The state of the art of macroprogramming in IoT: An update

Iwens G Sene Jr [Universidade Federal de Goiás] iwens@ufg.br
Thalia S. Santana [Universidade Federal de Goiás] thaliassantana15@gmail.com
Renato F. Bulcão-Neto [Universidade Federal de Goiás] rbulcao@ufg.br
Barry F. Porter [Lancaster University] b.f.porter@lancaster.ac.uk

Received 30 November 2021 • Accepted 7 July 2022 • Published 18 November 2022

Abstract

Macroprogramming’s primary goal is to increase developers’ productivity by providing high-level specifications of applications’ behaviour at the system level. Macroprogramming may be a viable solution for developing complex IoT applications, such as those manipulating high data volume and heterogeneity. This paper updates a recent work identifying and analysing primary research on macroprogramming in IoT through a systematic literature mapping (SLM). We extended the search strategy scope by conducting an automatic search over five new databases and also performed the snowballing technique. As a result, besides the 38 studies group found in previous SLM, nine new papers were classified as relevant and rigorously analysed, totalising forty-seven studies. In comparison to previous work, results still point out the recurrence of abstractions in the network infrastructure, highlighting the use of frameworks in one-third of the applications and contributing with an overview of macroprogramming by researchers in different knowledge areas.

Keywords: Internet of Things, Wireless Sensor Network, Systematic Mapping, Adaptation, Programming abstraction

1 Introduction

The literature has proposed a variety of macroprogramming approaches to mitigate the challenges of making programming more efficient for IoT and WSN applications (Madden et al., 2005; Newton et al., 2007). In this model, high-level programs are written to represent the overall control logic of an IoT deployment, and parts of this high-level program may then be compiled into low-level code and pushed directly into IoT devices.

Macroprogramming allows developers to specify a single program which defines the high-level collaboration behaviour for WSNs and IoT applications at the system level, treating the entire network as if it were a “single abstract machine” (Sugihara and Gupta, 2008). Low-level network and device details, such as state maintenance or message transmission, are intentionally hidden from the programmer through an automated translation and deployment of the macroprogram into per-node logic. As a result, macroprogramming, which has already been heavily used in works related to WSN (Newton and Welsh, 2004a; Gummadi et al., 2005; Mottola and Picco, 2011) is emerging as a viable technique for the development of complex and distributed IoT applications, as demonstrated by a number recent efforts (Noor et al., 2019; Hammoudeh et al., 2021a).

Despite a large range of proposals for macroprogramming paradigms, there remains a lack of convergence on the right abstractions that users most benefit from across different application domains. In situations such as when a new device is added into the target deployment, for example, an IoT application should therefore be capable of autonomously integrating the new device and assign to it a task that contributes to the overall goal of the application as defined in the macroprogram (Kephart and Chess, 2003; Salehie and Tahvildari, 2009; Alajlan and Elleithy, 2014). Self-adaptation is the ability of a system to reconfigure itself automatically and dynamically in response to changes, by installing, updating and integrating existing software elements with alternative ones at run-time.

Recently, researchers have proposed various macroprogramming approaches to mitigate the challenges of making programming more efficient for IoT and WSN applications (Mizzi et al., 2019; Qiao et al., 2018). In this model, high-level programs represent the overall control logic of an IoT deployment, and excerpts of this high-level program may then be compiled into low-level code and pushed directly into IoT devices. There are three significant benefits of this approach. Firstly, it allows the programmer to work at a higher level of abstraction and encode the business logic of an entire deployment in a top-down way, which is often significantly more straightforward than writing code for individual nodes. Secondly, as selected control logic from this high-level model can be deployed into the IoT, data processing can still be performed on IoT devices saving significantly in long-haul communication costs to the cloud. This strategy can also provide lower-latency decision and actuation control where IoT-resident control logic is positioned closer to the data sources on which decisions are being made. Finally, as high-level macro logic will often not specify the fine details of exactly where specific data should come from or where control logic should execute, this offers attractive degrees of freedom in the often dynamic deployment environments of IoT systems – so that the macroprogram translator can perform real-time tuning of which data sources are being used and work around failures and node mobility in the placement of control logic.

To establish baselines for comparison with ongoing, recent research results or even identify suitable areas for future research, a systematic literature mapping (SLM) may be of great usefulness (Petersen et al., 2015). An SLM identifies,
selects, evaluates, interprets, and summarises relevant studies about a topic, e.g., macroprogramming in IoT/WSN.

This paper extends a previous SLM we conducted on how the macroprogramming paradigm has been investigated in IoT and WSN (Santana et al., 2021). In such previous work, the search strategy included automatic search in three sources, namely ACM DL, IEEE Xplore, and Scopus. In this updated SLM, we searched on five new sources of studies and performed the snowballing technique to find more relevant studies. The snowballing technique (Wohlin et al., 2012) allows identifying relevant studies through the scanning of the list of bibliographic references or citations of a paper. Considering this broader search scope, this SLM classified nine new studies as relevant in a total of 47. The contribution is the mapping of the macroprogramming’s state of the art in IoT, identifying the level of adaptations performed, with trends in abstractions applied to the group of nodes. Besides, we show how it has been used in WSN and its increasing shift to IoT research in recent years.

This paper is organised as follows: Section 2 overviews the SLM, Section 3 discusses the SLM results, and Section 4 brings our concluding remarks and future work.

2 Materials and Methods

The SLM presented in this paper is depicted in Figure 1 and includes three phases: planning, conducting, and publishing. First, a protocol is planned so that one can reproduce it later. It includes research questions, search strategy, search string, sources of studies, and study selection criteria.

![Figure 1. Phases and activities of this SLM’s update.](image)

In the conductive phase, studies gathered from search sources are initially selected through studies’ metadata reading and applying inclusion and exclusion criteria previously planned. After, helpful information is extracted from these selected studies that, in turn, can still be excluded using the same selection criteria. As a novelty in this SLM’s update, snowballing is performed by checking the citation list of the resulting papers of the data extraction step. This process, called forward snowballing, finishes when no new study is included. Following the SLM goal, the studies remaining of this whole process constitute the set of relevant papers from which answers for the research questions of the protocol are analysed and synthesised.

Finally, the entire protocol and the results of each previous stage are documented as scientific papers or technical reports in the publishing phase.

2.1 Research questions and search terms

This SLM’s main goal is to identify primary research investigating macroprogramming with abstractions for WSN and IoT, which must also perform adaptations at the infrastructure level. The following are the research questions (RQ) we elaborated to be answered in this SLM:

- **RQ1:** What are the application domains found in primary studies?
- **RQ2:** When and where are primary studies published?
- **RQ3:** At what levels does adaptation occur, and what are abstraction types in the infrastructure?
- **RQ4:** How are adaptations carried out in primary research on WSN and IoT?
- **RQ5:** What are the adaptability-related issues found?

The next step was to select the proper search terms to identify the most relevant primary studies to answer these research questions. Helped by experts in macroprogramming and IoT, we chose the following set of candidate search terms for the definition of the search string: macroprogramming, macro-programming, declarative approach, imperative approach, programming abstraction, high level, internet of things, cyber physical, cyber-physical, sensor networks, and wireless sensor networks.

2.2 Automatic search

After evaluating the trade-off between coverage and relevance of the search results in a pilot search, we opted for the following combination of keywords as the final search string:

(macroprogramming OR “macro-programming” OR “declarative approach” OR “imperative approach” OR “programming abstraction”) AND (“high level”) AND (“internet of things” OR “sensor networks”)

Specialists in macroprogramming, IoT, and systematic literature research contributed to the search string definition process.

In our previous work, we adapted the final search string to the ACM DL, IEEE Xplore, and Scopus’s search engines (Santana et al., 2021). In this SLM’s update, we also performed searches on studies metadata at the Engineering Village, ScienceDirect, Springer Link, Web of Science, and Wiley websites. Finally, it is worth mentioning that we chose the *ACM Guide to Computing Literature* option because it indexes both the full-text collection of ACM publications and other digital databases on Computing. This search option turns the ACM DL into the most comprehensive bibliographic database on Computing.

3 More information can be found at https://libraries.acm.org/digital-library/acm-guide-to-computing-literature.
Table 1 details the number of studies retrieved in each source of study. There is a differentiation between the original research and this revisited work: forty-four studies were identified in this extended version (including duplicate documents) after adding five new sources and updating the search results in the three original sources.

<table>
<thead>
<tr>
<th>Source</th>
<th>Original</th>
<th>Extension</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM Digital Library</td>
<td>16</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>IEEE Xplore</td>
<td>15</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Scopus</td>
<td>80</td>
<td>85</td>
<td>5</td>
</tr>
<tr>
<td>Engineering Village</td>
<td>-</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>Science Direct</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Springer Link</td>
<td>-</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Web of Science</td>
<td>-</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Wiley</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>111</td>
<td>155</td>
<td>44</td>
</tr>
</tbody>
</table>

2.3 Study selection and data extraction

We applied the same original selection criteria to the 155 papers returned by the automatic search process (Santana et al., 2021). The exclusion criteria (EC) are:

EC1: The paper does not describe primary research.
EC2: The document retrieved is not a paper (e.g., preface or summary of journals or conference proceedings).
EC3: The full study text is not in English.
EC4: The full study text is not accessible.
EC5: The paper was not published before 2004.
EC6: The paper does not address the IoT or WSN domains.
EC7: The paper does not propose, report, or evaluate the usage of adaptation in the context of macroprogramming for programming abstractions.
EC8: The paper is a preliminary or short version of another study.

A paper is removed from this SLM whenever it meets at least one of the exclusion criteria (EC) presented. Otherwise, the study is categorised based on the only inclusion criteria (IC): “the study reports on the adoption of abstraction in programming and adaptation in infrastructure in IoT and WSN application domains.”

As previously presented in Figure 1, study selection occurs on two occasions: after performing the search strategy (with papers’ metadata reading) and during data extraction (with papers’ full-text reading). This strategy significantly reduces the number of non-relevant papers to the SLM.

After the automatic search process, we identified and removed 66 duplicate papers (from the 155 studies group) with the support of the Parsif.al tool (available at https://parsif.al). Next, we read the title, summary, and keywords of each of the 89, upon which we applied EC and IC and eliminated 20 papers (see Table 3). As a result, we selected 69 “probably relevant” studies since this selection only relies on the reading and interpretation of papers’ metadata.

Next, the data extraction activity requires a form whose fields must be mapped to the research questions in the planning phase. These fields are filled in during the full-text reading of each paper. Table 2 presents the mapping between form fields and research questions.

Table 2. Mapping between research questions and data extraction form fields.

<table>
<thead>
<tr>
<th>Research question</th>
<th>Data extraction form field</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1</td>
<td>Knowledge area</td>
</tr>
<tr>
<td>RQ2</td>
<td>Application domain</td>
</tr>
<tr>
<td>RQ3</td>
<td>Case study</td>
</tr>
<tr>
<td>RQ4</td>
<td>Publication vehicle</td>
</tr>
<tr>
<td>RQ5</td>
<td>Publication year</td>
</tr>
<tr>
<td>RQ6</td>
<td>Adaptation level</td>
</tr>
<tr>
<td>RQ7</td>
<td>Abstraction type</td>
</tr>
<tr>
<td>RQ8</td>
<td>Proposal</td>
</tr>
<tr>
<td>RQ9</td>
<td>Experimental validation</td>
</tr>
<tr>
<td>RQ10</td>
<td>Limitations</td>
</tr>
<tr>
<td>RQ11</td>
<td>Future work</td>
</tr>
</tbody>
</table>

The data extraction activity eliminated 29 papers more, totalling 49 excluded papers. As described in Table 3, the EC7 criterion excluded most. It means that only full-text reading allowed us to eliminate papers not focusing on adaptation and macroprogramming in IoT or WSN.

As a result of the data extraction activity, 40 papers are relevant considering the SLM goal. Thus, the automatic search found only two new studies in comparison with our previous work, which identified thirty-eight.

2.4 Snowballing

Besides automatic search, our search strategy includes forward snowballing (FSB) (Wohlin et al., 2012) as an attempt to obtain other relevant studies using the forty studies group as input. In this SLM, the citation list of each paper was retrieved from the Scopus search engine.

In the first round of FSB, we identified 535 papers, from which 43 were duplicates considering the one-hundred-fifty-five initial studies group. The set of EC rejected 485 studies after metadata and full-text reading.

As seven papers remained, a second-round of FSB was performed. Forty-five studies cited these seven papers. However, nine of them were duplicates, and EC rejected the remaining. As no new paper was identified, the snowballing procedure ended up identifying 580 studies, but only seven (from the first-round) relevant to this SLM.

Table 3. The number of studies excluded by exclusion criteria.

<table>
<thead>
<tr>
<th>Activity</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic search</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>Data extraction</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>1</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>Snowballing</td>
<td>54</td>
<td>28</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>7</td>
<td>425</td>
<td>4</td>
<td>521</td>
</tr>
<tr>
<td>Total</td>
<td>58</td>
<td>42</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>12</td>
<td>448</td>
<td>5</td>
<td>570</td>
</tr>
</tbody>
</table>

2Search carried out on July 14, 2020.
3Search update carried out on July 31, 2021.
Therefore, besides the 38 studies found in the original version of this SLM, this updated version retrieved nine new relevant studies: two by the automatic search and seven by the FSB procedure. The full list containing the forty-seven relevant papers is in Table 4. From now on, we identify them as S1 to S47 (S for study).

By analyzing the source of these relevant studies, we concluded that 98% of them came from Scopus. In other words, Scopus indexes most of the publication venues whose papers investigate abstraction, macroprogramming, and adaptation in infrastructures in IoT or WSN. Further information about these is also available elsewhere.

Finally, Figure 2 depicts the entire selection process with the respective number of primary studies chosen and removed in each activity of the conduction phase. Besides, data extracted from each relevant study is also available.

3 Results and Discussion

This section presents the analysis and synthesis of data extracted from the 47 studies to answer the SLM’s research questions.

3.1 About research question 1

To answer RQ1, “What are the application domains found in primary studies?”, we found out that 72% of the papers (34 out of 47) focused exclusively on WSN, being the area with the highest number of publications. The remaining papers’ subjects are IoT (7) or both IoT and WSN (6).

Figure 3 presents the distribution of papers per publication year. In 2015, the first IoT-oriented papers came out, and the number of such papers has increased since then. This IoT research’s growth is confirmed by the literature (Greer et al., 2019). As WSN is one of the IoT enabling technologies, it may explain the decreasing number of macroprogramming research in WSN favoring IoT.

As depicted in Figure 4, almost 80% of the studies (37 of 47) investigated macroprogramming concepts and practices in real-world case studies. In total, those studies cover sixteen application domains, such as intelligent environments and monitoring. However, no case study was reported in papers published in 2004 and 2015. Besides, we represented in the Others category those papers whose application domain was not explicit.

The smart application domain seems to be a trend since 2018, including smart homes, buildings, grids, and transportation. Moreover, all these scenarios may converge to smart cities, representing a more complex picture for adopting macroprogramming abstractions.

To summarise, the answer to RQ1 is roughly the same as this SLM’s previous version (Santana et al., 2021): most of the macroprogramming studies in WSN with an increasing focus shift to IoT since 2015, and a diversity of application domains with an apparent inclination to smart environments in the last years.
### Table 4. The forty-seven papers analysed on this SLM.

<table>
<thead>
<tr>
<th>ID</th>
<th>Paper</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>A component-based approach for service distribution in Sensor Networks</td>
<td>(Taberkordi et al., 2010)</td>
</tr>
<tr>
<td>S2</td>
<td>A constraint programming approach for managing end-to-end requirements in sensor network macroprogramming</td>
<td>(Hassani Bijarboonehs et al., 2014)</td>
</tr>
<tr>
<td>S3</td>
<td>A library for developing real-time and embedded applications in C</td>
<td>(Basanta-Val and García-Valles, 2015)</td>
</tr>
<tr>
<td>S4</td>
<td>A service-oriented approach to facilitate WSAN application development</td>
<td>(Cafète et al., 2011)</td>
</tr>
<tr>
<td>S5</td>
<td>A service-oriented middleware for wireless sensor and actor networks</td>
<td>(Cafète et al., 2009)</td>
</tr>
<tr>
<td>S6</td>
<td>A state-based programming model and system for wireless sensor networks</td>
<td>(Bischoff and Kortuem, 2007)</td>
</tr>
<tr>
<td>S7</td>
<td>Adaptive dynamic checkpointing for safe efficient intermittent computing</td>
<td>(Maeng and Lucia, 2018)</td>
</tr>
<tr>
<td>S8</td>
<td>Adaptive teams of autonomous aerial and ground Robots for situational awareness</td>
<td>(Hsieh et al., 2007)</td>
</tr>
<tr>
<td>S9</td>
<td>Adaptive Wireless Networks as an Example of Declarative Fractionated Systems</td>
<td>(Choi et al., 2014)</td>
</tr>
<tr>
<td>S10</td>
<td>An easy-to-use 3D visualization system for planning context-aware applications in smart buildings</td>
<td>(Su and Huang, 2014)</td>
</tr>
<tr>
<td>S11</td>
<td>An overview of the VigiNet architecture</td>
<td>(He et al., 2005)</td>
</tr>
<tr>
<td>S12</td>
<td>D’Artagnan: An Embedded DSL Framework for Distributed Embedded Systems</td>
<td>(Mizzi et al., 2018)</td>
</tr>
<tr>
<td>S13</td>
<td>Deductive Approach to Processing High-Level Video Activity Queries in UAV Networks</td>
<td>(Gupta, 2018)</td>
</tr>
<tr>
<td>S14</td>
<td>Defining Services and Service Orchestration Acting on Shared Sensors and Actuators</td>
<td>(Bouali Baghli et al., 2018)</td>
</tr>
<tr>
<td>S15</td>
<td>Design and compilation of an object-oriented macroprogramming language for wireless sensor network</td>
<td>(Oppermann et al., 2014)</td>
</tr>
<tr>
<td>S16</td>
<td>Developing wireless sensor network applications based on a function block programming abstraction</td>
<td>(Kerasiotis et al., 2012)</td>
</tr>
<tr>
<td>S17</td>
<td>EcoCast: Interactive, object-oriented macroprogramming for networks of ultra-compact wireless sensor nodes</td>
<td>(Tu et al., 2011)</td>
</tr>
<tr>
<td>S18</td>
<td>Efficient configuration and control of SANETs using FACTS</td>
<td>(Terfloth and Schiller, 2008)</td>
</tr>
<tr>
<td>S19</td>
<td>Efficient routing from multiple sources to multiple sinks in wireless sensor networks</td>
<td>(Ciciriello et al., 2007)</td>
</tr>
<tr>
<td>S20</td>
<td>Energy-efficient task mapping for data-driven sensor network macroprogramming</td>
<td>(Pathak and Prasanna, 2010)</td>
</tr>
<tr>
<td>S21</td>
<td>Logical neighborhoods: A programming abstraction for wireless sensor networks</td>
<td>(Mottola and Picco, 2006)</td>
</tr>
<tr>
<td>S22</td>
<td>Macro programming a spatial computer with bayesian networks</td>
<td>(Mamei, 2011)</td>
</tr>
<tr>
<td>S23</td>
<td>MBMF: A framework for macroprogramming data-centric sensor network applications using the Bird-Meertens formalism</td>
<td>(Loke and Nadarajah, 2009)</td>
</tr>
<tr>
<td>S24</td>
<td>Nano-CF: A coordination framework for macro-programming in Wireless Sensor Networks</td>
<td>(Gupta et al., 2011)</td>
</tr>
<tr>
<td>S25</td>
<td>PICO-MP: Decentralised macro-programming for wireless sensor and actuator networks</td>
<td>(Dulay et al., 2018)</td>
</tr>
<tr>
<td>S26</td>
<td>A platform independent communications middleware for heterogeneous devices in smart grids</td>
<td>(Chen et al., 2019)</td>
</tr>
<tr>
<td>S27</td>
<td>ProFaNTG: Assist for programming and managing performance-aware sensor network application</td>
<td>(Elots et al., 2015)</td>
</tr>
<tr>
<td>S28</td>
<td>Programming iMote networks made easy</td>
<td>(Bauderon et al., 2010)</td>
</tr>
<tr>
<td>S29</td>
<td>Intelligent IoT Systems with a Python-based Declarative Tool</td>
<td>(D’Urso et al., 2019)</td>
</tr>
<tr>
<td>S30</td>
<td>Programming the smart home</td>
<td>(Bischoff et al., 2007)</td>
</tr>
<tr>
<td>S31</td>
<td>PS-QUASAR: A publish/subscribe QoS aware middleware for Wireless Sensor and Actor Networks</td>
<td>(Chen et al., 2013)</td>
</tr>
<tr>
<td>S32</td>
<td>Region streams: Functional macroprogramming for sensor networks</td>
<td>(Newton and Walsh, 2004b)</td>
</tr>
<tr>
<td>S33</td>
<td>The omni macroprogramming environment for sensor networks</td>
<td>(Awan et al., 2006)</td>
</tr>
<tr>
<td>S34</td>
<td>TinyReef: A register-based virtual machine for wireless sensor networks</td>
<td>(Marques et al., 2009)</td>
</tr>
<tr>
<td>S35</td>
<td>Transaction: Where transactions meet the physical world</td>
<td>(Sengupta et al., 2019)</td>
</tr>
<tr>
<td>S36</td>
<td>UBIQUEST, For Rapid Prototyping of Networking Applications</td>
<td>(Ahmad-Kasem et al., 2012)</td>
</tr>
<tr>
<td>S37</td>
<td>USEME: A service-oriented framework for wireless sensor and actor networks</td>
<td>(Cafète et al., 2008)</td>
</tr>
<tr>
<td>S38</td>
<td>μsETL: A set based programming abstraction for wireless sensor networks</td>
<td>(Hossain et al., 2011)</td>
</tr>
<tr>
<td>S39</td>
<td>A Service-Oriented Approach for Sensing in the Internet of Things: Intelligent Transportation Systems and Privacy Use Cases</td>
<td>(Hammoud et al., 2021b)</td>
</tr>
<tr>
<td>S40</td>
<td>ACAOT: A Framework for Adaptable Context-Aware IoT applications</td>
<td>(ElKady et al., 2020a)</td>
</tr>
<tr>
<td>S41</td>
<td>A modular and extensible macroprogramming compiler</td>
<td>(Hnat et al., 2010)</td>
</tr>
<tr>
<td>S42</td>
<td>A Resource-Oriented Programming Framework Supporting Runtime Propagation of RESTful Resources</td>
<td>(Qiu et al., 2014)</td>
</tr>
<tr>
<td>S43</td>
<td>Enabling Scope-Based Interactions in Sensor Network Macroprogramming</td>
<td>(Mottola et al., 2007)</td>
</tr>
<tr>
<td>S44</td>
<td>Hybrid Macroprogramming Wireless Networks of Embedded Systems with Declarative Naming</td>
<td>(Intanagonwiwat, 2012)</td>
</tr>
<tr>
<td>S45</td>
<td>makeSense: Simplifying the Integration of Wireless Sensor Networks into Business</td>
<td>(Mottola et al., 2019)</td>
</tr>
<tr>
<td>S46</td>
<td>Role-based automatic programming framework for interworking a drone and wireless sensor networks</td>
<td>(Min et al., 2018)</td>
</tr>
<tr>
<td>S47</td>
<td>snBench: Programming and virtualization framework for distributed multitasking sensor networks</td>
<td>(Ocean et al., 2006)</td>
</tr>
</tbody>
</table>

### 3.2 About research question 2

To answer RQ2, "When and where are primary studies published?", we observed that 25% of the studies (12 of 47) about macroprogramming in IoT/WSN were published from 2018. Following our protocol, there was no paper about that subject in 2016 and 2017.

Concerning publication venues, conferences and workshops cover 72% (34 of 47) of the accepted papers (see Figure 5). Besides, from 45 distinct publication venues, only two published two papers each: the ACM/IEEE International Conference on Information Processing in Sensor Networks and the IEEE Conference on Local Computer Networks.
In brief, the answer to RQ2 is similar to the one described in Santana et al. (2021): a significant number of studies (25%) about macroprogramming in IoT/WSN during the last four years and a heterogeneous collection of publications venues. These results suggest an increasing interest in research on macroprogramming for IoT/WSN in recent years. Besides, the community has a great list of options to publish their research on that subject.

### 3.3 About research question 3

The RQ3 investigates “At what levels does adaptation occur, and what are abstraction types in the infrastructure?”. Concerning adaptation level, we followed the Krupitzer’s taxonomy that describes five levels of adaptation, as depicted in Figure 6: application (individual or a set of applications), software systems (middleware or operating system), communication (network infrastructure or communication patterns), context, and technical resource (Krupitzer et al., 2015).

The most investigated adaptation levels are, in this sequence, communication in the network infrastructure (25.5%), context (21.3%), and (ensemble of) applications (19.1%), represented by 12, 11, and 9 studies of the 47 accepted papers (see Figure 7). The most investigated adaptation levels are, in this sequence, communication in the network infrastructure (25.5%), context (21.3%), and (ensemble of) applications (19.1%), represented by 12, 11, and 9 studies of the 47 accepted papers.

Considering the adaptation level and the knowledge area of each study in Figure 7, a deeper analysis reveals that communication in the network infrastructure is studied most (11), followed by context (8) and communication pattern (6). Besides, adaptation is more frequent at the middleware (3) and the application (2) levels in IoT-oriented papers. It may be explained because middleware is helpful in situations with often resource-constrained IoT devices. Besides, there is no study examining adaptation at a single application level.

Regarding abstraction type, we used Motolla’s work that classifies it as nodes, groups, and systems (Mottola, 2008). At the node level, macroprogramming abstractions alter indi-

---

**Figure 4.** Application domain per publication year.

**Figure 5.** Publication venue per publication year.

**Figure 6.** A taxonomy for adaptation level (Krupitzer et al., 2015).

**Figure 7.** Adaptation levels and knowledge area.
vidual nodes’ states. At the group level, such modifications occur in a group of nodes. Finally, macroprogramming instructions spread over the network at the system level.

As shown in Figure 8, the group adaptation type is present in almost half of the studies (23 of 47)—we suppose the flexibility of subdividing a sensor network into smaller groups with common characteristics may explain this high percentage. Next, we crossed adaptation levels and abstraction types from the 47 accepted papers. Results reveal that context and communication in the network infrastructure are most present at the system and group levels, respectively. On the other hand, there is a more balanced distribution between adaptation levels at the node abstraction level.

![Figure 8. Adaptation level in relation to abstraction classification.](image)

The answer to RQ3 somewhat differs from the one presented in Santana et al. (2021). Communication in the network infrastructure remains the most investigated adaptation level; the same applies to the studies examining groups of nodes as abstraction type. However, this SLM’s update shows that the number of studies about nodes as abstraction succeeds the number of studies about the system abstraction.

### 3.4 About research question 4

To identify “How are adaptations carried out in primary research on WSN and IoT?”, we found out twelve different ways of implementing adaptation using macroprogramming in IoT/WSN, as depicted in Figure 9.

Software frameworks are present in more than one-third of the studies (17 of 47). Frameworks hide low-level details of designers’ and programmers’ tasks, automate part of these tasks, and ease software development. We believe these assumptions explain the high number of studies implementing adaptation demands in a software framework. Other implementations of the adaptation requirement include programming languages, middleware, and systems (six studies each).

We also classified the studies under the validation point of view: implementation, prototype, simulation, and testbed. Approximately two-thirds of the studies (30 of 47) validated their research proposals through implementation. On the other hand, simulations were performed in ten studies, and the testbed was the less frequent validation type (only 2 of 47).

As shown in Figure 9, implementation was also the most used validation type among the four most employed adaptation proposals (i.e., framework, programming languages, middleware, and system).

Thus, a software framework is the most frequent adaptation implementation, as also described in Santana et al. (2021). However, this SLM’s update describes more studies employing programming language, middleware, and the system as adaptation implementations.

### 3.5 About research question 5

To answer “What are the adaptability-related issues found?” we identified problems, limitations, and future work proposals in each accepted paper. This SLM revealed 25 distinct issues: communication, network topology, network traffic, context-awareness, coordination, among others. Communication was the highest cited issue (4 of 47), even in IoT-oriented studies.

Observing the knowledge area (Figure 10), in IoT, the communication limitations were present in two papers. That represented the majority. In WSN, however, concentrated middleware and studies in development (two papers each).

One of the key results of our study is that very few research papers examine the opportunities for adaptation of a deployment guided by a high-level macroprogram. This is a key opportunity that we seek to exploit in our future research—building on the challenges in RQ5, we aim to develop formal approaches to continually guide a deployed system towards a more optimal form according to its current deployment environment conditions, while using a high-level macroprogram to ensure that the deployed system remains within an envelope of behaviour expected by the system designer.

Of these, 37 have different directions, and the others converge on the following themes, which show the target problems that researchers aim to solve using macroprogramming. This provides insight into challenges researchers view as being particularly suited to a macroprogramming-based solution. Overall, the dominant target problems across the study period are energy efficiency, aiming to extend the lifetime of deployed infrastructures, and scalability, given the large sizes typical of most deployments. We also note that scalability became the dominant target problem in the three latest years of the period comprised by our study. Besides energy efficiency and scalability, other target problems that have received significant interest include device location, collaboration, fault resilience, and time synchronization.

Finally, the answer to QP5 showed a large number of different types of limitations, as well as trends for future work, and it was not possible to identify any particular kind of trend. Similar to previous work (Santana et al., 2021), communication had the most significant number of limitations, with 4 studies, most of them in IoT.
3.6 Results synthesis

Figure 11 illustrates a bubble graph synthesizing the most relevant information we extracted and analysed from the accepted papers in this SLM. Three axes of information compose that bubble chart: adaptation level, abstraction type, and application area. The bubble size represents the number of studies that investigate the intersection of each two axes.

Notice the high concentration of macroprogramming research involving the communication adaptation level in WSN-oriented work. Besides, observe the number of studies in which modifications caused by macroprogramming abstractions disseminate in a group of nodes. Finally, there was no study in which adaptations take place in a single application. This finding confirms that macroprogramming should not be tackled at IoT/WSN isolated components.

4 Conclusions and future work

Overall, we posit that macroprogramming remains a topic of significant interest and a natural approach for IoT systems. Because these systems are often composed of a large number of devices controlled by a single organization, and because these devices are typically heterogeneous and relatively difficult to program in themselves, it is highly desirable to gain high-level abstractions to program the entire system.

We draw on the main results of our study to present a discussion of the challenges and opportunities for future research on macroprogramming for WSNs and IoT:

Converging on the right paradigms: our study revealed various macroprogramming paradigms for different applications and problems. For example, many existing programming abstractions for WSN and IoT provide a specification of actions performed by individual de-
sives or instead allow one to program the network and customize the underlying run-time, which is often dynamic. However, we did not observe any notable convergence on accepted macroprogramming paradigms in general or for specific applications/challenges. Among the notable exceptions observed in the papers, ACAIOT (Adaptive Context-Aware IoT applications) was the proposed framework by comparing its architecture with recent research studies (ElKady et al., 2020b). Also, we use ACAIOT to implement smart home application services by using a real dataset.

Embracing the dynamic nature of the environment: the devices and services of an IoT deployment can change frequently and vary their availability at any given time. This can make it challenging for developers to define applications that seamlessly persist across this volatility to offer a continuous level of service. Macroprogramming appears to provide a straightforward solution to this problem, in that the overall business or scientific logic of an application can be defined separately from specific devices, with the deployment of a macroprogram able to adjust autonomously to the currently available resources.

Variable distribution of logic: as IoT deployments envision each device becoming a uniquely addressable Internet endpoint, and with the prevalence of cheap cloud computing, there is an inclination to use IoT nodes as non-intelligent data endpoints or actuation endpoints, with all business logic placed on cloud services which collect data from all nodes and make decisions based on that data. However, this architecture requires significant network capacity to get all data into the cloud and denies opportunities to perform at least some processing within the network. Macroprogramming offers a potential chance to automate the distribution of logic both within cloud services and within the IoT network itself, with automated macroprogram deployment tool chains able to decide which logic is best suited for which location based on available processing, network, and energy capabilities. One of the challenges relates to the degrees of freedom offered by macroprograms: as the system description is inherently high-level, the operationalisation of macroprograms has significant freedom in how they are deployed over time – including the placement of logic and the adaptation to fluctuations in the environment and resources. Let’s take this opportunity to its extreme. We could envision a macroprogram acting as a specification of what the system is designed to do in an ideal scenario and an envelope of acceptable ways to implement that functionality. A smart deployment manager could then take that idealized specification and intelligently map it onto the available resources continuously, reporting how close the actual deployment is to the idealized specification of the microprogram.

This work contributes to the IoT and WSN fields, with the results of a systematic literature mapping (SLM). This SLM brings important work and reporting aspects as an alternative to propose the programming of devices at a high level, mainly with the growth of networks in the volume of data (high number of sensors) and device heterogeneity.

Finally, SLM aims to categorize the main findings of primary research about a topic and to benefit researchers in establishing baselines for other research activities. SLM is an open form of what the literature calls a systematic literature review (SLR), i.e., a deeper analysis and comparison of a collection of studies. As such, future work may consist of the conduction of an SLR on macroprogramming in IoT, focusing on those papers exploring multiple adaptation levels in groups of network nodes (see Figure 11). It is common practice to perform an SLR on pieces of evidence found in an SLM. Results of an SLR can be used to understand the efficacy and efficiency of a method or technology or the strengths and weaknesses of methods and technologies under certain circumstances. As study quality assessment is a widely deployed technique in SLMs and SLRs, we can also elaborate on a set of quality criteria to evaluate the different contributions of each of the 47 papers selected.

---

**Figure 11.** A bubble chart describing the mapping among adaptation level, abstraction type, and application area.
Acknowledgements

This work was partly funded by the Royal Society – Newton Mobility Grant NMG-R2-170105. This study was financed in part by the CAPES - Brazil. This research is also part of the INCT of the Future Internet for Smart Cities funded by CNPq proc.465446/2014-0, CAPES proc.88887.136422/2017-00, and FAPESP proc.14/50937-1 and 15/24485-9.

References


The state of the art of macroprogramming in IoT: An update

Sene Jr et al. 2022


Mottola, L. (2008). Programming wireless sensor net-
works: from physical to logical neighborhoods. Available at: https://mottola.faculty.polimi.it/theses/mottola08programming.pdf.


