An Edge-based Framework for Cooperation in Internet of Things Applications

Zach Leidall, Abhishek Chandra, Jon Weissman
University of Minnesota, Twin-Cities

Abstract

Edge computing and the Internet of Things (IoT) are irrevocably intertwined and much work has proposed enhancing the IoT through the use of edge computing. These solutions have typically focused on using the edge to increase the locality of cloud applications, achieving benefits mainly in terms of lower network latency. In this work, we argue that IoT systems can benefit much more from semantic properties which are best recognized and exploited in situ, at the edge of the network where the data streams and actuators exist. We outline the idea of a real-time semantic operating system, hosted on the edge, which can provide higher performance in energy consumption, latency, accuracy, and improve not only individual application performance but that of an entire IoT infrastructure. We have implemented a prototype system and show some initial results demonstrating the efficacy of our proposed optimizations, and provide insights into how to handle some of the most critical issues faced in such a system.

1 Introduction

With immense quantities of data being produced by sensors at the edge, and the ever-growing use of real-time automated actuators by consumers at the edge, there is no denying the Internet of Things (IoT) is innately tethered to edge computing. The IoT has enabled thousands of 'smart' applications, perhaps chief among them is the concept of the smart city [15,24]. Metropolitan areas are increasingly equipped with potentially millions of sensors for things such as movement, air quality, noise, light, etc. and actuators controlling things like traffic lights, street lamps, or even digital displays for notifying users in the environment. The rise of many real-time applications which are highly geo-centric means the need for local compute resources has never been greater.

Work on edge computing up until now has largely focused on the need to provide these compute resources in an effort to avoid the large network bottleneck of utilizing the cloud [2,6,8,9,12,26]. We propose an alternative use for the edge, as a real-time semantic operating system of sorts, providing access and control over such a vast network. Such a system is motivated by the need to provide traditional OS capabilities such as timeliness, concurrency, access control, and security as well as new requirements unique to such systems like energy usage, location privacy and context-awareness.

New opportunities and challenges arise as a growing number of applications begin to coexist in a rich IoT environment. Applications in this environment are likely to have overlapping needs in terms of the sensor data streams they utilize as well as the actuators they are interested in, and it is the role of any operating system to control and optimize the use of common resources by multiple applications. As an example, consider a city planner who wishes to monitor foot traffic in real-time to be better informed of the effectiveness of walk signal timings and to discover possible obstructions. This application would benefit from the use of motion sensors, cameras, traffic signals, etc. all being monitored and controlled at the edge. Another application which may concurrently need to make use of these sensors could be police pursuing a suspect on foot. Ensuring the fast, reliable, and efficient control of these systems is important for optimizing user experience while also controlling for things like energy use, which is likely to become a serious issue with the number of sensors being deployed in these environments. The examples in this work focus on individual areas in a smart city, though our concepts could easily be applied to other, smaller locales.

Another motivating scenario, this time of a potential conflict between applications in such a system, could involve the control of traffic lights and digital displays. Police pursuing a suspect in a vehicle may wish to control traffic signals which are currently being controlled by the public transit system in an effort to speed traffic flow or block it for passing commuter or freight trains. While a simple priority level may seem like a solution, things like public safety may factor into the immediacy of such control switches or changes in the actuator occurring. Further, there may be multiple public emergencies in a particular area occurring simultaneously, and controlling public displays to advise pedestrians to stay clear of the area

may cause another point of contention requiring a nuanced approach to resolve the conflict.

In this work, we advocate and discuss the research challenges involved in making such a real-time semantic operating system for smart environments possible. As the main point of our discussion, we propose the idea of making application synergies and conflicts due to overlapping resource needs 'first class concepts' in such a system and develop optimizations based on them. To our knowledge, no such work in the IoT or edge has focused on such a system to meet these needs, which are likely to become extremely important in the future for cyber-physical systems, as illustrated by our examples. We believe the edge is more appropriate than the cloud for providing these mechanisms not only due to the real-time aspects of the applications but also for safety and connectivity concerns when it comes to managing localized equipment.

2 The Enabling Architecture

We envision a system architecture consisting of an environment with heavily dispersed sensors and actuators, such as a smart city, home, or office building. These devices are capable of interfacing with localized edge resources that would instantiate the real-time semantic OS abstractions that we advocate in this work. The edge resources could further be connected in a peer-to-peer network to provide wider coverage for distributed tasks. Figure 1 shows the architecture involving various edge devices such as PCs and mobile devices. These compute nodes are meant to interface with the local sensory and actuation devices, such as cameras, lights, etc. portrayed in the figure as surrounding the edge nodes. Applications, which could exist either on the edge or the cloud, would interface with the system by contacting the edge node responsible for controlling the desired sensors and/or actuators, indicated via the dashed circles. We focus our attention on a system for managing IoT devices rather than scheduling computation as there exist many systems which already provide such services and are complementary to our work. The purpose of this system is to not only provide the timeliness common in other edge-based solutions [2, 12, 21], but to provide a layer of semantic optimizations between IoT applications and the end devices they wish to use. Naturally, any such optimizations will be highly dependent on individual applications and their contexts. We have built a prototype of this framework, called Constellation [16], using which we evaluate some of our optimizations. It is our vision that this system would, as part of future work, be extended to encompass a P2P network of local edge nodes for latency and availability purposes.

2.1 Tasks and Objects

We envision such a system to have several abstractions. These include representations of IoT tasks such as sensor readings and actuations, as well as objects providing virtual representation of their physical device counterparts. IoT tasks are submitted to our system utilizing a declarative query language, like in tinyDB [17], but in our case IoT specific. The declarative nature of these queries allows for dynamic coordination that is specific to the applications involved. These statements are compiled to predefined code inside the Constellation OS which makes all the device driver calls. This means that no user written code runs on our system, applications simply submit queries over the network. While this provides some security, digital signatures could be used to secure driver code and other techniques are likely necessary to ensure bad actors do not abuse the underlying devices. The operating system we envision would provide a real-time scheduler to execute the IoT tasks as well as the concurrency and other access control artifacts necessary to ensure efficiency and program correctness of the calling applications, some of which we explore in Section 4. Our own prototype system provides such features, and enables several optimizations we detail in later sections. We utilize our system to provide initial results demonstrating the usefulness of such a semantic runtime.

2.2 The Device Set

As a key innovation in such a system, we introduce the concept of a device set (DevSet) as a first-class abstraction. A DevSet is a set of IoT devices that for the purposes of an application can be considered functionally equivalent in terms of their semantic properties and can therefore be used interchangeably by a semantic runtime system. For example, if a user is interested in increasing the temperature in a room, the system may choose between heat sources such as a furnace, windows, a space heater, etc. based on competing interests from other applications or based on some internal system priority, like energy efficiency. DevSets can also provide fault-tolerance by enabling redundant devices to be utilized in the event of failures. The criteria to create a DevSet are specified by declarative queries submitted by each application. Such sets may overlap, notifying our system of potential synergies or conflicts. We go over other examples of DevSet usages in Sections 3 and 4. The ability of our system to choose a device from functionally equivalent sets means that optimizations for energy use, accuracy, latency, etc. can all be made for the system as a whole, while being transparent to the end-user.

2.3 Semantic Relations

Many IoT environments are equipped with sensors for measuring certain environmental factors such as luminosity, humidity, temperature, etc. as well as actuators that can potentially alter those factors like lights, window blinds, humidifiers, and thermostats. We propose the notion of semantic relations between high-level semantic goals and low-level sensing and actuation tasks. These semantic relations can be derived through the

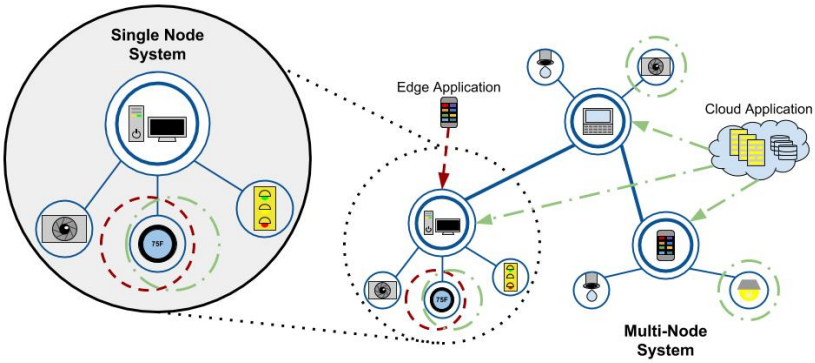

Figure 1: Proposed Architecture

specification of rules, or discovered via machine learning. For example, a user may specify a desired level of lighting and temperature in a room, which could be accomplished by a combination of different actuations that the user wouldn't need to explicitly specify.

3 Enabling Synergies Across Applications

Many IoT sensing systems today focus on blindly polling data from sensors on the edge and storing them in large cloud backends. We advocate that by having an operating system like the one described in the previous section we can create mechanisms which identify and exploit common sensor streams between multiple applications to increase overall efficiencies in latency and energy usage. Such mechanisms would involve reducing the latency of obtaining sensor readings or reducing the number of readings in order to conserve energy, an important metric for keeping costs down as well as in terms of current environmental concerns. To this end, we give example optimization such as tolerance-based caching, period synchronization, and device substitution, all made possible by having an edge-based semantic OS. We chose these particular optimizations as we believe they are likely to be beneficial to many IoT sensing applications. They are not exhaustive and additional optimizations are left to further consideration.

3.1 Tolerance-based Caching

As in a traditional systems context, caching a sensor reading makes it available with greater immediacy to applications. In the IoT, it would also reduce energy consumption by reusing the same reading. This isn't limited to simple sensor readings like temperature or humidity, video streams can be buffered or recent triggers can be held to increase the availability and efficiency of all these data sources. Figure 2 shows a set of latencies derived through experiments performed using our proof-of-concept system, Constellation. We setup a Particle WiFi-enabled microcontroller with a temperature sensor to act as an IoT device, and connected this to a single Raspberry Pi edge node running our Constellation prototype, which would periodically cache sensor readings based on perceived volatility in the data stream. Latencies shown include those experienced with and without the use of our semantic edge node (those without are denoted as edge to sensor and sensor to cloud). One may notice that the latency experienced with a cache hit is actually less than the direct to sensor approach. This is because the edge node hosting the application and the node running our prototype communicated over 5 GHz WiFi (and Ethernet for the Pi) as opposed to the slower 2.4GHz WiFi the microconctroller is capable of.

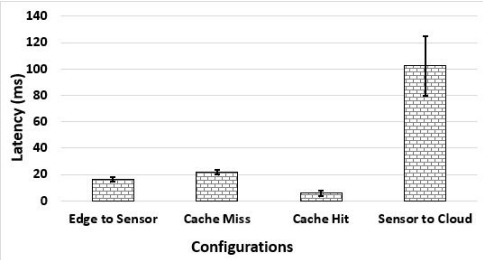

Figure 2: Average (N>1000) Round-Trip Latencies

These readings are read-only, meaning that coherency is not an obstacle, but freshness or relevance of the data is. This introduces the research question of determining when to refresh the cache. While applications utilizing systems like Amazon's AWS IoT [1] often utilize a constant periodic timer to refresh the values in their virtual or 'shadow' devices, we advocate that a semantic operating system would be able to take into account things like application tolerances as well as trends in the data streams to determine dynamic refresh intervals.

3.2 Period Synchronization

Period synchronization is the idea of matching the start times of various periodic tasks to allow for the same reading to be shared by many applications. Unlike caching, speed and energy efficiency are still improved without sacrificing freshness or accuracy. Figure 3 below shows the energy savings from

synchronizing five periodic queries, each with periods of 12, 15, 20, 30 and 60 minutes, respectively.

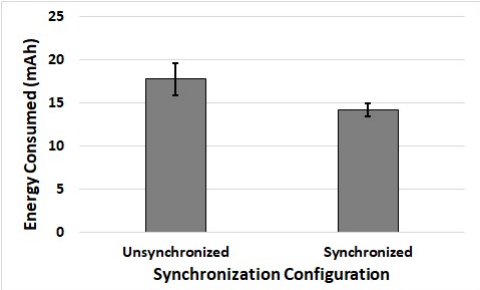

Figure 3: Average (N=20) Energy Savings for Synced Queries

The energy we measure is that of the same IoT node used in the caching example. We see period synchronization as being a useful optimization because as the number of concurrent applications increases so will the number of synchronizations points, thereby scaling the reduction in energy use. An OS that maintains IoT tasks on the edge near their desired resources would be able to effectively perform such an optimization.

3.3 Device Substitution

Device substitution is the idea that multiple devices are potentially capable of satisfying a particular task. Returning to the example of police pursuing a suspect on foot, said officers could glean knowledge of the suspect's location and escape route from many sensors (cameras, motion detectors, etc.) in the area. Choosing which devices to satisfy the task could be based on the metrics of interest. In our system, such related devices would be represented as DevSets, as described in Section 2, and the target device(s) could be chosen based on several strategies. For one, the system may desire to reduce latency and energy usage by utilizing devices whose caches have been recently refreshed. Similarly, the system may choose to distribute the load of requests across the many sensors in an effort to distribute energy usage. All of these decisions would benefit immensely from having a real-time localized edge-based system that has access to the necessary context and compute resources to perform such optimizations.

4 Handling Conflicts

While some application requests may be constructive in a large-scale IoT environment, others, particularly those involving actuators, are likely to conflict. This is evident in our previous example of several overlapping emergencies, where there exists a problem of how to regulate access to local displays to notify passing pedestrians to stay away from a particular location. Sensing involves observing the state of an area, while actuations involve changing it. Often, these changes are only relevant if the state is maintained for a period of time. This could involve controlling the lighting in a room, or a dishwasher in a smart home. Other conflicts can arise because even if an actuator can complete a task quickly, like a coffee dispenser, this could still cause conflicts if the coffee becomes depleted. Because all actuators involve changing state, most actuations likely have the potential to encounter conflict. One exception could be the act of sending a text message, as this is essentially writing to an append-only log.

4.1 Conflict Resolution Techniques

For the sake of ensuring a seamless, automated solution, there may be several ways to resolve the conflict. One way could be to augment traditional concurrency mechanisms like a lock or semaphore through the use of a quantum to divide the use of the actuator into certain time intervals. Multiple applications could each possess the lock, while the access time is divided between them. The system could calculate the duration to give each task control of an actuator based on many factors such as priority or a minimum duration specified by each task.

4.2 Dynamic Prioritization

While the quantum-based lock solution may work for our display actuator, others may not be so easily resolved. For example, our previously stated conflict between police in pursuit of a criminal and a train wishing to pass cannot be resolved by simply alternating between who has control of the traffic lights. Other factors such as the physics involved in stopping a large train could be factored into a dynamic calculation of priorities for the two competing applications.

4.3 Utilizing Alternatives

Another form of conflict may arise when two applications seek to alter an environment in different ways by using the same actuator. For example, one application may wish to increase the amount of light in an office building by opening the blinds on the windows. While another may be seeking to regulate the temperature in the building by keeping the blinds closed and avoiding the extra heat. A semantically aware operating system would be able to find an alternative solution by noticing it is cooler outside and thus opening a window to cool the interior, thereby allowing the blinds to be open, satisfying both applications. This could be realized with the idea of environmental semantics presented in Section 2. Assuming programmers had some means of specifying the relationship between temperature and these actuations, either through declarative rules or machine learning, a semantic operating system could assuage such a conflict. All of these conflicts necessitate the use of a context-aware local system which can not only provide computation but immediate enforcement of control mechanisms.

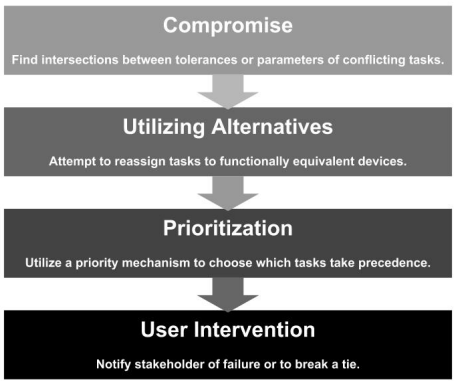

Figure 4: Proposed Policy for Automated Conflict Resolution

4.4 Automated Conflict Resolution

While we recognize the diversity of application scenarios, we advocate the following general priority, illustrated in Figure 4, when it comes to resolving conflicts. This is driven by the goal of an IoT system being to ensure automated responses while at the same time recognizing that some conflicts involve such critical safety and semantic intricacies as to require rigid priorities or even human intervention. We propose this priority list, seen in Figure 4, after considering the list of conflict resolution ideas advocated by Nakamura et al in [19].

As a first step, we advocate that applications should specify tolerances for the freshness or accuracy of sensor readings as well as for certain actuation parameters, like the brightness of a light bulb, so that the system may find an intersection between these tolerances to determine a solution. If this is not possible, we would advocate our ideas of DevSets and semantic relations, described in Section 2, be utilized to potentially find an alternate method for some of the conflicting applications. This again would allow for multiple applications to be potentially satisfied without user intervention. The final possibility before prompting a user would be to utilize some form of priority, either static or dynamic, to allow at least one application to be satisfied in the short term. Otherwise, a user will need to be contacted to resolve scenarios which require excessive context not easily captured through logic or rules. Another factor which could influence such a semantic operating system, especially conflict resolution, are political or legal concerns about jurisdiction, ethics, as well as ownership of data and physical resources. This is left to future work.

5 Related Work

Edge computing has been advocated as a pairing with the IoT field many times previously, most notably in [9], but also in such works as [25] where the ineffectiveness of the cloud is elaborated on, and in [12, 14, 21] which all advocate or use the edge to support IoT applications. The idea of an edge-based intermediary for IoT devices and applications has now become commercial, with industry players such as Apple [3], Google [4], Amazon [1], and Microsoft [5] all developing concepts involving hubs in the local environment that provide computation while offloading larger jobs to the cloud. These artifacts are often highly limited in their functionality with little semantic appreciation for concurrent applications.

The concept of period synchronization has been used in other contexts such as in sensor network protocols utilizing constructive interference (CI) [11, 13, 22]. These protocols involve duty cycling devices simultaneously so their transmissions can interfere constructively, increase their range while reducing energy usage on a per-device basis. It's worth noting that our work involves the synchronization of periods of disparate lengths. The abstraction of sensory resources has also existed in the sensor network field for many years, particularly in networks utilizing tinyDB's database abstraction [17].

The issue of resolving conflicts in an environment through the use of semantic context has been studied in such works as [18, 19, 23]. Several declarative methods have been developed for specifying IoT relationships and environmental semantics, such as in Brick [7] and MUSIC [20]. We envision such schemas to be useful in extending our prototype system to exploit semantics in ways outlined in Section 4.

The application of smart cities has been used previously [15, 24]. The design and implementation of an example smart city project located in Padova, Italy is given in [24]. While our work uses the smart city for many of its motivational examples, we envision such a semantic system to be useful even in home environments where IoT operating systems have been proposed previously, as in [10].

6 Conclusion

In this work we advocate that the use of the edge for supporting IoT applications should go beyond simply offloading computation to a less distant sever. The edge, through its computational capacities and location as an intermediary has the power to provide semantic optimizations to improve the efficiency with which IoT applications execute in what will in the future be highly instrumented environments. In particular, we showed the opportunities and challenges that arise from such a highly instrumented environment with many cooperating applications. We provided concrete examples and preliminary results using a prototype to demonstrate the efficacy of our opinions and outlined numerous avenues of potential future work to help bring such semantic-aware systems to fruition.

Acknowledgments

The authors would like to thank Particle, an IoT startup, which provided much of the hardware resources used in the experiments of this paper. We would also like to recognize the NSF which provided support under grant CNS-1619254.

Discussion Topics

Desired Feedback The feedback we are seeking to elicit from the reviewers is on the need for, as well as the practical difficulties of bringing such a semantic runtime system to the edge, capable of communicating with so many devices with so many potentially disparate protocols.

Controversial Points

- Our work assumes the ubiquity of sensors, computation, and some amount of standards and homogeneity in terms of how these things communicate. One could argue there are currently many financial incentives by industry players to market their own siloed and holistic solutions.
- Our system would introduce the issues of privacy, due to the potentially intrusive nature of cameras, motion detectors, and other forms of sensing, as well as security for the compute nodes responsible for regulating access to IoT devices in a particular area.

Discussion Points

- Alternative uses for the edge as more than just a local computation source.
- The topic of conflict resolution by semantically oriented concurrency controls, like those described herein.
- Such an environment introduces issues of ownership, both of the sensors/actuators and the data produced from public places. And, if the edge nodes are volunteers, how is their trustworthiness assessed? How are users incentivized to volunteer their compute resources?

Open Issues A major open issue of our work involves the instantiation of our proposed system as a P2P edge network. To provide access to data over a wide area, and to enable applications hosted in locales other than that of the sensors or actuators being used, it is necessary to scale our system to a full-fledged P2P network. This will involve exploring and overcoming several challenges such as discovery and authentication of both edge compute resources as well as IoT devices which may have their own protocols and communication mediums.

Depreciating Circumstances *Moore's Law:* The need for a middleware abstraction as described to efficiently utilize IoT resources could be rendered obsolete if significant improvements were to be made such that devices could actively maintain many open connections, provide their own robust synchronization features, as well as analyze and recognize trends in their own data streams to perform cross-application optimizations.

Next-Gen Networking: Significant improvements in wide-area networking could make the enforcement of real-time application constraints from the cloud more feasible. This would render the need for much of the edge obsolete.

Heterogeneity Concerns: A lack of standards adoption in industry is a potential pain point in imagining a seamless IoT environment for interfacing with thousands of applications.

References

[1] Amazon web services. https://aws.amazon.com/iot, 2018.

[2] Apache edgent. http://edgent.apache.org/, 2018.

[3] Apple homekit. https://developer.apple.com/homekit/, 2018.

[4] Google iot solutions. https://developers.google.com/iot/, 2018.

[5] http://www.lab-of-things.com/, 2018.

[6] Brian Amento, Bharath Balasubramanian, Robert J Hall, Kaustubh Joshi, Gueyoung Jung, and K Hal Purdy. Focusstack: Orchestrating edge clouds using location-based focus of attention. In *Edge Computing (SEC), IEEE/ACM Symposium on*, pages 179–191. IEEE, 2016.

[7] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. Brick: Towards a unified metadata schema for buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, pages 41–50. ACM, 2016.

[8] Ketan Bhardwaj, Sreenidhy Sreepathy, Ada Gavrilovska, and Karsten Schwan. Ecc: Edge cloud composites. In *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on*, pages 38–47. IEEE, 2014.

[9] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[10] Colin Dixon, Ratul Mahajan, Sharad Agarwal, AJ Brush, Bongshin Lee, Stefan Saroiu, and Paramvir Bahl. An operating system for the home. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 25–25. USENIX Association, 2012.

[11] Manjunath Doddavenkatappa, Mun Choon Chan, Ben Leong, et al. Splash: Fast data dissemination with constructive interference in wireless sensor networks. In *NSDI*, pages 269–282, 2013.

[12] Andy Rosales Elias, Nevena Golubovic, Chandra Krintz, and Rich Wolski. Where's the bear?-automating wildlife image processing using iot and edge cloud systems. In *Internet-of-Things Design and Implementation (IoTDI), 2017 IEEE/ACM Second International Conference on*, pages 247–258. IEEE, 2017.

[13] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. Efficient network flooding and time synchronization with glossy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 73–84. IEEE, 2011.

[14] Sehyeon Heo, Sungpil Woo, Janggwan Im, and Daeyoung Kim. Iot-map: Iot mashup application platform for the flexible iot ecosystem. In *Internet of Things (IOT), 2015 5th International Conference on the*, pages 163–170. IEEE, 2015.

[15] Dimosthenis Kyriazis, Theodora Varvarigou, Daniel White, Andrea Rossi, and Joshua Cooper. Sustainable smart city iot applications: Heat and electricity management & eco-conscious cruise control for public transportation. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pages 1–5. IEEE, 2013.

[16] Zach Leidall, Abhishek Chandra, and Jon Weissman. Creating an abstraction to exploit iot synergies. In *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 347–348. IEEE, 2017.

[17] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on database systems (TODS)*, 30(1):122–173, 2005.

[18] Verena Majuntke, Sebastian VanSyckel, Dominik Schäfer, Christian Krupitzer, Gregor Schiele, and Christian Becker. Comity: Coordinated application adaptation in multi-platform pervasive systems. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 11–19. IEEE, 2013.

[19] Masahide Nakamura, Hiroshi Igaki, and Ken-ichi Matsumoto. Feature interactions in integrated services of networked home appliances. In *Proc. of Int'l. Conf. on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI'05)*, pages 236–251, 2005.

[20] Roland Reichle, Michael Wagner, Mohammad Ullah Khan, Kurt Geihs, Massimo Valla, Cristina Fra, Nearchos Paspallis, and George A Papadopoulos. A context query language for pervasive computing environments. In *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 434–440. IEEE, 2008.

[21] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta N Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. Farmbeats: An iot platform for data-driven agriculture. In *NSDI*, pages 515–529, 2017.

[22] Yin Wang, Yuan He, Xufei Mao, Yunhao Liu, and Xiangyang Li. Exploiting constructive interference for scalable flooding in wireless networks. *IEEE/ACM Transactions on Networking*, 21(6):1880–1889, 2013.

[23] Miki Yagita, Fuyuki Ishikawa, and Shinichi Honiden. An application conflict detection and resolution system for smart homes. In *Proceedings of the First International Workshop on Software Engineering for Smart Cyber-Physical Systems*, pages 33–39. IEEE Press, 2015.

[24] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[25] Ben Zhang, Nitesh Mor, John Kolb, Douglas S Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward A Lee, and John Kubiatowicz. The cloud is not enough: Saving iot from the cloud. In *HotCloud*, 2015.

[26] Irene Zhang, Adriana Szekeres, Dana Van Aken, Isaac Ackerman, Steven D Gribble, Arvind Krishnamurthy, and Henry M Levy. Customizable and extensible deployment for mobile/cloud applications. In *OSDI*, volume 14, pages 97–112, 2014.