. Aside this functionality network level system management can be realized, such as optimizing energy consumption, automatically managing this network, different anomaly detection, etc. Thus, the distributed data processing layer includes the tinyDSM and SmartDSM services, which are responsible for processing the measurement data. It is a kind of container for the distributed algorithm, which takes the data flowing over the Data Interface and processes it according to the specifications (see Fig. 2).

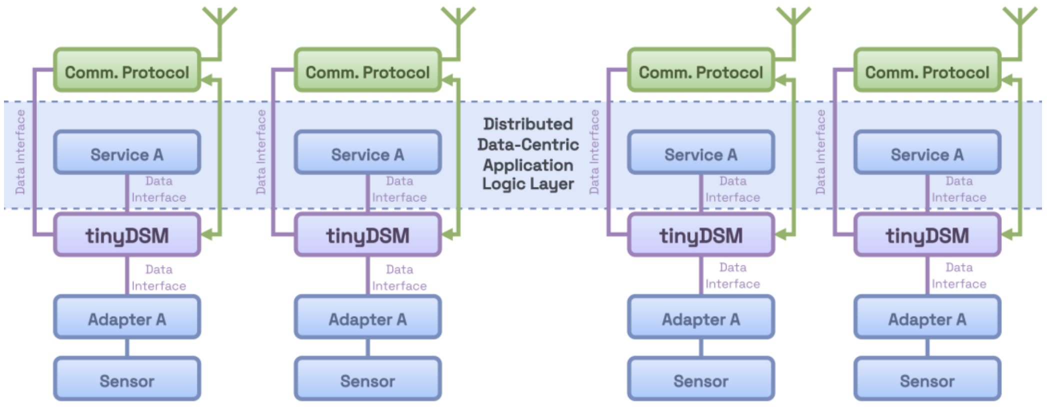

Fig. 4. Distributed Data-Centric Application Logic Layer Representation

The separation of the layer responsible for distributed data processing in such a system as sensor networks allows the introduction of various improvements. These include:

- Data aggregation - A distributed logic layer can coordinate the process of aggregating data from multiple sensors, where data collected by different sensors is combined and processed to provide more comprehensive information. For example, when monitoring environmental conditions, data from multiple sensors can be aggregated to calculate the average temperature or humidity for the entire monitored area.
- Pattern recognition - Nodes in a sensor network can work together to recognize patterns in the data, such as anomalies or sequences of events. A distributed logic layer can coordinate the process of analysing the data and detecting these patterns in real time.
- Parallel processing - nodes can have each other perform complex calculations to optimize time when neighbouring nodes are idle (e.g., when waiting for a measurement).
- Efficient real-time processing - Using a distributed architecture, a network of sensors can process data in real time, enabling rapid response to changing conditions. For example, in traffic monitoring systems, distributed computing can be used to analyse data from multiple sensors to detect traffic jams or accidents as soon as they occur.

The level of complexity of the algorithms that can be run in this layer depends mainly on the computational capabilities of the hardware on which this system runs. Starting from the level of the simplest data control algorithms, through various types of statistical algorithms, even ending with artificial intelligence algorithms. The article *Integration of artificial intelligence with sensor networks* [4] presents a set of good practices for implementing different types of AI algorithms on embedded devices. It presents in the form of tables a comparison of these algorithms from different angles, such as learning speed, complexity level or power consumption. It is a good introduction to the implementation of services for the presented logic layer.

2.1. Use cases of using Distributed data-centric application logic layer

There are many types of service that can operate in the described layer. This subsection is intended to better illustrate the usefulness of the proposed solution.

2.1.1. Statistic scenario for increasing the capacity of the wireless sensor network (WSN)

One of the tasks that can be done in the distributed data-centric application logic layer is to reduce the number of packets transmitted on the network - known as Time on Air (ToA). Let's look at how simple averaging can affect network performance. The task of the service on the edge nodes is to subscribe to measurement data, e.g. temperature, and average it using a digital filter to avoid a noise. The network in which the data flows is of a tree structure with multi-hop capability. The task of the repeater is no longer as simple as to transmit data uncritically to the gateway, but to create various statistical calculations. In this case, it could be the average of locally collected air temperature data. On the other hand, if the network is equipped with nodes with relatively high computing performance, it can generate information from this data, which ultimately contributes even more to reducing the number of packets sent. A service located at the highest point of the WSN - the Gateway - can analyse the incoming data from the network in terms of a predetermined scope of data, and can place them via the SmartDSM middleware in databases, saving a very large amount of server’s memory.

2.1.2. Distributed data-centric application logic layer as a neural network

Examples can, of course, be more abstract and it all depends on the inventiveness of the service developer. This subsection focuses on analysing the structure and operation of described layer using the example of a wireless sensor network as a traffic monitoring system. In such an environment, sensors collect a variety of data, such as vehicle speed, traffic density or noise levels. In order to effectively manage this data and extract useful information from it, an appropriate data application logic layer is required.

For this purpose, the described system can be used by representing each SmartDSM/tinyDSM instance as a single neuron of a sensor network [5]. The following shows how the layers of this neural network would be implemented.

Input layer (tree leaves) - At the lowest level of this structure is the input layer, representing the sensors that collect raw data from the environment. For example, a speed sensor measures the speed of vehicles and a traffic density sensor determines the number of vehicles on a given stretch of road. This data is the basis for the subsequent processing steps.

First hidden layer – Next, there is the first hidden layer, in which the nodes represent individual streets or road sections. Here the data from the sensors along the road section is aggregated and processed. Examples of operations include calculating the average speed or traffic density on a given road section.

Second hidden layer - The next level is the second hidden layer, where nodes can represent larger units such as neighbourhoods. In this layer, the data from the first hidden layer is further aggregated, for example calculating the average speed or traffic density at the level of the entire neighbourhood.

Output layer (root of the tree) - Finally, at the top level is the output layer, represented by the gateway of the network (root of the tree). It is here that the processed data from the second hidden layer is finally analysed and subjected to final calculations. Example tasks could include traffic jam detection, predicting future traffic conditions based on current data, or even estimate air quality in different city areas.

This is, of course, a simplified model, since the weights of neural connections are not considered. It is only intended to show the direction of implementation of the various solutions.

3. CONCLUSION

Implementing an EDGElevel architecture opens up new possibilities for IoT systems, enabling efficient data processing at the network edge. Using a distributed data application layer, significant optimisation of energy management and rapid response to changing conditions can be achieved. Further research into algorithms and the development of hardware infrastructure can additionally enhance the performance and flexibility of such systems, contributing to technological developments in the Internet of Things area. Enhancing this area with edge computing increases its independence from cloud solutions, which is an added advantage. The EDGElevel architecture is a tool that makes it easier for developers to create IoT systems with an even higher level of quality.

REFERENCES

1. Chiba, T., Yoo, M., Yokoyama, T.: A Distributed Real-Time Operating System with Distributed Shared Memory for Embedded Control Systems. 2013.
2. Koropiecki I., Piotrowski K.: SmartDSM: Towards User-Centric IoT Middleware Platform for Privacy-Focused Smart Systems. 2023 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS) 2023.
3. Piotrowski K., Langendoerfer P., and Peter S.: tinyDSM: A highly reliable cooperative data storage for Wireless Sensor Networks. Collaborative Technologies and Systems, International Symposium, 2009.
4. El khediri, S., Benfradj, A., Thaljaoui, A., Moulahi, T., Khan, R. U., Alabdulatif, A., Lorenz, P.: Integration of artificial intelligence (AI) with sensor networks: Trends, challenges, and future directions. Journal of King Saud University - Computer and Information Sciences, 2024.
5. Sukhadeve A.: Understanding Neural Network: A beginner’s guide. Available at: https://www.datasciencecentral.com/understanding-neural-network-a-beginner-s-guide/ Accessed on: 22.04.2024. 2019.