A Review on Visual Programming for Distributed Computation in IoT

Margarida Silva[1], João Pedro Dias[1,2],
André Restivo[1,3], and Hugo Sereno Ferreira[1,2]

[1] DEI, Faculty of Engineering, University of Porto, Porto, Portugal
[2] INESC TEC, Porto, Portugal
[3] LIACC, Porto, Portugal
[4] {ana.margarida.silva,jpmdias,arestivo,hugo.sereno}@fe.up.pt

Abstract. Internet-of-Things (IoT) systems are considered one of the most notable examples of complex, large-scale systems. Some authors have proposed visual programming (VP) approaches to address part of their inherent complexity. However, in most of these approaches, the orchestration of devices and system components is still dependent on a centralized unit, preventing higher degrees of dependability. In this work, we perform a systematic literature review (SLR) of the current approaches that provide visual and decentralized orchestration to define and operate IoT systems, reflecting upon a total of 29 proposals. We provide an in-depth discussion of these works and find out that only four of them attempt to tackle this issue as a whole, although still leaving a set of open research challenges. Finally, we argue that addressing these challenges could make IoT systems more fault-tolerant, with an impact on their dependability, performance, and scalability.

Keywords: Internet-of-Things · Orchestration · Visual Programming · Decentralized Computation · Large-Scale Systems

1 Introduction

The Internet-of-Things (IoT) comprises many devices with a wide range of capabilities, directly or indirectly connected to the Internet. This allows them to transfer, integrate, analyze and act according to data generated among themselves [11]. IoT systems use devices at an unprecedented scale, with applications ranging from mission-critical to entertainment and commodity solutions [14].

The widespread usage of IoT led to a mostly uncontrollable and ever-growing heterogeneity of devices, differing in computational power, protocols, and architectures, comprising a large-scale and distributed (geographically and logically) system of systems. These characteristics raise many development challenges in guaranteeing their scalability, maintainability, security, and dependability [56]. Consequently, the active pursuit of reducing the complexity and technical knowledge needed to configure and adapt such systems to their needs (from manufacturing floor automation to *smart home* system customization) eventually led to

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

the exploration of *low-code* [10, 20] and conversational approaches [36]. Visual programming (VP) approaches (*model-* or *mashup*-based) provide such means via the arrangement of visual elements, which are then automatically translated into executable artifacts, by leveraging some kind of a Visual Programming Language [45, 13, 20]. One of the most popular approaches is Node-RED [39, 34], which provides both a visual editor and a run-time environment for IoT systems.

Most VP solutions (Node-RED included) provide a centralized approach (which can be either *on-premises* or cloud-based) where the main component transforms and processes most of the computation on data provided by edge and fog devices. Consequences of this approach are well-known: (1) single point of failures, (2) violation of data boundaries (private, technological, and political), and (3) unused edge computational power. Recent research effort put in *Fog and Edge Computing* [54, 20] focus on solving these by leveraging the resources available in lower-tier devices to improve overall dependability [46, 24, 23, 22], performance [44], scalability, observability [52], and reproducibility [19, 21].

In this paper we present a systematic literature review (SLR) on VP approaches for IoT, focusing on those related to orchestration of multiple components. Our initial search yielded 2698 results across three different scientific databases. We refined this selection with *inclusion* and *exclusion* criteria, resulting in 21 papers. Through snowballing and taking into account previous (non-systematic) surveys, we found 8 new works, resulting in 28 approaches across 22 papers. We compared their characteristics, including scope, architecture, scalability, and VP paradigms. We then carried out an in-depth analysis on the subset that provided mechanisms for decentralized computation.

The remainder of this paper is structured as follows: Section 2 presents the methodology used in this research, Section 3 presents the results of the literature review, followed by an in-depth analysis of the current alternative for visual IoT decentralized orchestration in Section 4. An overview of the current issues and research challenges, and some final remarks, is given in Section 5.

2 Literature Review Methodology

This work follows a Systematic Literature Review (SLR) methodology to gather information on the state of the art of VP applied to the IoT paradigm, with a particular emphasis on orchestration concerns. The goal of a systematic literature review is to synthesize evidence with emphasis on its quality [43]. We started by defining the research questions to be answered and choosing data sources to search for publications. We outlined the following research questions (RQ):

RQ1. *What are the relevant VP approaches applied to distributed computation and orchestration in IoT?* Using VPs to make IoT development easier for the end-user is a common go-to approach. However, we argue that there is a scarcity of those that provide decentralized approaches;

RQ2. *What architectures and tiers characterize the approaches found in* **RQ1***?* IoT systems can target one or more tiers, as well as be implemented in a

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

Table 1. Inclusion and exclusion criteria.

	ID	Criterion
Exclusion	EC1	Not written in English.
	EC2	Presents just ideas, tutorials, integration experimentation, magazine publications, interviews, or discussion papers.
	EC3	Does not address multiple devices' orchestration.
	EC4	Has less than two (non-self) citations when more than five years old.
	EC5	Duplicated articles.
	EC6	Articles in a format other than camera-ready (PDF).
Inclusion	IC1	Must be on the topic of VP in IoT.
	IC2	Contributions, challenges, and limitations are detailed.
	IC3	Research findings include sufficient explanation on how the approach works.
	IC4	Publication year in the range between 2008 and 2019.

centralized or decentralized architecture. A VP tool applied to IoT can facilitate the development of systems that operate and distribute computing tasks among the available tiers. Each tier and type of architecture offers advantages and disadvantages; understanding these characteristics is essential to understand how they can be used;

RQ3. *What was the evolution of VP approaches applied to IoT over the years focusing on its decentralized operation?* To understand the field of VP applied to IoT, more specifically, its visually-defined decentralized operation, it is essential to perceive its evolution.

```
((vpl OR visual programming OR visual-programming) OR
 (node-red OR node red OR nodered) OR (data-flow OR
 dataflow)) AND (iot OR internet-of-things)
```

Listing 1.1. Search query for relevant literature on *IEEEXplore*, *ACM DL* and *Scopus*.

Answering these questions will provide valuable insights for both practitioners (in terms of summarizing the current practices on the usage of VP methodologies for IoT orchestration are) and researchers (showing current challenges and issues that can be further researched). We follow the criteria detailed in Table 1 and outlined in Fig. 1. Three popular and reputable scientific databases were used, namely *IEEE Xplore*, *ACM Digital Library*, and *Scopus*. Listing 1.1 shows the query we used to convey the most probable keywords to appear in our target candidates, including popular variants. This search was performed in October 2019, and the results can be seen in the first step of Fig. 1. The evaluation process of the publications then followed eight steps:

1. **Automatic Search**: Run the query string in the different scientific databases and gather results;
2. **Filtering** (*EC1*, *IC4*, and *EC6*): Publications are selected regarding its (1) language, being limited to the ones written in the English language,

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

(2) publication date, being limited to the ones published between 2008 and 2019, and (3) publication status, being selected only the ones that are published in their final versions (camera-ready PDF format);
3. **Filtering to remove duplicates** (*EC5*): The selected papers are filtered to remove duplicated entries;
4. **Filtering by *Title* and *Abstract*** (*EC2–EC4*, and *IC1–IC3*): Selected papers are revised by taking into account their *Title* and *Abstract*, by observing the (1) stage of the research, only selecting papers that present approaches with sufficient explanation, some experimental results, and discussion on the paper contributions, challenges, and limitations, (2) contextualization with recent literature, filtering papers that have less than two (non-self) citations when more than five years old, and (3) leverages the use of visual notations for orchestrating and operating multi-device systems;
5. **Filtering by *Introduction* and *Conclusions*** (*EC2–EC4*, and *IC1–IC3*): The same procedure of the previous point is followed but taking into consideration the *Introduction* and *Conclusion* sections of the papers;
6. **Selected Papers Analysis**: Selected papers are grouped, and surveys are separated; their content is analyzed in detail;
7. **Survey Expansion**: For surveys found, the enumerated approaches are analyzed and filtered, taking into account their scope and checking if they are not duplicates of the currently selected papers;
8. **Wrapping**: Works gathered from the *Selected Papers Analysis* (individual papers) and the *Survey Expansion* are presented and discussed.

The total number of publications was 2698, 22 of each were selected (*cf.* Fig. 1). One was a survey [47] (pointing to 8 new works), and the others presented approaches relevant for our RQs.

3 Results

From the 22 publications, 28 different approaches were analyzed and distributed among categories, according to several characteristics:

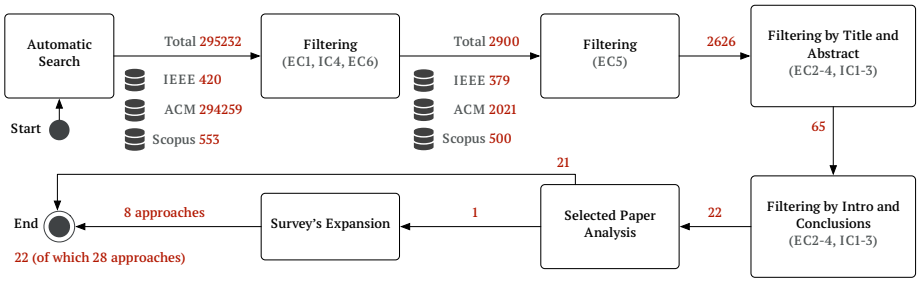

Fig. 1. Pipeline overview of the SLR Protocol.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

1. **Scope.** Some approaches have specific use cases in mind. Therefore, knowledge of a tool's scope helps assess if it solves a problem or fills a specific gap in the literature. Example values consist of *Smart Cities, Home Automation, Education, Industry* or *Several* if there is more than one;
2. **Architecture.** VP approaches applied to IoT can have a centralized or decentralized architecture, based on their use of Cloud, Fog, or Edge Computing architecture. Possible values are *Centralized*, *Decentralized*, and *Mixed*;
3. **License.** The license of software or tool is essential in terms of its usability. Normally, an open-source software reaches a bigger user base, allowing everyone to expand and contribute to it. Possible values are the name of the tool license or N/A if it does not have one;
4. **Tier.** IoT systems are composed of three tiers — *Cloud*, *Fog*, and *Edge*. A tool can interact in several of these, shaping its features and how it is built;
5. **Scalability.** Defines how the tool or framework scales. It can be calculated based on metrics used to test the performance of the system. We considered scalability in terms of number and different types of devices supported. Possible values are *low*, *medium*, *high*, or N/A if information not sufficient;
6. **Programming.** According to Downes and Boshernitsan [9], VPs can be classified in five (possibly overlapping) categories: (1) Purely visual languages, (2) Hybrid text and visual systems, (3) Programming-by-example systems, (4) Constraint-oriented systems and (5) Form-based systems. It is important to know which type so that it might be possible to assess the type of experience the tool provides to the user and its architecture;
7. **Web-based.** Defines if the VP and/or environment can be used in a browser. It is useful in terms of the usability of the tool.

The resulting categorization is depicted in Table 2. Some key takeaways are easily observable, namely: (1) most approaches use a centralized architecture, (2) the hybrid visual-textual programming paradigm is predominant, and (3) most approaches are web-based. The extended findings and their categorization is presented in Table 3, following the same previously defined categories.

3.1 Analysis and Discussion

The approaches presented in this SLR passed the evaluation process defined in Section 2. Approaches supporting only one device or extending an existent VPL by applying it to IoT were left out. From the resulting ones, we analyzed the following aspects:

Domain. The surveyed approaches target different domains: six were specific to home automation, four to education, three to specific domains, and one for the industry; the remainder 14 had a wide range of use cases;

Architecture. Sixteen have a centralized architecture, three are decentralized, and the remaining nine do not present enough information on this topic;

License Most did not mention a license; those that did were mostly open-source (*e.g.*, GPL v2, GPL v3, Apache v2 and LGPL v3);

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

Table 2. VP approaches applied to IoT and their characteristics. Small circles (•) mean *yes*, hyphens (-) means *no information available*, empty means *no*.

Tool	Scope[8]	Centralized	License	Tier	Scalability	Programming	Web-based
Belsa et al. [5]	*	•	-	Cloud	High	Hybrid	•
Ivy [25]	*	•	-	Cloud	Medium[7]	Purely visual	
Ghiani et al. [29]	HA	•	-	Cloud	-	Form-based	•
ViSiT [2]	*	•	-	Cloud	High	Hybrid	•
Valsamakis et al. [53]	AAL	•	-	Cloud	-	Hybrid	•
WireMe [42]	EDU, HA	•	-	Cloud	-	Hybrid	
VIPLE [17]	EDU	•	-	Cloud	-	Hybrid	
Smart Block [4]	HA	•	-	Cloud	-	Hybrid	•
PWCT [28]	*	•	GPL v2	-[1]	High	Hybrid	
DDF [31]	-		Apache v2	Fog	High	Hybrid	•
GIMLE [51]	IND	•	-	Cloud	High	Hybrid	•
DDFlow [41]	SEC		-	Fog/Edge	-	Hybrid	•
Kefalakis et al. [35]	-	•	LGPL v3[3]	Cloud	-	Hybrid	
Eterovic et al. [26]	HA	-[4]	-	-	-	Hybrid	-
FRED [8]	*	•	-[5]	Cloud	High	Hybrid	•
WoTFlow [7]	-		-	Fog/Edge	-	Hybrid	•
Besari et al. [6] [49]	EDU	•	-	Cloud	-	Hybrid	
CharIoT [50]	HA	•[6]	-	Cloud/Edge[6]	High[6]	Form-based	•
Desolda et al. [18]	SM	-	-	-	-	Hybrid	
Eun et al. [27]	HA	•	-	-	-	Form-based	•

[1] Used for several purposes, did not specify the tier it is located in regarding IoT.
[2] Since it uses Node-RED, this information was based on its architecture.
[3] Under the same license of OpenIoT.
[4] No information *w.r.t* the architecture of the environment created, only the VPL.
[5] No information about the license is given, but further research discovered that it had paid plans and no source code available.
[6] CharIoT uses the Giotto stack [1] from where we retrieved this information.
[7] Certainty regarding this information is low.
[8] Several (*), Home Automation (HA), Ambient Assisted Living (AAL), Education (EDU), Industry (IND), Security (SEC), and Smart Museums (SM).

Scalability. The majority do not consider scalability (*e.g.* the number of devices they were tested with); those that do, claim high scalability;

Programming. 22 employ a hybrid text and visual system VP paradigm, while three use a purely visual, and the other three a form-based one;

Web-based. The majority of analyzed approaches are web-based. One tool did not specify the environment, only mentioned being a VPL.

The following paragraphs present an evolution-over-time analysis and attempt to give an answer to the aforementioned research questions.

Evolution Analysis To understand the evolution of VP approaches applied to IoT, we analyzed the years when the selected papers were published and the surveyed approaches launched. Fig. 2 clearly display an increased trend in research during the last years.

Research Questions The research questions presented in Section 2 served to direct this SLR and obtain answers to relevant questions regarding the available visual programming approaches for IoT. These answers are:

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

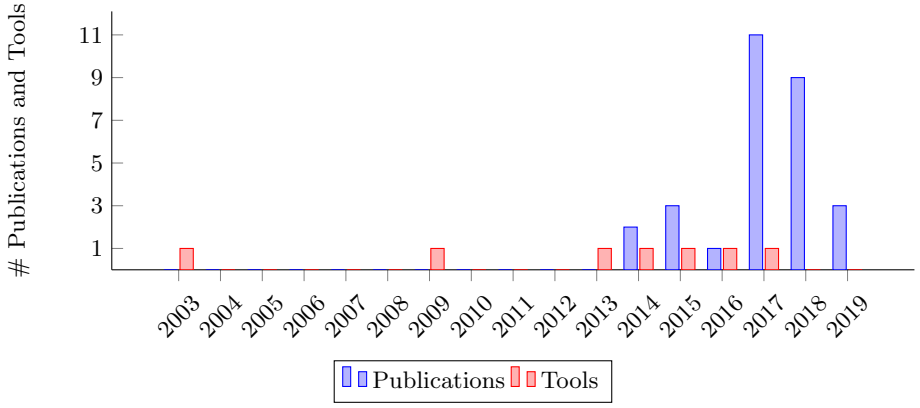

Fig. 2. Evolution of publications and number of visual approaches per year.

RQ1. *What are the relevant VP approaches applied to distributed computation and orchestration in IoT?* From the analyzed approaches in Section 3, we found 28 that share these concerns in IoT-scope;

RQ2. *What architectures and tiers characterize the approaches found in* **RQ1***?* Tables 2 and 3 give an overview of the surveyed approaches characteristics. Our analysis (Section 3.1) concludes most of them have a centralized architecture and work in the Cloud tier;

RQ3. *What was the evolution of VP approaches applied to IoT over the years focusing on its decentralized operation?* As seen in Section 3.1 and Fig. 2, some approaches share this concern since 2003, though 2017–2018 saw a significant increase in publications focusing on it.

4 Visually-defined Distributed Computing

From the analyzed approaches, we found four trying to tackle visual and decentralized orchestration in IoT. We discuss them in the following subsections.

Table 3. Characterization of the visual programming approaches for IoT [47].

Tool	Scope[3]	Centralized	License	Tier	Scalability	Programming	Web-based
Node-Red [39]	*	Yes	Apache v2	Cloud/Edge	High	Hybrid	Yes
NETLab Toolkit [38]	N/A	N/A	GPL	Edge[2]	N/A	Hybrid	Yes
NooDL [40]	*	N/A	NooDL[1]	Cloud[2]	N/A	Hybrid	No
DGLux5 [40]	*	N/A	DGLux	Cloud/Fog[2]	High[2]	Purely visual	No
AT&T Flow Designer [3]	*	N/A	GPL v3	Cloud[2]	High[2]	Hybrid	Yes
GraspIO [32]	EDU	N/A	BSD	Cloud[2]	N/A	Purely visual	No
Wyliodrin [55]	*	N/A	GPL v3	All[2]	N/A	Hybrid	Yes
Zenodys [12]	*	N/A	GPL v3	Cloud[2]	High[2]	Hybrid	Yes

[1] Available at https://www.noodl.net/eula
[2] Certainty regarding this information is low.
[3] Several (*), Education (EDU), and Not Available (N/A).

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

4.1 DDF

WoTFlow [7], DDF [41], and subsequent works [31, 30] are systems extending Node-RED and focusing on Smart Cities. Their goal is to make it more suitable for developing fog-based applications that are context-dependent on edge devices where they operate. DDF starts by implementing D-NR (Distributed Node-RED), which contains processes that can run across devices in local networks and servers in the Cloud. The application, called *flow*, is built with a VP environment, running in a development server. All the other devices running D-NR subscribe to an MQTT topic that contains the status of the flow. When a flow is deployed, all devices running D-NR are notified and subsequently analyze the given flow. Based on a set of constraints, they decide which nodes they may need to deploy locally and which sub-flow (parts of a flow) must be shared with other devices. Each device has characteristics, from its computational resources, such as bandwidth and available storage, to its location. The developer can insert constraints into the flow by specifying which device a sub-flow must be deployed in or the computational resources needed. Further, each device must be inserted manually into the system by a technician.

Subsequent work focused on support for the Smart Cities domain, including the deployment of multiple instances of devices running the same sub-flow and the support for more complex deployment constraints of the application flow [31]. The developer can specify requirements for each node on device identification, computing resources needed (CPU and memory), and physical location. In addition to these improvements, the coordination between nodes in the fog was tackled by introducing a coordinator node. This node is responsible for synchronizing the device's context with the one given by the centralized coordinator. Recent work [30], support for CPSCN (Cyber-Physical Social Computing and Networking) was implemented, making it possible to facilitate the development of large scale CPSCN applications. Additionally, to make this possible, the contextual data and application data were separated so that the application data is only used for computation activities. The contextual data is used to coordinate the communication between those activities.

4.2 uFlow and FogFlow

Szydlo *et al.* [48] focused on the transformation and decomposition of data flow. Parts of the flow can be translated into executable parts, such as Lua. Their contribution includes data flow transformation concepts, a new portable run-time environment (`uFlow`) targetting resource-constrained embedded devices, and its integration with Node-RED. Their solution transforms a given data flow by allowing the developer to choose the computing operations run on the devices. These operations are implemented using `uFlow`. The communication between the devices requires a Cloud layer, without support for peer-to-device communication. The results are promising, showing a decrease in the number of measurements made by the sensors. However, there is room for improvement *w.r.t.* the

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

automatic decomposition and partitioning of the initial flow, and detecting current conditions in deciding when to move computations between fog and cloud. Later, the authors proposed `FogFlow`[48], which enables the decomposition into heterogeneous IoT environments according to a chosen decomposition schema. To achieve a certain level of decentralization and heterogeneity, they abstract the application definition from its architecture and rely on graph representations to provide an unambiguous, well-defined computation model. The application definition is infrastructure-independent and only contains data processing logic, and its execution should be possible on different sets of devices with different capabilities. Several algorithms for flow decomposition are mentioned [37, 33], but none were explored/provided results.

4.3 FogFlow (yet another)

A different tool with the same name `FogFlow` by Cheng *et al.* [16, 15] proposes a standards-based programming model for Fog Computing and scalable context management. The authors start by extending the dataflow programming model with hints to facilitate the development of fog applications. The scalable context management introduces a distributed approach, which allows overcoming the limits in a centralized context, achieving much better performance in throughput, response time, and scalability. The `FogFlow` framework focuses on a Smart City Platform use case, separated into three areas: (1) Service Management, typically hosted in the Cloud, (2) Data Processing, present in cloud and edge devices, and (3) Context Management, which is separated in a device discovery unit hosted in the Cloud and IoT brokers scattered in Edge and Cloud. This was later improved to empower infrastructure providers with an environment that allows them to build decentralized IoT systems faster, with increased stability and scalability. Dynamic data representing the IoT system flows are orchestrated between sensors (Producers) and actuators (Consumers). An application is first designed using the `FogFlow` Task Designer (a hybrid text and VP environment), which outputs an abstraction called *Service Template*. This abstraction contains details about the resources needed for each part of the system. Once the Service Template is submitted, the framework determines how to instantiate it using the context data available. Each task is associated with an operator (a Docker image), and its assignment is based on (1) how many resources are available on each edge node, (2) the location of data sources, and (3) the prediction of workload. Edge nodes are autonomous since they can make their own decisions based on their local context without relying on the central Cloud. Obviously, the dependency in Docker completely discards constrained devices.

4.4 DDFlow

`DDFlow` [41], presents another distributed approach by extending Node-RED with a system run-time that supports dynamic scaling and adaption of application deployments. The distributed system coordinator maintains the state and assigns tasks to available devices, minimizing end-to-end latency. Dataflow notions of

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

Table 4. IoT decentralized visual programming approaches and their characteristics.

Tool	Leveraging edge devices	Communication capabilities	Open-source	Computation decomposition	Run-time adaptation
DDF [30]	Limited[1]	Yes	Yes	Limited[2]	Yes
`uFlow` [48]	Yes	Limited[3]	No	Limited[2]	Limited[3]
`FogFlow` [15]	Yes	N/A	Yes	Limited[2]	Yes
`DDFlow` [41]	Limited[4]	Yes	No	Limited[2]	Yes

[1] Assumes all devices run Node-RED (doesn't apply to constrained devices).
[2] Does not specify the algorithm used.
[3] Communication between devices is made through the cloud (Internet-dependent).
[4] Assumes all devices have a list of specific services they can provide.

node and *wire* are expanded, with a *node* in `DDFlow` representing an instantiation of a task deployed in a device, receiving inputs and generating outputs. *Nodes* can be constrained in their assignment by optional parameters, *Device*, and *Region*, inserted by the developer. A *wire* connects two or more nodes and can have three types: *Stream* (one-to-one), *Broadcast* (one-to-many), and *Unite* (many-to-one).

In a `DDFlow` system, each device has a set of capabilities and a list of services that correspond to an implementation of a *Node*. The devices communicate this information through their Device Manager or a proxy if it is a constrained device. The coordinator is a web server responsible for managing the `DDFlow` applications. It is composed of: (1) a VP environment where `DDFlow` application are built, (2) a Deployment Manager that communicates with the Device Managers of the devices, and (3) a Placement Solver, responsible for decomposing and assigning tasks to the available devices. When an application is deployed, a network topology graph and a task graph are constructed based on the real-time information retrieved from the devices. The coordinator proceeds with mapping tasks to devices by minimizing the task graph's end-to-end latency of the longest path. Dynamic adaptation is supported by monitoring the system; if changes in the network are detected, such as the failure or disconnection of a device, adjustments in the assignment of tasks are made. The coordinator can also be replicated into many devices to improve the system's reliability and fault-tolerance. They show `DDFlow` recovering from network degradation or device overload, whereas in a centralized system this would likely cause its (total) failure.

5 Conclusion

The mentioned approaches were characterized based on their mentions or support for the following features and characteristics:

Leveraging edge devices. A decentralized architecture takes advantage of the computational power of the devices in the network, assigning them tasks. However, some approaches have limitations on the type of supported devices or only focus on the Fog tier and not Edge;

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

Communication capabilities. The orchestrator must know each device's capabilities so that it can make informed decisions regarding the decomposition and assignment of tasks;

Open-source. The license of software or tool is essential in terms of its usability. Open-source allows access to the code, making it possible for its analysis, improvement, and reuse;

Computation decomposition. To implement a decentralized architecture, it is important to decompose the computation of the system into independent and logical tasks that can be assigned to devices. This is made using algorithms, which can be specified or mentioned;

Run-time adaptation. A system needs to adapt to run-time changes, such as non-availability of devices or even network failure. The system notices these events and can take action to circumvent the problems and keep functioning;

From the analysis of Table 4, we can conclude that the current research for visual programming approaches that leverage the decentralized nature of IoT is incomplete. All the surveyed approaches leverage the devices in the network but in a different way. `DFF` assumes that all devices run Node-RED, limiting the types of devices used. `FogFlow` and `uFlow` are the only ones that specify how they truly leverage constrained devices, with the transformation of sub-flows into Lua code. `DDFlow` assumes that all devices have a list of specific services they can provide that should match the node assigned to them. Regarding the method used to decompose and assign computations to the available devices, `DDFlow` describes the process using the longest path algorithm focused on reducing end-to-end latency between devices. `FogFlow` and `uFlow` mention several algorithms that could be used but do not specify which one was implemented. Both `DDF` and `FogFlow` do not specify the algorithm used besides some constraints but are the only ones with accessible source code and an open-source license. All the surveyed approaches claim to support run-time adaptation to changes in the system, such as device failures.

Acknowledgement. This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020.

References

1. Agarwal, Y., Dey, A.K.: Toward building a safe, secure, and easy-to-use internet of things infrastructure. IEEE Computer **49**(4), 88–91 (2016)
2. Akiki, P.A., Bandara, A.K., Yu, Y.: Visual simple transformations: Empowering end-users to wire internet of things objects. ACM Transactions on Computer-Human Interaction **24**(2), 1–43 (Apr 2017). https://doi.org/10.1145/3057857
3. AT&T Mobility LLC: AT&T Flow Designer. Available: https://flow.att.com, last access 2020
4. Bak, N., Chang, B.M., Choi, K.: Smart Block: A Visual Programming Environment for SmartThings. In: International Computer Software and Applications Conference. vol. 2, pp. 32–37 (2018). https://doi.org/10.1109/COMPSAC.2018.10199

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

5. Belsa, A., Sarabia-Jacome, D., Palau, C.E., Esteve, M.: Flow-based programming interoperability solution for IoT platform applications. In: IEEE International Conference on Cloud Engineering, IC2E 2018. pp. 304–309 (2018)
6. Besari, A.R.A., Wobowo, I.K., Sukaridhoto, S., Setiawan, R., Rizqullah, M.R.: Preliminary design of mobile visual programming apps for Internet of Things applications based on Raspberry Pi 3 platform. In: International Electronics Symposium on Knowledge Creation and Intelligent Computing. pp. 50–54 (2017)
7. Blackstock, M., Lea, R.: Toward a distributed data flow platform for the Web of Things (Distributed Node-RED). In: ACM International Conference Proceeding Series. vol. 08, pp. 34–39 (2014)
8. Blackstock, M., Lea, R.: FRED: A hosted data flow platform for the IoT. In: 1st International Workshop on Mashups of Things and APIs (2016)
9. Boshernitsan, M., Downes, M.S.: Visual programming languages: a survey. Tech. Rep. UCB/CSD-04-1368, EECS Department, University of California, Berkeley (Dec 2004), http://www2.eecs.berkeley.edu/Pubs/TechRpts/2004/6201.html
10. Burnett, M., Kulesza, T.: End-User Development in Internet of Things: We the People. In International Reports on Socio-Informatics (IRSI), Proceedings of the CHI 2015 - Workshop on End User Development in the Internet of Things Era **12**(2), 81–86 (2015)
11. Buyya, R., Dastjerdi, A.V.: Internet of Things: Principles and Paradigms. Elsevier (2016)
12. B.V., Z.: Zenodys. Available: https://www.zenodys.com/, last access 2020
13. Chang, S.: Handbook of Software Engineering and Knowledge Engineering. World Scientific Publishing Co. (2002)
14. Chen, S., Xu, H., Liu, D., Hu, B., Wang, H.: A vision of IoT: Applications, challenges, and opportunities with China Perspective. IEEE Internet of Things Journal **1**(4), 349–359 (2014)
15. Cheng, B., Kovacs, E., Kitazawa, A., Terasawa, K., Hada, T., Takeuchi, M.: Fogflow: Orchestrating iot services over cloud and edges. NEC Technical Journal **13**, 48–53 (11 2018)
16. Cheng, B., Solmaz, G., Cirillo, F., Kovacs, E., Terasawa, K., Kitazawa, A.: Fogflow: Easy programming of iot services over cloud and edges for smart cities. IEEE Internet of Things Journal **PP**, 1–1 (08 2017)
17. De Luca, G., Li, Z., Mian, S., Chen, Y.: Visual programming language environment for different IoT and robotics platforms in computer science education. CAAI Transactions on Intelligence Technology **3**(2), 119–130 (2018)
18. Desolda, G., Malizia, A., Turchi, T.: A tangible-programming technology supporting end-user development of smart-environments. In: Proceedings of the Workshop on Advanced Visual Interfaces. pp. 59:1—59:3. ACM, USA (2018)
19. Dias, J.P., Couto, F., Paiva, A.C.R., Ferreira, H.S.: A brief overview of existing tools for testing the internet-of-things. In: IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 104–109 (2018)
20. Dias, J.P., Faria, J.P., Ferreira, H.S.: A reactive and model-based approach for developing internet-of-things systems. In: 11th International Conference on the Quality of Information and Communications Technology. pp. 276–281 (2018)
21. Dias, J.P., Ferreira, H.S., Sousa, T.B.: Testing and deployment patterns for the internet-of-things. In: Proceedings of the 24th European Conference on Pattern Languages of Programs. EuroPLop '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3361149.3361165

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

22. Dias, J.P., Restivo, A., Ferreira, H.S.: Empowering visual internet-of-things mashups with self-healing capabilities. In: 2021 IEEE/ACM 2nd International Workshop on Software Engineering Research Practices for the Internet of Things (SERP4IoT) (2021)
23. Dias, J.P., Sousa, T.B., Restivo, A., Ferreira, H.S.: A pattern-language for self-healing internet-of-things systems. In: Proceedings of the 25th European Conference on Pattern Languages of Programs. EuroPLop '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3361149.3361165
24. Dias, J.P., Lima, B., Faria, J.P., Restivo, A., Ferreira, H.S.: Visual self-healing modelling for reliable internet-of-things systems. In: Proceedings of the 20th International Conference on Computational Science. pp. 27–36. Springer (2020)
25. Ens, B., Anderson, F., Grossman, T., Annett, M., Irani, P., Fitzmaurice, G.: Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments. In: Proceedings - Graphics Interface. pp. 156–163 (2017)
26. Eterovic, T., Kaljic, E., Donko, D., Salihbegovic, A., Ribic, S.: An Internet of Things visual domain specific modeling language based on UML. In: Proceedings of the 25th International Conference on Information, Communication and Automation Technologies (2015)
27. Eun, S., Jung, J., Yun, Y.S., So, S.S., Heo, J., Min, H.: An end user development platform based on dataflow approach for IoT devices. Journal of Intelligent and Fuzzy Systems **35**(6), 6125–6131 (2018). https://doi.org/10.3233/JIFS-169852
28. Fayed, M.S., Al-Qurishi, M., Alamri, A., Al-Daraiseh, A.A.: PWCT: Visual language for IoT and cloud computing applications and systems. In: ACM International Conference Proceeding Series (2017)
29. Ghiani, G., Manca, M., Paterno, F., Santoro, C.: Personalization of context-dependent applications through trigger-action rules. ACM Transactions on Computer-Human Interaction **24**(2), 14:1—14:33 (Apr 2017)
30. Giang, N.K., Lea, R., Leung, V.C.M.: Exogenous coordination for building fog-based cyber physical social computing and networking systems. IEEE Access **6**, 31740–31749 (2018)
31. Giang, N.K., Blackstock, M., Lea, R., Leung, V.C.: Developing IoT applications in the Fog: A Distributed Dataflow approach. In: Proceedings of the 5th International Conference on the Internet of Things. pp. 155–162 (2015)
32. Grasp IO Innovations Pvt. Ltd.: GraspIO. Available: https://www.grasp.io/, last access 2020
33. Gupta, H., Vahid Dastjerdi, A., Ghosh, S.K., Buyya, R.: iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Software: Practice and Experience **47**(9), 1275–1296 (2017)
34. Ihirwe, F., Di Ruscio, D., Mazzini, S., Pierini, P., Pierantonio, A.: Low-code engineering for internet of things: A state of research. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. MODELS '20, USA (2020)
35. Kefalakis, N., Soldatos, J., Anagnostopoulos, A., Dimitropoulos, P.: A visual paradigm for iot solutions development. In: Interoperability and Open-Source Solutions for the Internet of Things. vol. 9001, pp. 26–45. Springer, Cham (2015)
36. Lago, A.S., Dias, J.P., Ferreira, H.S.: Managing non-trivial internet-of-things systems with conversational assistants: A prototype and a feasibility experiment. Journal of Computational Science **51**, 101324 (2021)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34

37. NAAS, M.I., Lemarchand, L., Boukhobza, J., Raipin, P.: A graph partitioning-based heuristic for runtime iot data placement strategies in a fog infrastructure. In: 33rd Annual ACM Symposium on Applied Computing. p. 767–774 (2018)
38. NETLabTK: Tools for Tangible Design. Available: www.netlabtoolkit.org/, last access 2020
39. Node-RED. Available: https://nodered.org/, last access 2020
40. NooDL. Available: https://classic.getnoodl.com/, last access 2020
41. Noor, J., Tseng, H.Y., Garcia, L., Srivastava, M.: DDFlow: Visualized declarative programming for heterogeneous IoT networks. In: Proceedings of the 2019 Internet of Things Design and Implementation. pp. 172–177. ACM (2019)
42. Pathirana, D., Sonnadara, S., Hettiarachchi, M., Siriwardana, H., Silva, C.: WireMe - IoT development platform for everyone. In: 3rd International Moratuwa Engineering Research Conference, MERCon 2017. pp. 93–98 (2017)
43. Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology **64**, 1–18 (2015)
44. Pinto, D., Dias, J.P., Sereno Ferreira, H.: Dynamic allocation of serverless functions in iot environments. In: 2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC). pp. 1–8 (Oct 2018)
45. Prehofer, C., Chiarabini, L.: From IoT Mashups to Model-based IoT. W3C Workshop on the Web of Things (2013)
46. Ramadas, A., Domingues, G., Dias, J.P., Aguiar, A., Ferreira, H.S.: Patterns for Things that Fail. In: Proceedings of the 24th Conference on Pattern Languages of Programs. PLoP '17, ACM - Association for Computing Machinery (2017)
47. Ray, P.P.: A Survey on Visual Programming Languages in Internet of Things. Scientific Programming **2017**, 1–6 (2017)
48. Sendorek, J., Szydlo, T., Windak, M., Brzoza-Woch, R.: Fogflow - computation organization for heterogeneous fog computing environments. In: Computational Science – ICCS 2019. pp. 634–647. Springer International Publishing, Cham (2019)
49. Setiawan, R., Anom Besari, A.R., Wibowo, I.K., Rizqullah, M.R., Agata, D.: Mobile visual programming apps for internet of things applications based on raspberry Pi 3 platform. In: International Electronics Symposium on Knowledge Creation and Intelligent Computing. pp. 199–204 (Oct 2019)
50. Tomlein, M., Boovaraghavan, S., Agarwal, Y., Dey, A.K.: CharIoT: An end-user programming environment for the IoT. In: ACM International Conference Proceeding Series (2017). https://doi.org/10.1145/3131542.3140261
51. Tomlein, M., Grønbæk, K.: A visual programming approach based on domain ontologies for configuring industrial IoT installations. In: ACM International Conference Proceeding Series (2017). https://doi.org/10.1145/3131542.3131552
52. Torres, D., Dias, J.P., Restivo, A., Ferreira, H.S.: Real-time feedback in node-red for iot development: An empirical study. In: IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications. pp. 1–8 (2020)
53. Valsamakis, Y., Savidis, A.: Visual end-user programming of personalized AAL in the internet of things. In: Lecture Notes in Computer Science. vol. 10217 LNCS, pp. 159–174 (2017). https://doi.org/10.1007/978-3-319-56997-0_13
54. Varshney, P., Simmhan, Y.: Demystifying fog computing: Characterizing architectures, applications and abstractions. In: 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC). pp. 115–124. IEEE (2017)
55. Wyliodrin. Available: https://wyliodrin.com/, last access 2020
56. Zhang, K., Han, D., Feng, H.: Research on the complexity in internet of things. IET Conference Publications **2010**(571 CP), 395–398 (2010)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_34