EPiC Series in Computing
Volume 82, 2022, Pages 32–41
Proceedings of 37th International Conference on Computers and Their Applications

Dynamic Meta-Information Management for IoT-based Applications

Madhu Kumari[1], Sugyan Kumar Mishra[2], Narayan C. Debnath[3] and Anirban Sarkar[4]
[1, 2, 4]Department of Computer Science and Engineering,
National Institute of Technology, Durgapur, India.
[3]Department of Software Engineering,
Eastern International University, Binh Duong Province, Vietnam
[1]madhubce2255@gmail.com, [2]sugyan3@gmail.com,
[3]narayan.debnath@eiu.edu.vn [4]sarkar.anirban@gmail.com

Abstract

In recent days, the uses of Internet of Things (IoT) applications have been growing enormously. IoT considers the integration of business process models or process execution with resources of intelligent devices. The concept of process meta store (PMS) [1] optimizes the interactions between IoT devices and business processes. However, the mechanism of PMS highly depends on the meta-information of the system, associated devices, and interactions among them. In this context, a novel semantics of meta-information of process store (MIPS) for IoT-based applications is proposed in this paper. The semantics of various MIPS elements and their relationships are described using a class diagram. Further, a B+ tree-based indexing approach for the MIPS semantic is presented for efficient searching of meta-information. The semantics of MIPS are illustrated using the case study related to the clinical decision support system (CDSS). Moreover, a detailed comparative analysis has been carried out to show the expressiveness of MIPS.

Keywords: IoT, Process Store (PS), MIPS, B+ tree, Searching Algorithm, CDSS, JSON Schema

1 Introduction

IoT has been used in different fields such as healthcare, academics, industry, etc., for providing various customer services in real-life [2]. The existence of heterogeneous IoT devices, standards, and communication protocols raises several challenges for IoT-based applications in terms of interoperability, scalability, and flexibility [3, 4]. In general, business processes communicate with the IoT devices through request-response or publish-subscribe methods for collecting information. As a

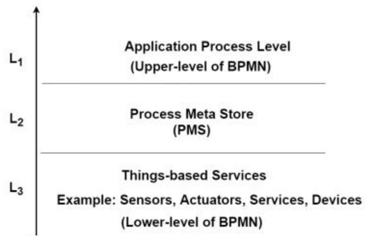


Figure 1: A connection between upper-level BPMN and IoT devices [5]

result, the messages are exchanged between the IoT devices and business processes. This interaction leads to maximum battery consumption of IoT devices [1]. Therefore, a process meta store (PMS) [1] is represented between the IoT device and the business process to optimize the number of interactions between the Things-based Services and Application Process level, as shown in Figure 1. PMS consists of device id, connected service, atomic service, region, location, and flag.

A suitable meta-information is required to effectively coordinate the devices' interactions and business processes. The meta-information is used to preserve and secure the future accessibility of the devices, ensuring that new devices can communicate with the existing legacy devices with specific business goals. As a result, meta-information is critical for IoT-based applications and capable of handling issues like integrability, interoperability, and heterogeneity. Interoperability implies that devices can communicate with one another in an IoT-based system. Meta-information can be helpful in the integration of the new devices with the existing dynamically in order to exchange the information with the IoT-based applications. So, meta-information management is important to handle the diverse number of processes associated with multiple devices. Various meta-information management mechanisms have not addressed the crucial system features like flexibility and scalability [6]. Therefore, devise of a dynamic meta-information management scheme is a key step for the development of an IoT-based system. Various research problems exist for efficient management of IoT, including (i) there is a lack of detailed information about PMS in the existing literature [1]; (ii) how to devise an efficient method for searching meta-information in IoT [6]. There should be some efficient mechanism to handle all the existing and upcoming challenges.

In this context, this paper proposes MIPS semantics for IoT-based systems. This approach provides detailed records of each entity of PS. A B+ tree-based indexing approach for the MIPS semantic is also described for efficient searching of meta-information. Further, a searching algorithm is discussed to find the information in the B+ tree. The CDSS is considered a real-life case study to demonstrate the MIPS. Various dynamic features are considered to compare the semantics of MIPS with the existing approaches.

The remaining sections are organized as follows. Section 2 states existing approaches about meta-information management for IoT-based systems. The meta-information of the PS is described along with its detailed information in Section 3. A case study is mapped with the MIPS. Section 4 shows a detailed comparative study of the proposed approach with the existing methodologies. The logical representation of the proposed semantics of MIPS has been described using JavaScript Object Notations (JSON) in Section 5. Finally, Section 6 concludes with the scope of future research.

2 Related Work

This section provides an overview of the existing works on IoT meta-information management. Mandal *et al.* [6] have presented a flexible and scalable meta-information management system for software as a service (SaaS). This approach is used to find the related business processes and services.

This approach has also been validated in the Hadoop environment. The experimental results are shown by various operations such as searching, modifying, and updating efficiently. A framework [7] is discussed to manage numerous industrial data in an IoT-based framework. This layout gives service-oriented architecture (SOA) for the service consumers. This framework consists of different layers such as physical, network, middleware, database, and application. The physical layer is responsible for generating raw data and distinct events. The communication layer provides a secure connection between each layer of the system. The middleware is responsible for data discovery and applying data processing processes. The database layer partially stores industrial data in the local repositories. This solves the problem of transmission delays and a huge workload on the cloud server. The application layer receives the query from the service consumer and returns the result based on the needs of the service consumer.

An IoT data management framework [8] is discussed to solve the issues such as storing, collecting, and processing data. The proposed distributed data service (DDS) [9] handles many data sources of IoT environments in parallel. The set of functionalities is introduced for DDS to support various operations such as collecting, filtering, and storing data. A metadata model [10] and methodology are discussed to make OLAP cubes from data lakes according to the consumers' needs. A meta-data database (DB) management approach is presented in [11] for live data. The performance of the proposed method is evaluated in the simulated environment. On the other hand, the proposed method consumes more search costs. A cross-domain meta-data framework [12] is discussed to present the related service-based information. This framework makes a generalized standard for all types of data. Thus, it solves the problem of heterogeneity. This architecture enables scalable stream processing systems (SSPS) to provide metadata. This framework reduces the computational overhead and constraints for scalability. Two finite state-based formalisms such as monitoring microservice automata (MMA) and container microservice automata (CMA) [13] are used to design microservice infrastructures. These two microservices are efficiently used for different microservice operations in a dynamic environment. A finite state-based automata is used to find different microservice characteristics. A collection of different architectural views at different granularity levels [14] is discussed for the real-time systems. This paper provides a survey on different orthogonal architectural decay paradigms for predicting, forecasting, and detecting architectural decay. An efficient meta-information management method for IoT is presented in the next section.

3 Meta-Information of IoT

Meta-information explains the data that is mapped with the IoT-based system. This meta-data consists of descriptions of system entities, resources, and IoT services. PMS layer of Figure 1 can be considered as two logical sublayers, namely, Process store (PS) and Meta-Information of PS (MIPS), for efficient usage of the IoT-based system processes, devices, and interactions thereof [Figure 2]. PS is the abstract of PMS. The PS sublayer provides the connectivity semantics among the IoT devices and the business processes. The MIPS sub-layer keeps records of all the information related to PS. This provides the background information those are necessary for establishing the connectivity among the business processes and IoT devices. The semantics of MIPS are described using a class diagram, and the indexing of MIPS information is presented in the form of a B+ tree to minimize the memory size

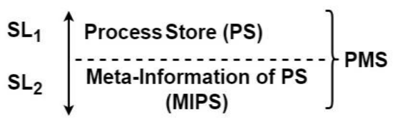


Figure 2: Detail Analysis of PMS

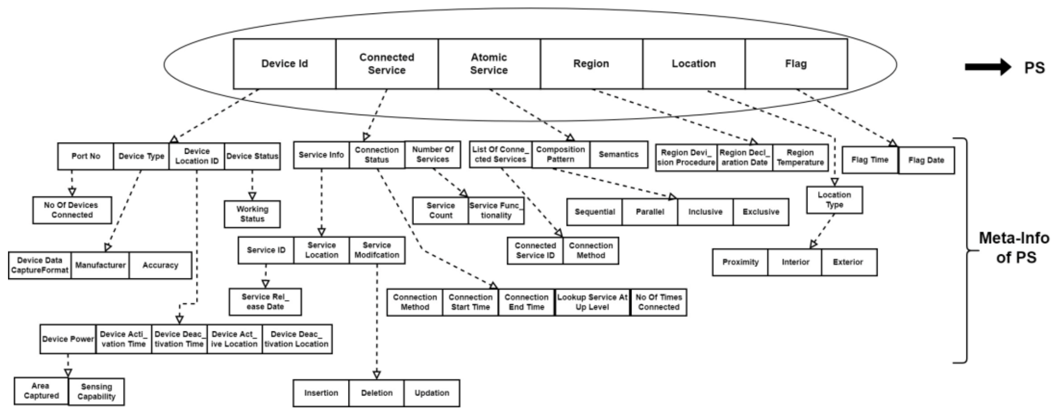
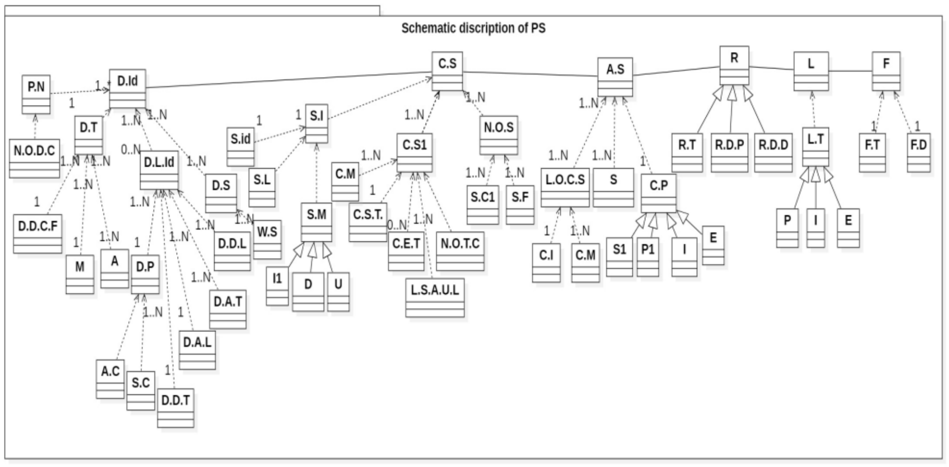


Figure 3: Schematic Description of PS

and efficient searching. When there are changes in the device information at the lower layer, PS provides the necessary updates to MIPS. IoT data is annotated with meta-information to provide context, such as additional functioning attributes, positioning, structural relationships with other items, and so on. Their main job is to give contextual semantics so that rich data (essentially data that has been made usable) may be created for a range of pre-processing and post-processing services and applications, including analytics and device management.



List of annotations:

D.Id: Device Id
C.S: Connected Service
A.S: Atomic Service
R: Region
L: Location
F: Flag
P.N: Port No
D.T: Device Type
D.L.Id: Device Location Id
D.S: Device Status
N.O.D.C: No of devices connected
D.D.C.F: Device data capture format
M: Manufacturer
A: Accuracy

D.D.T: Device deactivation time
D.A.L: Device activation location
D.D.L: Device deactivation location
A.C: Area Captured
S.C: Sensing Capability
W.S: Working status
CS1: Connection status
S.I: Service info
N.O.S: No of services
C.M: Connection method
C.S.T: Connection start time
C.E.T: Connection end time
L.S.A.U.L: Lookup service at up level
N.O.T.C: No of times connected

C.P: Composition Pattern
S: Semantics
C.M: Connection method
S1: Sequential
P1: Parallel
I: Inclusive
E: Exclusive
P: Proximity
I: Interior
E: Exterior
F.T: Flag time
F.D: Flag date
S.C1: Service count
S.F: Service functionality
L.O.C.S: List of connected services
C.I: Connected service id

R.T: Region temperature
R.D.P: Region division procedure
R.D.D: Region declaration date
L.T: Location type
S.id: Service Id
S.L: Service Location
S.M: Service Modification
S.R.D: Service Release Date
I1: Insertion
D: Deletion
U: Updation
D.P: Device power
D.A.T: Device activation time

Figure 4: Class Diagram of Schematic Description of PS

PS consists of device id, connected service, atomic service, region, location, and flag. These six components are the primary components of PS. Each IoT device has a device id, which serves as the device's unique identifier. The device id also includes port numbers, device types, device location identifiers, and device status. Multiple IoT devices can be connected to a port number in a sequential way. As a result, the port number displays the total number of devices connected. The device type specifies the data capture format, manufacturer, and accuracy of the instrument. The accuracy of event identification is crucial for an IoT detection system. The measurements generated by IoT devices are noisy and often marked with a degree of ambiguity, which might reduce the accuracy of detection. The accuracy and correctness of IoT devices are two crucial variables to consider in order to improve their correctness. Device location id indicates the location of a specific device and consists of device power, device activation time, device deactivation time, and device deactivation location. Area captured indicates the area captured by IoT devices, and the ability of sensing parameter is stored in sensing capability. Device status is further categorized into the working status. Working status shows the device's status, whether it is active or not. Connection method, connection start time, connection end time, lookup service at up level, no of times connected are detailed information of connection status. The composition pattern is summarized information of sequential, parallel, inclusive, and exclusive. Flag maintains the information of flag time and flag date. Figure 3 shows the schematic description of PS, and the associated class diagram is exhibited in Figure 4. The main focus area for using this class diagram is the addition of one more feature i.e., cardinality. This shows the relation between the two elements. There are four types of cardinalities that are used here i.e., one-to-one, one-to-many, many-to-one, and many-to-many. The flexibility and dynamicity behaviors of this proposed model can easily be illustrated by using the class diagram.

3.1 Indexing for MIPS

Let $DI= \{di_1, di_2, di_3, di_4, di_5, \dots\dots, di_n\}\ n \in N$, be the set of device ids, $SDI=\{sdi_1, sdi_2, sdi_3, sdi_4, sdi_5, \dots\dots\dots, sdi_m\}\ m \in N$, be the set of device ids to be searched act as the input, and $MDI=\{mdi_1,mdi_2,mdi_3,mdi_4, \dots\dots, mdi_q\}\ q \in N$, be the meta-information set of DI act as the output of Algorithm 1. Initially, the searching algorithm verifies whether the set DI is empty or not. If the set DI is empty, no device is found, and the algorithm terminates its execution, as described in lines 2-3. Otherwise, it matches each element of DI with each element of SDI, as stated in lines 4-7. If matches are found, then the algorithm prints the DI and MDI, as discussed in lines 8-9. On the other hand, if matches are not found, then the algorithm continues the match operation up to i==n, as illustrated in

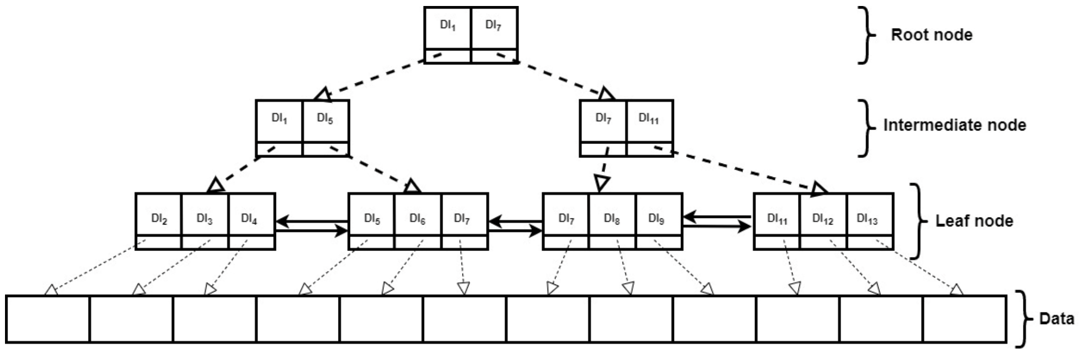


Figure 5. Representation of MIPS through B+ Tree

line 11. This process continues until the SDI is empty or not (j==m), as illustrated in line 14. The device

Algorithm 1: Searching Algorithm

```
     Input: Set of DI and SDI
     Output: Set of MDI
1    function searching (DI, MDI)
2        if (DI== $\Phi$)
3            Then  Print "No device found" and Algorithm Terminates.
4        Else
5                for j=1 to m       // j is the index for SDI.
6                    for i=1 to n   // i is the index for DI.
7                        if (SDI[j] == DI[i])
8                            Then Print the matched Device id.
9                                 Print the MDI [DI[i]].
10                       Else
11                               Continue the match operation up to i==n
12                       End
13                   End
14                   Continue the match operation until j == m
15               End
16       End
17   End
```

indices are arranged according to the B+ tree structure, as displayed in Figure 5. The B+ tree is a data storage system that can store a large amount of data. The data stored in a B+ tree can be accessed both sequentially and directly. This is a self-balancing tree. B+ tree facilitates the faster search query as the data is stored only on the leaf node. In Figure 5, DI represents the device id. This functions as a key that may be used to fetch the data connected with it. In the B+ tree, there may be chances that a redundancy key can be present. This algorithm uses two *for* loops for searching meta-information. So, the time complexity of this algorithm is *O(m*n)*.

3.2 A Case Study

This subsection explains the MIPS through CDSS as a case study. CDSS consists of two sub-parts, patient health record (PHR) and treatment procedure. The PHR includes patient detail and pathology test. Patient details are name, contact number, email id, address. Similarly, various pathology tests are oxygen saturation, sugar level, body temperature, and blood pressure. The PHR consists of two sub-layers such as process health store (PHS), meta-information of PHS (MIPHS), as shown in Figure 6. PHS consists of sensor id, sensor value, final pathology report, region, location, ticket number. The patient wears different sensors for the pathology test. These devices continuously record measurements such as oxygen saturation, sugar level, body temperature, blood pressure, etc. Simultaneously they monitor whether these measurements are in the normal range. They notify to the doctor as soon as it observes any abnormal measurements. These data are maintained in the registry of the hospital. This registry is called patient health store (PHS). MIPHS maintains this PHS. MIPHS is used for the purpose of log file. If the PHS will be lost or crash, then MIPHS helps to get the information of PHS.

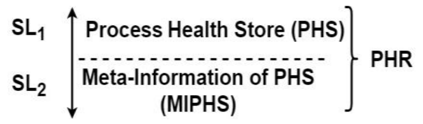


Figure 6: Mapping the MIPS into CDSS in health care

The primary components of PHS are displayed in Figure 7. Sensor id consists of two types of id: static and dynamic. The static id of the device is the unique identifier. The dynamic id can change depending on the situation. Working status, sensor activation time, date, and location, sensor deactivation time, date, and location are included in dynamic id. This dynamic id keeps track of a sensor's history. Sensor value is the abstraction of sensor data. The first step in sensor data abstraction is to collect sensor data; the second step is to simplify the collected data into meaningful information, and the third step is to use this information for application services. There are two key data fields in this data abstraction: static and dynamic metadata. The database describing the profile, device specification, and device characteristics is maintained using static metadata. Dynamic metadata is used to keep track of access information, entry names, types of sensed values, and actual sensor data values in the log file. The date, time, name, result, standard data, and suggestion fields are all included in the final pathology report of a patient. A ticket-generated date, time, and ticket validity period are included in the ticket number's detail section. A detailed comparison of MIPS features is elaborated in the tabular form and will be discussed next.

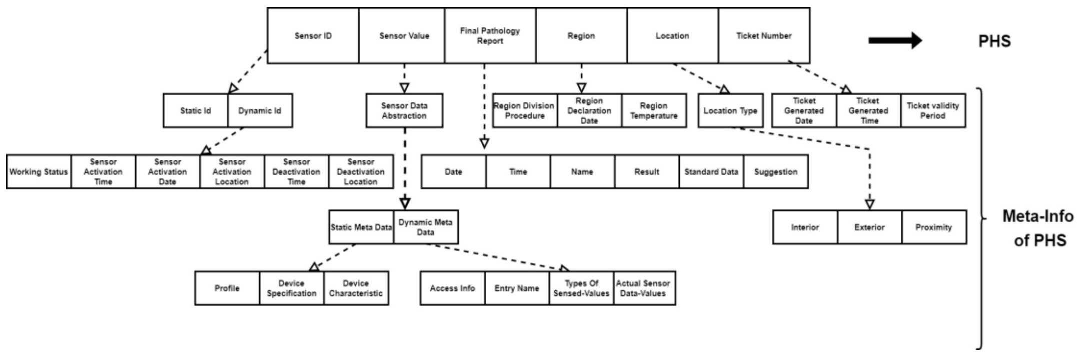


Figure 7: Detailed Analysis of PHS

4 Logical Representation of MIPS index

The MIPS is logically represented in JSON format. Each PS element is stored in its own logical JSON files. These files are linked to each other and will be used for efficient meta-information searching from the logical representation. Figure 8 depicts the logical representation of device id using JSON due to space constraints. JSON files are used to manage the device and its associated components. Similarly, linked JSON can be used to represent other remaining components of PS. This representation allows for the searching, testing, and validation of sensor-captured data via a web-based platform.

```
{
"$id": http://one-part.com/schemas/device id,
"$schema":"http://jsonschema.org/draft04/schema#",
  "type": "object",
  "properties": {"Device Id": {"type": "object",
    "properties": {
       "Port No": {"$ref": "/schemas/port_no"},
    "Device Type": {"$ref": "/schemas/device_type"},
"Device Location Id": {"type": "object",
     "properties": {"Device Power": {"type": "object",
                  "properties": {
                   "Area Captured": {
                      "type": "string"},
              "Sensing Capability": {
                      "type": "string"},},
"required": ["Area Captured", "Sensing Capability]},
          "Device Activation Time": {
                     "type": "string"},
        "Device Deactivation Time": {
                    "type": "string"},
      "Device Activation-Location": {
                   "type": "string"},
   "Device Deactivation Location": {
                        "type": "string"}
        },
"required": ["Device Power", "Device Activation Time",
"Device Deactivation Time", "Device
Activation Location", "Device Deactivation Location"
] },
   "Device Status": {"ref": "/schemas/device_status"},
"required": ["Port No", "Device Type","Device
Location Id", "Device Status" ] } },
"required": ["Device Id"]
}
```

```
"defs": {
"$schema":
"http://jsonschema.org/draft04/
schema#",
   "port_no": {"$id": "/
schemas/port_no",
         type": "object",
            "properties": {
"No Of Devices Connected":
       {"type": "integer"}
          },
 "required":
["No Of Devices Connected"]
     "device_type": {
     "$id": "/schemas/device_type",
       "type": "object",
        "properties": {
"Device Data CaptureFormat":
{"type": "string"},
 "Manufacturer": {"type": "string"},
   "Accuracy": {"type": "integer"}
        },
"required": [
"Device Data CaptureFormat",
     "Manufacturer", "Accuracy"]
"device_status":
{"$id": "/schemas/device_status",
        "type": "object",
          "properties": {
"Working Status": {"type": "string"}
               },
        "required": ["Working Status"]
      }
    },
```

Figure 8: A snapshot of JSON schema for Device Id

5 Comparative Analysis of MIPS

Table. 1 compares and contrasts various existing meta-information approaches based on various parameters. The existing model lacks some of these parameters. However, all of these characteristics are present in the proposed model. A detailed explanation of these features is given below.

(a) *Dynamicity:* Due to the dynamic nature of IoT systems, it is difficult to forecast any new devices and associated events in the system that will occur with certainty. MIPS is open to late binding with any new devices introduced in the IoT-based system. The semantic of MIPS ensures that new devices can communicate with the existing legacy devices with specific business goals and thus capable of handling the dynamicity.

(b) *Interoperability:* Each device and port have their own configurations and specifications. The device is connected with the port to satisfy the service consumer's requirements. Thus, this model handles interoperability in IoT-based applications.

(c) *Flexibility:* Each device has its own specification, flag, configuration, and data format. These devices may be manufactured by different companies. So, the system is capable enough to handle heterogeneous devices to achieve consumer's goals. Here, the multiple devices are supported by a single port. On the other hand, one device can be connected through any available ports

(d) *Scalability:* It indicates the system's capability to handle load according to the service consumer requirements. The MIPS is capable of holding any number of devices and services without affecting the system structure [see in Figure 3].

(e) *Fault-Tolerant:* The log file is the final output of the MIPS are stored in a JSON-based document. These documents can be used to handle fault-tolerant issues programmatically.

Table 1: A Comparative study of different approaches for meta-information management of IoT

Approach	(a)	(b)	(c)	(d)	(e)
Metadata and Z-Score based load Sheeding Technique [15]	✕	✕	✕	✕	✕
Industrial Data Management System (IDMS) [7]	✓	✕	✓	✓	✕
Metadata driven approach [10]	✕	✕	✓	✓	✕
Live Data Model [11]	✕	✕	✕	✕	✕
Cross-Domain Metadata Environment [12]	✕	✕	✕	✕	✕
Meta-information management system for SAAS [6]	✕	✕	✕	✕	✕
MIPS (Proposed Approach)	✓	✓	✓	✓	-

Legends: ✓: Fully Supported; ✕: Not Mentioned/Supported; -: Partially Supported

6. Conclusion

This paper proposed a dynamic meta-information mechanism for the IoT. This meta-information is stored in MIPS and maintained by PS. The PS and MIPS are further extensions of PMS. The representation of this meta-information is shown using a class diagram. This class diagram depicts numerous relationships among MIPS elements. For the faster searching of meta-information, B+ tree is used. This B+ tree uses device id to search the data associated with that device-id. A searching method is described for effectively searching meta-information. This viewpoint is mapped with the real-life scenario through CDSS as a case study. The proposed idea supports various features such as interoperability, flexibility, dynamicity, scalability, and fault tolerance. The dynamic behavior of MIPS is evaluated with the help of JSON schema. In JSON schema, each element of PS is illustrated as an object, and the meta-information of each object is represented as property. This schema is used to validate the type of data, i.e., used in JSON, and it aids in dealing with the issue of dynamicity. The MIPS is compared with several other existing approaches by considering different dynamic features.

In the future, the MIPS semantics will be enriched by adding relevant features and constraints like the primary key of the device id. These will enable the indexing of the meta-information for efficient searching operation. The time complexity of the searching algorithm can be further improved by considering a suitable indexing structure. Detailed experimental analysis of the MIPS performance will be a prime focus of future work.

References

[1] S. Mishra and A. Sarkar, "*Things-Aware Business Process Model*", Accepted in 3rd International Conferences on Advances in Distributed Computing and Machine Learning (ADCML), NIT Warangal, India, 2022.

[2] D. C. Nguyen *et al*., "*6G Internet of Things: A Comprehensive Survey*", in *IEEE Internet of Things Journal*, 2021, pp.1-25, https://doi.org/10.1109/JIOT.2021.3103320.

[3] A. I. A. Ahmed *et al*., "*Service Management for IoT: Requirements, Taxonomy, Recent Advances and Open Research Challenges*", in IEEE Access, 7, 2019, pp. 155472-155488, https://doi.org/10.1109/ACCESS.2019.2948027.

[4] Y. Chen, M. Li, P. Chen, S. Xia, "*Survey of cross-technology communication for IoT heterogeneous devices*", IET Communications, 13(12), 2019, pp. 1709-1720, https://doi.org/10.1049/iet-com.2018.6069.

[5] F. Martins., D. Domingos, D. Vitoriano, "*Automatic decomposition of IoT aware business processes with data and control flow distribution",* in Proceedings of the 21*st* International Conference on Enterprise Information Systems (ICEIS), Heraklion, Crete, Greece, vol. 2, 2019, pp. 516–524, doi:https://doi.org/10.5220/0007766405160524.

[6] A. K. Mandal and A. Sarkar, "*A Novel meta-Information Management System for SaaS"*, International Journal of Cloud Applications and Computing, 9(3), 2019, pp. 1-21 doi:https://doi.org/10.4018/IJCAC.2019070101.

[7] M. Saqlain, M. Piao, Y. Shim, J. Y. Lee, "*Framework of an IoT-based Industrial Data Management for Smart manufacturing"*, Journal of Sensor and Actuator Networks, 8(2), 25, 2019, doi:https://doi.org/10.3390/jsan8020025.

[8] M. Abu-Elkheir, M. Hayajneh, and N. A. Ali, "*Data Management for the Interhnet of Things: Design Primitives and solution"*, Sensors, 13(11), 2013, pp. 15582-15612, doi:https://doi.org/10.3390/s131115582.

[9] R. C. Huacarpuma, R. T. De S. Junior, M. T. De Holanda, R. De O. Albuquerque, L. J. G. Villalba, T-H Kim, "*Distributed Data Service for Data Management in Internet of Things Middleware"*, Sensors, 17(5), 25, 2017, doi:https://doi.org/10.3390/s17050977.

[10] O. Latreche and D. Boukraa, "*Self-Service, On-Demand Creation of OLAP Cubes over Big Data: a Metadata-Driven Approach"*, IEEE International Conference on Big Data (Big Data), 2020, pp. 2907-2914, doi:https://doi.org/10.1109/BigData50022.2020.9378026.

[11] T. Ito, H. Noguchi, M. Kataoka, T. Isoda, Y. Yamato and T. Murase, "*Evaluation of a Realistic Example of Information-Centric Network Metadata Management"*, IEEE 10*th* Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2019, pp. 0888-0893, doi:https://doi.org/10.1109/UEMCON47517.2019.8992942.

[12] H. Choi, D. Kang, N. Kim and W. Rhee, "*Cross-domain metadata environment for relative information-based service"*, IEEE Conference on Wireless Sensors (ICWiSE), 2016, pp. 15-20, doi:https://doi.org/10.1109/ICWISE.2016.8187755.

[13] A. Fellah, A. Bandi, "*Microservice-based Architectures: An Evolutionary Software Development Model"*, EPiC Series in Computing, 75, 2021, pp. 41-48.

[14] A. Fellah, A. Bandi, "*On architectural decay prediction in real-time software systems*", EPiC Series in Computing, 64, 2019, pp. 98-108.

[15] M. J. Diván and M. L. Sánchez-Reynoso, "*A Metadata and Z Score-based Load-Shedding Technique in IoT-based*", International Journal of Mathematical, Engineering and Management Sciences, 6(1), 2021, pp. 363-382, doi:https://doi.org/10.33889/IJMEMS.2021.6.1.023.