Towards Distributing Effort in Validating Run-Time Models for Internet of Things Architectures

Erik M. Fredericks
College of Computing
Grand Valley State University
Allendale, MI, USA
frederer@gvsu.edu

Byron DeVries
College of Computing
Grand Valley State University
Allendale, MI, USA
devrieby@gvsu.edu

***Abstract*—Internet of Things (IoT) applications can comprise heterogeneous components that are responsible for delivering a quality of service that ideally supports the objectives of the system as a whole. However, individual components may have competing objectives with other components, be improperly implemented, and/or experience uncertainty that impacts the overall quality of service. Run-time modeling of such systems provides an approach for quantifying objective success, however distributing the outputs of such models in a peer-to-peer environment is non-trivial given the constraints of IoT environments. This paper presents a proof-of-concept framework for managing adaptive P2P networks using goal-based modeling techniques and fuzzy logic operators to enable run-time flexibility in goal satisfaction and describes challenges to overcome.**

***Index Terms*—software engineering, goal modeling, utility functions, internet of things, RELAX, fuzzy logic**

I. Introduction

Peer to peer (P2P) Internet of Things (IoT) applications must manage a number of difficult constraints to ensure the success of the overall application. Specifically, managing an ad hoc network of potentially-heterogeneous devices that must deliver a high quality of service is non-trivial as coordination of different device capabilities must occur to satisfy key objectives [1]. Moreover, uncertainty can manifest in terms of environmental problems (e.g., precipitation, wildfires, etc.), hardware can intermittently or terminally fail, and humans/animals may interfere with deployed devices [2]–[4]. Depending on the construction of the network, a temporary disruption of communication with one or more nodes can effectively break the overall system, thereby violating its key objectives.This paper introduces and motivates `RELAXed-IoT`, a run-time framework for leveraging goal models, utility functions, and fuzzy logic operators to support online decision making in noisy environments.

In this paper we discuss background and related work to the domain of a goal-based IoT application, including how to mathematically quantify them in a lightweight fashion. Additionally, we discuss how fuzzy logic operators (i.e., RELAX operators [5], [6]) can be applied to enable greater tolerance of transient uncertainty. Finally, we discuss each step of our proposed framework with a motivating example and the challenges anticipated for each. Our overall motivation and goal for this paper is to lay the groundwork for a deployable IoT network governed by heterogeneous goal models that each support RELAX operators to enable run-time resilience against uncertainty.

II. Background and Related Work

Goal modeling provides a high-level approach for visualizing the key objectives and relationships between those objectives for a system. Common approaches include KAOS [7] and iStar [8]. For example, a goal (i.e., $Goal_B$) for a sensor mote may be specified as $[Achieve] Battery \geq 5\%$ in KAOS notation, indicating that this is a non-invariant goal (denoted by $Achieve$) requiring that the battery level be above 5%. Invariant goals are typically specified as $Avoid$ or $Maintain$ and cannot be violated. Previously, goal modeling has been used in the IoT space to manage the hierarchical concerns inherent between agents with competing concerns [9]–[11], proposed for use in security and efficiency of applications [12], and used for developing social IoT networks [13].

While our approach uses goal modeling at its core, our focus is on leveraging utility functions [14] and RELAX operators [5] to enable run-time introspection and flexibility in goal satisfaction. Goal modeling has also been deployed in multi-agent systems (i.e., systems comprising multiple independent agents that have their own objectives) [15], self-adaptive systems (i.e., systems that can monitor themselves and self-reconfigure to resolve transient issues) [16], [17], and software product lines (i.e., codebases that support a range of applications with the same/similar base code) [15].

Utility functions can calculate the performance of a software engineering artifact (e.g., goals, requirements, tests) by using a mathematical formula for quantification [14]. Such functions are generally “low overhead” and are ideal for IoT-based systems where processing time is precious. One approach for calculating utility values normalizes a utility function within $[0.0, 1.0]$, where 0.0 indicates a violation, 1.0 indicates satisfaction, and any value in between indicates the degree of satisfaction. Tolerances/thresholds can be defined within the range to specify “cutoff” values where performance has degraded and a resolution is required. The degree of satisfaction, coupled with tolerance definitions, can be used as part of a self-adaptive overlay to enable run-time reconfigurations

We gratefully acknowledge support from Grand Valley State University.

to temporarily tolerate failure. A sample utility function is defined in Equation 1, demonstrating a utility value for $Goal_B$ that ensures battery life for a sensor mote never falls below 5% and returns a linear value correlated with decreasing battery. A threshold value of 10% can be specified to indicate that an adaptation is necessary prior to critical failure at 5%.

$$util_{batt} = \begin{cases} [1.0, x, 0.0] & \text{if } batt \geq 5\% \\ 0.0 & \text{else} \end{cases} \qquad (1)$$

Utility functions have been used in the IoT space, particularly using the MQTT (Message Queuing Telemetry Transport) protocol to determine a reputation model for participating nodes [18]. Additionally, utility has a parallel definition in this space for defining the usefulness of a particular node that can be akin to measuring performance [19]. Our approach focuses on calculating performance via mathematical functions derived for each key objective. There exists other approaches for validating models at run time that use significantly-deeper formalisms [20], however our inclusion of deriving and including run-time monitoring of utility functions aims to keep overhead minimal to a distributed, embedded environment.
RELAX is an approach for introducing flexibility into artifact monitoring by using fuzzy logic functions in non-invariants to temporarily tolerate transient problems [5], [6]. For example, $Goal_B$ specifies that battery life must never fall below 5% and an associated RELAX operator may specify that the battery life is "AS HIGH AS POSSIBLE" and uses a *right shoulder membership function* as shown in Figure 1. This operator can be applied to the utility function from Equation 1 to enable additional flexibility. Note that there are a number of other RELAX operators and fuzzy logic membership functions as well, including AS CLOSE AS POSSIBLE TO (triangle membership function), AS EARLY AS POSSIBLE (left shoulder membership function), AS MANY AS POSSIBLE (right shoulder membership function), among others. Each fuzzy logic membership function provides a configurable function for desired value(s), minimum/maximum value(s) allowed, and the output utility value. Similarly, fuzzy logic has been leveraged in the IoT space for optimizing message transmission policies [21]. RELAXed-IoT aims to include either manually- [5] or automatically-applied [6] RELAX operators to the goal models derived for each participating device within the network.

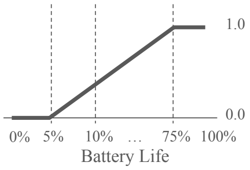

Fig. 1. Right shoulder RELAX operator for battery life goal ($Goal_B$). Its utility value is 1.0 when battery life is greater than 75%, 0.0 when less than 5%, and linearly decreasing between 75% and 5%.

III. Distributed Modeling at Run Time

This section describes our proposed framework RELAXed-IoT for modeling and managing heterogeneous IoT devices at run time. We discuss each step of the framework with a motivating example and describe the challenges expected for each.

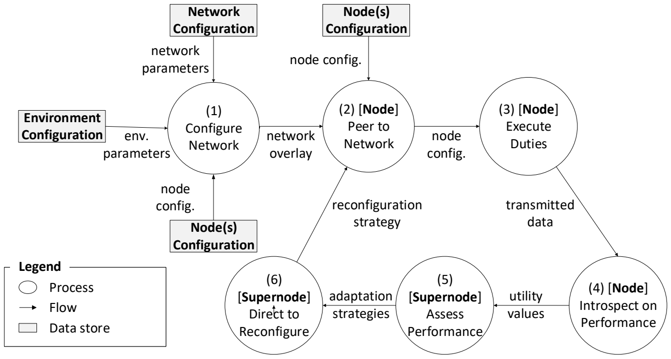

Fig. 2. Data flow diagram of RELAXed-IoT.

A. Illustrative Example

To motivate RELAXed-IoT we will consider a smart agriculture P2P network tasked with monitoring a farm comprising multiple growing and livestock areas, such as corn, soy, and cattle, with the aim of managing the competing concerns of optimizing/maintaining healthy fields, livestock, and crop yield [22], [23]. Specific use cases for the devices can include monitoring soil moisture and quality (i.e., necessitating hydration and/or fertilizer application), ensuring livestock remain within designated areas, and that emergency conditions are communicated to appropriate parties (e.g., gate left open, fire/flooding hazard, etc.). Note that the intention of RELAXed-IoT is to be applicable to *any* P2P application where goal models, utility functions, and RELAXations can be derived to support run-time introspection and reconfiguration.

A number of sensor motes (i.e., nodes) are available of varying capacity. Super nodes are motes with a larger storage capacity, processing capabilities, and communication range. Additionally, super nodes can act as cluster heads within a clustered P2P environment. Within RELAXed-IoT, super nodes are additionally responsible for maintaining requisite information about each node within its cluster, including its goal model, utility functions, and any goal RELAXations that are applied. This information is necessary to determine which goals are invariant or non-invariant and to provide information needed for any reconfiguration decisions. Nodes are required to calculate and provide their respective utility values to the super node periodically, where the super node will decide if a reconfiguration is warranted based on utility scores.

B. The RELAXed-IoT Framework

RELAXed-IoT comprises a P2P network involving normal nodes (hereafter nodes) and super nodes. *Nodes* are responsible for introspecting on their performance and transmitting those results to super nodes either cyclically or on request. In addition to normal node responsibilities, *super nodes* are responsible for monitoring nodes within their local cluster and act as cluster heads for P2P communication. Each node at minimum is required to have a corresponding goal model, a set of utility functions to calculate performance of each goal, and a potential set of RELAX operators that can be applied to desired goals to provide a tolerance to performance degradation. We next discuss each step of our proposed `RELAXed-IoT` framework, as illustrated in Figure 2.

1) Configure Network: For its initial deployment, an IoT network must be deployed with an appropriate networking scheme to enable P2P communication. The network should comprise a minimum number of super nodes to support the deployment and define respective clusters. Additionally, a P2P communication protocol must also be chosen. For the purpose of this paper we assume a relevant protocol is selected, however an ID-based protocol (e.g., Chord [24]) may be preferable to minimize effort in node lookup and identification when determining if an adaptation is necessary. This step would result in a minimum viable network for the target application, however additional nodes are expected to be peered (2) to provide more sensing capabilities.

Motivating Example: A new network is to be deployed at a farm as previously described, with key concerns including maximizing crop and livestock yield, minimizing overall costs and maintenance of the network, and forecasting needs for future seasons. Moreover, the information in aggregate can be used to predict crop and grazing area rotations to further enhance yield.

Challenges for this step include determining the number and location of initially-deployed super nodes, as well as deriving initial goal models/utility functions. Additionally, the selection of the P2P communication protocol and data dissemination strategies are critical to maximize network lifetime while minimizing the number of required nodes, along with minimizing the overall cost of the network.

2) Peer to Network: A node that is being added to the network must be configured to follow the specified P2P protocol and advertise its capabilities to the network upon joining. In addition to "normal" activities for joining a network, a `RELAXed-IoT` node must advertise its goal model, corresponding utility functions, and any RELAX operators its respective cluster head (i.e., super node). The handshake between new node and cluster head will also include the periodic cycle at which utility values will be transmitted from node to super node to enable scheduling of potential adaptations. Depending on the overhead required, node information can be mirrored to all other clusters in the network as well.

Motivating Example: Following (1), a set of sensor motes are to be deployed across the farm to monitor soil, weather, and livestock conditions. These motes can comprise low cost sensors that report on soil metrics (e.g., humidity, acidity, etc.) to higher cost cameras that track the movement of cattle across the fields. Upon activation, each node broadcasts its desire to join the network and transmits its salient metadata to other nodes within range to identify a path to its local cluster head (i.e., super node).

Challenges for this step include minimizing network overhead when transmitting all metadata from new nodes to super nodes (i.e., goal model, utility function, and RELAX operator representations) and minimizing the storage required for that metadata. As this is a P2P application, an additional challenge lies in ensuring that the data transmitted does not overburden low-capability devices in between the new node and its respective cluster head.

3) Execute Node Duties: Following peering, nodes perform their typical duties in terms of monitoring and communicating as configured. These duties are programmed according to the specified goals defined for the node. Periodically, each node evaluates its performance according to its utility functions (that assess each goal) and transmit those values to its cluster head. If RELAX operators were applied to non-invariant goals then those will either subsume or supplement the calculated value, depending on the preference of the network designer. If a utility value for a non-invariant goal is unsatisfactory and the node can resolve the issue locally, then the node is expected to take appropriate action to rectify the problem. However, if a significant action is necessary then the cluster head is responsible for managing and executing the reconfiguration.

Motivating Example: A cattle monitoring node is responsible for capturing a grayscale image every five minutes and transmitting the image to a local super node for image processing to ensure that the requisite number of livestock are within the grazing area. Goals for this node can include capturing the image, successfully transmitting it over the network, and maintaining an acceptable battery level, among others. Upon determination that an introspection cycle has begun, each goal's utility function (and RELAX operator) is assessed to yield a utility value. The utility values are packed into an appropriate data structure supported by the communication protocol and transmitted to the cluster head for further analysis.

Challenges for this step focus on the overhead of both self-introspection internal to each individual node and the network as a whole. In constrained environments such as sensor motes, the additional storage necessary to maintain the goal models and utility functions may be non-trivial. Moreover, incorporating the additional processing effort necessary to calculate performance may exceed the capabilities of the node, depending on the complexity of the calculation. Previous works have investigated unused processor cycles for code execution [25], however this remains a difficult task. Moreover, the network will need to support the additional periodic data in terms of transmitted utility values from all nodes. Scheduling algorithms may be deployed to avoid overburdening the network, though this introduces additional complexity to the network overlay.

4) Introspect on Performance: Following utility calculation and transmission, nodes will determine if a local reconfiguration is necessary to improve utility values. This step is intended to try to minimize the number of network-level adaptations to avoid issues with overloading the network infrastructure and super nodes. Here, a node may take action to improve itself in the event that a self-adaptation is feasible. This step relies upon both the designer specifying tolerance thresholds within utility functions or RELAX operators that provide a range of flexibility in which the node can reconfigure, as well as providing localized adaptation strategies that a node may perform autonomously.

Motivating Example: Following the prior example of $Goal_B$ monitoring battery life with the RELAX operator "AS HIGH AS POSSIBLE" applied, a node may detect that its battery level is at 15%. While this is not within the "danger" range of 10%, the RELAX operator indicates that its utility value is degrading and a reconfiguration is necessary (c.f., Figure 1). Given that this is a localized problem, the node reconfigures to reduce the time its communications stack is active thereby preserving battery life. The node communicates its intent to the local cluster head so that it may also adjust its scheduling algorithm, and a human operator is notified that the node requires a battery replacement. While this does not immediately improve the utility value of $Goal_B$, its lifetime is extended significantly. Upon manual battery replacement, the utility value will return to a maximum value of 1.0.

Challenges include the ability for a constrained node to have additional capabilities in terms of self-introspection and the power to self-reconfigure. The local reconfigurations supported by the node also need to minimize impact to the network as a whole. Additionally, the orchestration between node and cluster head may be non-trivial to adjust any schedules for monitoring.

5) Aggregate and Assess Performance Information: Each super node/cluster head should have a stored representation of each node's goal model in addition to a set of RELAX operators applied to enable tolerance of transient issues. If a goal is invariant then violation indicates an immediate failure, requiring node replacement. If a goal is non-invariant then a reconfiguration can be used to resolve transient issues. There are a number of methods for inducing a reconfiguration, depending on the complexity and autonomy of the network. Ideally, the network would implement a self-adaptive feedback loop such as MAPE-K (Monitor-Analyze-Plan-Execute-Knowledge) [26], [27] that provides mechanisms for reconfiguration at run time. However, if resources are limited then manual or lightweight adaptations can be implemented as well.

In the event that a local reconfiguration is infeasible (e.g., either from not supporting local adaptation or that an adaptation would impact adjoining nodes), the cluster head analyzes all received data from its nodes to determine which adaptation is most likely to be successful. Here, the cluster head will determine if received utility values necessitate a reconfiguration, where the cluster head may direct individual nodes to update their communication strategy, select a different algorithm, or transmit a new set of parameters for its local configuration in the following step.

Motivating Example: A node within a cluster indicates that it has been damaged and has reduced sensing capabilities, where the damage is indicated by a low utility value from a goal that monitors its sensing radius. The cluster head receives this information and assesses the capabilities of the nodes in its cluster to determine which reconfiguration strategy will be most effective. Given a list of potential reconfiguration strategies, the cluster head selects one that directs all nearby nodes to increase their sensing radius, where feasible, to compensate for the region that cannot be sensed by the damaged node. As one of the nodes is damaged, the strategy includes transmission of a signal to a human operator that a node is damaged and requires eventual replacement. A tradeoff for this strategy is that nodes directed to increase their sensing radius will note decreased battery levels, resulting in lowered utility values for any goals monitoring battery life.

Challenges include defining an appropriate list of network-wide reconfiguration strategies based upon the capabilities of each node. This list of strategies may be manually or automatically generated based on the capabilities of the network. Manual generation requires a designer to explicitly specify each potential reconfiguration strategy, where the benefit is that all node strengths are known by the designer, however this can require a non-trivial amount of effort. Automatic generation can use advertised node capabilities based on transmitted goal models and utility functions. This approach requires an additional layer of engineered complexity to the network, however would be more flexible in terms of heterogeneous node types. An additional challenge lies in defining strategies that do not induce further damage to the system as a whole when executing the strategy.

6) Direct to Reconfigure: Upon completion of (5), the super nodes will determine if a reconfiguration is necessary at the cluster or network level based on received utility values. A reconfiguration may be automatically performed (i.e., leveraging a self-adaptation loop such as MAPE-K that enables run-time reconfigurations [26]) or manually performed (i.e., by a human participant replacing and/or updating a node). Note that reconfiguration is only supported for non-invariant goals, as a violation of an invariant goal is an indication of a complete system failure or safety concern that cannot be resolved.

Within a cluster, a reconfiguration can include replacing a node, accepting and implementing a live hotfix or patch, or updating configuration parameters to induce different behaviors. At the network level reconfigurations can include updates to the network overlay/topology and changes in system objectives. Our approach then loops back to (2) and continues for the duration of the deployment.

Motivating Example: For this step we provide two motivating examples to demonstrate reconfigurations: manual and automatic. A *manual* reconfiguration requires a human participant to enact change. Here, a super node may detect that a node's battery is failing by monitoring its communicated battery life. Upon its battery level falling below a defined threshold (e.g., 10%), a human operator is notified that the mote's battery must be replaced.

For an *automatic* reconfiguration the super node can direct the failing node to switch to a power optimization strategy (e.g., reducing communication range, decreasing periodic rate of sensing/communication, etc.) to extend its lifetime. The super node can also note that there is a problematic node that requires human intervention, however such an intervention may be non-critical and can wait until a person is nearby. For both cases, the monitored utility value should improve following reconfiguration. Previously super nodes (peers) have been shown to manage network reconfiguration and management [28]. Our approach is similar, but focuses on leveraging run-time model evaluation to determine when reconfigurations are necessary.

Challenges include ensuring that reconfigurations are both feasible and do not disrupt normal network activities. Additionally, network issues such as avoiding/minimizing network partitions, ensuring that all nodes are reachable, and that network channels are not overloaded are concerns to manage as well. The human factor is an additional challenge in that manual reconfigurations (i.e., node replacement) may result in improperly-deployed/configured nodes.

IV. Discussion

This paper has introduced `RELAXed-IoT`, a proof of concept framework for incorporating goal models and RELAX operators in an adaptive P2P environment. Nodes within `RELAXed-IoT` each use a goal model and associated set of utility functions for monitoring run-time performance. In addition to their normal behaviors, super nodes accept calculated utility values for each node within their cluster to determine if a system reconfiguration is necessary to improve the received values. `RELAXed-IoT` was motivated in the context of a P2P network in an agricultural application with anticipated challenges described for each step of the process. Future directions for this work include an empirical experiment of `RELAXed-IoT` in a real-world situation, automatic application of RELAX operators to each individual goal model, and incorporation of a self-adaptive feedback loop within the network to further validate our approach.

References

[1] A. Tissaoui and M. Saidi, "Uncertainty in iot for smart healthcare: Challenges, and opportunities," in *The Impact of Digital Technologies on Public Health in Developed and Developing Countries: 18th Intl. Conference, ICOST 2020, Hammamet, Tunisia, June 24–26, 2020, Proceedings 18*, pp. 232–239, Springer, 2020.

[2] B. H. C. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, and et al., "Software engineering for self-adaptive systems: A research roadmap," in *Software engineering for self-adaptive systems*, ch. Software Engineering for Self-Adaptive Systems: A Research Roadmap, pp. 1–26, Berlin, Heidelberg: Springer-Verlag, 2009.

[3] P. McKinley, S. Sadjadi, E. Kasten, and B. H. C. Cheng, "Composing adaptive software," *Computer*, vol. 37, pp. 56 – 64, July 2004.

[4] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein, "Requirements-aware systems: A research agenda for re for self-adaptive systems," in *Requirements Engineering Conference (RE), 2010 18th IEEE Intl.*, pp. 95 –103, 2010.

[5] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J. Bruel, "Relax: Incorporating uncertainty into the specification of self-adaptive systems," in *17th IEEE Intl. Requirements Engineering Conference (RE'09)*, pp. 79–88, 2009.

[6] E. M. Fredericks, B. DeVries, and B. H. C. Cheng, "Autorelax: Automatically relaxing a goal model to address uncertainty," *Empirical Software Engineering*, vol. 19, no. 5, pp. 1466–1501, 2014.

[7] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[8] M. Dowson, "Istar—an integrated project support environment," in *Proceedings of the second ACM SIGSOFT/SIGPLAN software engineering symposium on Practical software development environments*, pp. 27–33, 1987.

[9] A. Jantsch, A. Anzanpour, H. Kholerdi, I. Azimi, L. C. Siafara, A. M. Rahmani, N. TaheriNejad, P. Liljeberg, and N. Dutt, "Hierarchical dynamic goal management for iot systems," in *2018 19th Intl. Symposium on Quality Electronic Design (ISQED)*, pp. 370–375, IEEE, 2018.

[10] J. Cao, E. Kurniawan, A. Boonkajay, S. Sun, P. Popovski, and X. Zhu, "Goal-oriented integration of sensing, communication, computing, and control for mission-critical internet-of-things," *IEEE Network*, 2024.

[11] N. C. Narendra, N. Deb, and S. Das, "Dynamic contextual goal management in iot-based systems," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10708–10718, 2020.

[12] J. E. Siegel, S. Kumar, and S. E. Sarma, "The future internet of things: Secure, efficient, and model-based," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2386–2398, 2017.

[13] P. Kasnesis, C. Z. Patrikakis, D. Kogias, L. Toumanidis, and I. S. Venieris, "Cognitive friendship and goal management for the social iot," *Computers & Electrical Engineering*, vol. 58, pp. 412–428, 2017.

[14] P. deGrandis and G. Valetto, "Elicitation and utilization of application-level utility functions," in *Proc. of the 6th Intl. Conference on Autonomic Computing*, ICAC '09, pp. 107–116, ACM, 2009.

[15] I. Ayala, M. Amor, J.-M. Horcas, and L. Fuentes, "A goal-driven software product line approach for evolving multi-agent systems in the internet of things," *Knowledge-Based Systems*, vol. 184, 2019.

[16] E. M. Fredericks, "Automatically hardening a self-adaptive system against uncertainty," in *Proceedings of the 11th Intl. Workshop on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '16, pp. 16–27, 2016.

[17] N. Bencomo and A. Belaggoun, "Supporting decision-making for self-adaptive systems: from goal models to dynamic decision networks," in *Requirements Engineering: Foundation for Software Quality*, pp. 221–236, Springer, 2013.

[18] B. Aziz, P. Fremantle, R. Wei, and A. Arenas, "A utility-based reputation model for the internet of things," in *ICT Systems Security and Privacy Protection: 31st IFIP TC 11 Intl. Conference, SEC 2016, Ghent, Belgium, May 30-June 1, 2016, Proceedings 31*, pp. 261–275, Springer, 2016.

[19] Y. Hui, Z. Su, and S. Guo, "Utility based data computing scheme to provide sensing service in internet of things," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 2, pp. 337–348, 2017.

[20] S. Mitsch and A. Platzer, "Modelplex: Verified runtime validation of verified cyber-physical system models," *Formal Methods in System Design*, vol. 49, no. 1, pp. 33–74, 2016.

[21] L. Abbas, U. Shoaib, and A. K. Bashir, "Priority based dynamic spectrum management using virtual utility functions in cognitive radio enabled internet of things," *Computer Communications*, vol. 196, pp. 239–248, 2022.

[22] M. Aitkenhead, D. Donnelly, M. Coull, and H. Black, "E-smart: environmental sensing for monitoring and advising in real-time," in *Environmental Software Systems. Fostering Information Sharing: 10th IFIP WG 5.11 Intl. Symposium, ISESS 2013, Neusiedl am See, Austria, October 9-11, 2013. Proceedings 10*, pp. 129–142, Springer, 2013.

[23] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour, and E.-H. M. Aggoune, "Internet-of-things (iot)-based smart agriculture: Toward making the fields talk," *IEEE access*, vol. 7, pp. 129551–129583, 2019.

[24] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on networking*, vol. 11, no. 1, pp. 17–32, 2003.

[25] P. Domingues, L. Silva, and J. G. Silva, "Drmonitor-a distributed resource monitoring system," in *Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2003. Proceedings.*, pp. 127–133, IEEE, 2003.

[26] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, pp. 41 – 50, January 2003.

[27] M. Riegler, J. Sametinger, and M. Vierhauser, "A distributed mape-k framework for self-protective iot devices," in *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 202–208, IEEE, 2023.

[28] B. Beverly Yang and H. Garcia-Molina, "Designing a super-peer network," in *Proceedings 19th Intl. Conference on Data Engineering (Cat. No.03CH37405)*, pp. 49–60, 2003.