AN ARCHITECTURE FOR SELF-AWARE IOT APPLICATIONS

Lukas Esterle[*]

Aston Lab for
Intelligent Collective Engineering
Aston University
Birmingham, UK

Bernhard Rinner

Institute of Networked and
Embedded Systems
Alpen-Adria-Universität Klagenfurt
Austria

ABSTRACT

Future Internet of Things (IoT) applications will face challenges in increased flexibility, uncertainty, dynamics and scalability. Self-aware computing maintains knowledge about the applications state and environment and then uses this knowledge to reason about and adapt behaviours. In this position paper, we introduce self-aware computing as design approach for IoT applications which is centred around a self-aware architecture for IoT nodes. This architecture particularly supports adaptations based on node interactions. We demonstrate our approach with an IoT case study on multi-object coverage with mobile cameras.

Index Terms— self-aware computing; architecture; internet-of-things; node interactions; mobile cameras

1. INTRODUCTION

Over the last years, sensing, processing and networking capabilities have pervaded into many everyday devices enabling the Internet of Things (IoT). Most IoT devices analyse sensed data locally and extract relevant information, collaborate with other devices, and provide the user with descriptions of captured events. IoT applications are ubiquitous including security, automation, entertainment or smart homes. Current trends show that future IoT applications will scale up in the number of nodes and be required to offer more complex functionalities, a much higher degree of flexibility, and an increased autonomy [1]. A traditional design approaches for IoT devices with fixed functionalities and static configurations will hardly meet these challenges.

Self-aware computing describes a novel paradigm for systems and applications that pro-actively maintain knowledge about their internal state and its environment and then use this knowledge to reason about behaviours [2, 3]. This paradigm clearly distinguishes between self-aware (SA) capabilities, which maintain knowledge, and self-expressive (SE) capabilities, which adapt the system's behaviour based on this knowledge accordingly. Self-aware computing can leverage IoT systems with advanced levels of autonomous behaviour to enable runtime self-adaptation and management of complex trade-offs in rapidly changing conditions. In this paper, we propose self-aware computing as a design approach for IoT applications by adapting the generic reference architecture [2]. We demonstrate our approach with an IoT case study on multi-object coverage with mobile cameras.

In recent years, self-aware computing has received a lot of attention in embedded and multi-core systems [4, 5, 6, 7] but also other application areas such as surveillance and security [8, 9, 10, 11], IoT [12], cloud and data centres [13, 14, 15], automotive systems [16], as well as robotic and space applications [17, 18] have been investigated. In the vast majority, SA has mainly focussed on individual nodes and their local objectives rather than the collaborating collective and their common, network-wide goals. In this paper we explicitly consider the interaction with others and the models generated to improve the performance towards achieving global, network-wide goals.

The remainder of this paper is structured as follows. Section 2 presents our novel architecture for self-aware IoT devices with a discussion on the difference between node-level and network-level SA. Section 3 introduces our case study and presents simulation results demonstrating performance improvements by node- and network-level self-awareness. Section 4 concludes the paper with a short discussion on future work.

2. SELF-AWARE ARCHITECTURE

While Lewis et al. [2] and Kounev et al. [3] consider self-awareness as an attribute of a single device, we propose a separation of concerns for self-awareness on individual devices. In our architecture, illustrated in Figure 1, we use core elements of the proposed SA reference architecture [2], but distinguish explicitly between node- and network-level aspects of SA and SE. This means, each block still uses different levels of self-awareness (i.e. stimulus-, interaction-, time-, goal-, and meta-self-aware) in each block but is able to focus on dif-

[*]This work is supported by the SOLOMON project (Self-Organisation and Learning Online in Mobile Observation Networks) funded by the European Union H2020 Programme, grant n°705020.

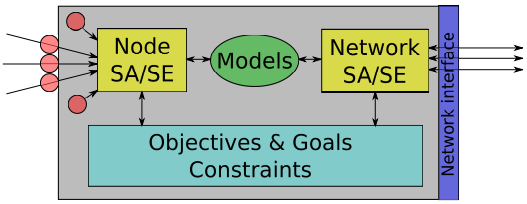


Fig. 1. Illustration of a single IoT device and two separated self-awareness/self-expression components. Sensors, illustrated as red circles, can be internal and external and concern the *Node SA/SE*. Interactions with other devices is relised via the *Network interface* and concerns the *Network SA/SE*.

ferent requirements on the node- and the network level while maintaining and utilising a common model base.

In IoT applications, nodes have typically input from multiple sensors which may observe a variety of aspects in the environment (e.g. using accelerometers, visual, temperature, magnetic, or wind and lighting sensors) as well as internal properties of the device (e.g. battery levels or available memory). These sensors feed into the node-level SA/SE. On the other hand, the device also interacts with the other devices in the environment in order to receive information and to share own data. Both SA/SE blocks develop and refine models on their observations. These models are shared between both SA/SE blocks but might be utilised in a completely different way. In addition, the decisions and behaviours of the individual SA/SE blocks are driven by the objectives and goals of the device as well as local constraints. Goals as well as constraints may change during runtime. Techniques to achieve SA/SE often rely on complex and resource-intensive machine learning approaches, in contrast, we rely on local data collection and exchange of this information as basis for our SA/SE.

2.1. Node self-awareness

Self-awareness on the node-level is concerned with all aspects that are directly related to the individual device and its immediate environment. This can range from selecting different algorithms based on changing environmental conditions (e.g. lighting) or changing goals and constraints (e.g. low battery) to explore and apply different behavioural approaches to overcome rapidly unfolding situations (e.g. increase/reduce resource utilisation).

2.2. Network self-awareness

When an IoT device attempts to interact with others in order to overcome common challenges, being aware of the state, capabilities and goals of other devices in the environment allows each node to make profound decisions about the others intended actions. Only by having this knowledge, the best possible outcome can be achieved. Network-level self-awareness is handling this while being decoupled from the node-level self-awareness in an effort to separate concerns—focussing on information from other devices in the network rather than locally determined knowledge.

Network-level SA collects information from other devices in the network and generates knowledge about their behaviour, location, and capabilities, allowing for more efficient collaboration. The network-level SE interacts with them in accordance with given local objectives and constraints, i.e. requesting support from others but also helping them when required.

3. MULTI-OBJECT k-COVERAGE

To study and evaluate our approach of node and network SA/SE, we apply our proposed architecture to visual mobile sensor networks and the problem of covering moving objects with a certain number of cameras. This is just one application where multiple IoT devices try to achieve common goals. Alternative applications are in the area of smart production systems, smart transportation or elderly care. The multi-object k-coverage problem has been introduced by Esterle and Lewis [19] and is related to the well-studied problem of Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) [20]. Here, a set of cameras $C = \{c_1, c_2, \ldots, c_n\}$ has to cover each moving *object of interest* $O = \{o_1, o_2, \ldots, o_m\}$ with at least k cameras where the object's interest can change during runtime:

$$imp(o_j, t) = \begin{cases} 1, & \text{if } o_j \text{ is of interest} \\ 0, & \text{otherwise.} \end{cases}$$

In their initial work, they showed how coordination in general allows to improve the tracking of objects with k cameras over non-coordinated approaches. In order to keep an object within its field of view (FOV) a camera is able to relocate their position $\mathbf{x}_i = (x_i, y_i)$ within the environment with a maximum velocity ($\vec{s_i}$) and change their orientation ω_i with a maximum angular velocity u_i. An object o_j is considered covered by a camera c_i at time t, if the object is within the FOV of the camera, i.e.

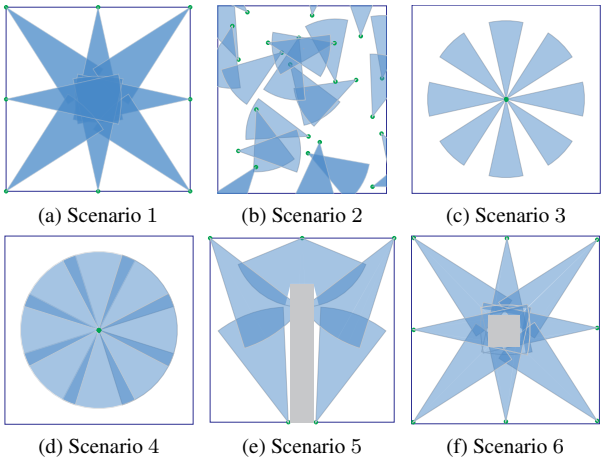
$$cov(o_j, c_i, t) = \begin{cases} 1, & \text{if } o_j \text{ within FOV of } c_i \\ 0, & \text{otherwise.} \end{cases}$$

An object o_j is k-covered by the n cameras if

$$kcov(o_a, k, t) = \begin{cases} 1, & \textit{if } \sum_{i=1}^{n} cov(o_j, c_i, t) \geq k \\ 0, & \textit{otherwise.} \end{cases}$$

Finally, we are interested in increasing the times where as many objects as possible are being covered by at least k cameras in parallel. For a finite time horizon T this can be formulated as

$$performance = \frac{\sum_{t=1}^{T} \sum_{j=1}^{m} kcov(o_j, k, t)}{\sum_{t=1}^{T} \sum_{j=1}^{m} imp(o_j, t)}. \quad (1)$$



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

(d) Scenario 4 (e) Scenario 5 (f) Scenario 6

Fig. 2. Evaluated scenarios. Green dots represent cameras and blue cones their respective FoVs. Grey blocks illustrate opaque walls/areas.

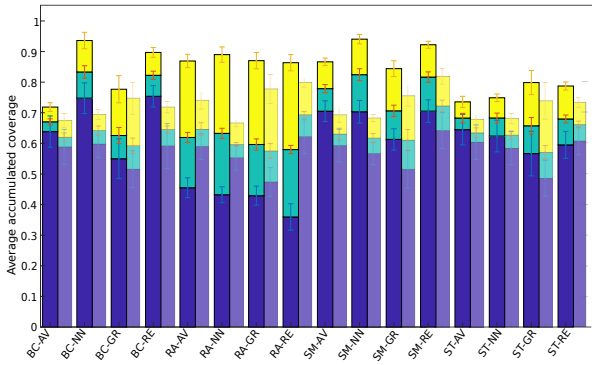


Fig. 3. Comparison of normalised accumulated coverage over all objects and the entire simulation (cp. Equation 1) for the Scenario 2 using different communication strategies (BC, RA, SM, and ST) and different response models (AV, NN, GR, and RE). The mean and standard deviation of k-coverage (blue), 2-coverage (turquoise) and 1-coverage (yellow) is shown. Results from the corresponding approaches without network-level self-awareness are shown in lightly shaded colours.

In order to maximise the coverage of each object with at least k cameras, cameras observing an object of interest (i.e. having the object within their FOV), will request other cameras to help observing it. Each camera can determine by itself whether an object is of interest or not (e.g. by detecting suspicious behaviour). In addition, cameras will follow the object through the environment in order to prolong the observation. However, while a camera can observe multiple objects at once, it can only select a single object to follow. The individual camera decides which object to provision.

In contrast to this prior work, we induce our nodes with network-level self-awareness, enabling them to reason about the behaviour of other nodes and their location. In addition, we introduce node-level self-awareness allowing the node itself make a decision whether to request other cameras to support provisioning a specific object or not.

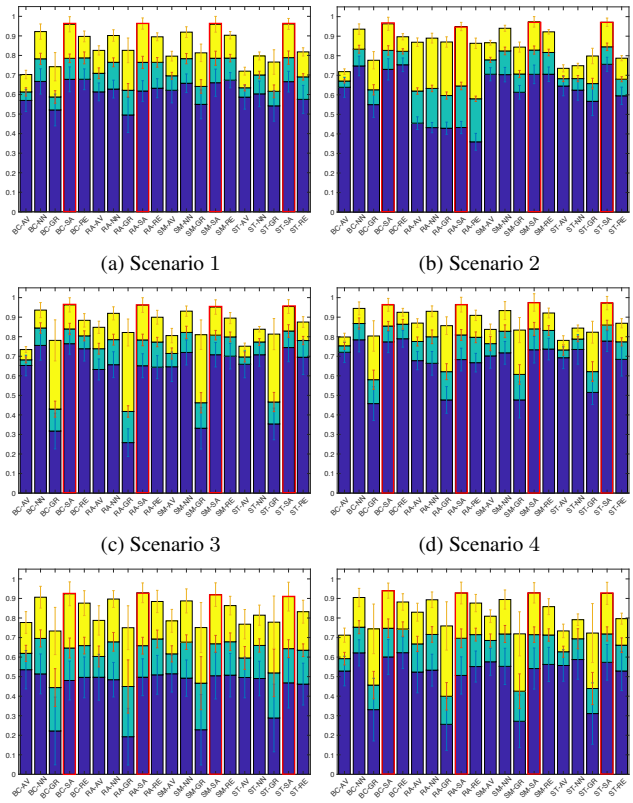
3.1. Node and network self-awareness

Specifically in our case study, node-level self-awareness covers the following aspects. First, each node can make an individual decision whether to follow an object of interest or not based on its current state of engagement and environmental conditions (e.g. visibility of object). Second, the decision whether to request help from others is based on local information alone. However, this information could also be based on knowledge generated by the network-level SA/SE block.

Network-level self-awareness keeps track of the current engagement of the other cameras. This can be achieved implicitly through observing their behaviour and communication or explicitly based on direct information exchange. Having knowledge about the state of other cameras allows each individual camera to reason about the potential response to received help requests. In our case, each camera keeps track of the cameras it received requests from. Furthermore, each requesting camera includes information about the object it is primarily provisioning. This information is kept for a limited time window only, ensuring to not work with outdated information. This allows each camera to keep track on one hand of cameras currently provisioning objects and on the other hand of the number of camera provisioning each known object.

For our experimental evaluation we utilised the camera network simulation tool CamSim [21]. This allowed us to directly compare our novel approach on the multi-object k-coverage problem and the previous work. Objects and cameras follow a straight trajectory but bounce back randomly when reaching the boundary of the environment. This keeps the number of objects and cameras constant. Each experiment has been repeated 30 times. Figure 2 gives an overview of the scenarios [19]. A green dot is a camera and blue circular segments the corresponding FOV, grey blocks are opaque walls.

In our evaluation, we employ some of the communication strategies and response models proposed by Esterle and Lewis [19], namely *Broadcast (BC)*, where each camera communicates with every other camera in the network; *Random (RA)* communication, where each camera communicates with a $k-1$ random other cameras; and *Step (ST)* and *Smooth (SM)* communication, where cameras learn their neighbourhood online and communicate with the k closest neighbours. A *response model* decides whether to respond to a help request or not. We use the *Newest-Nearest (NN)*, a camera follows the latest object that is closest; *Available (AV)*, just as with NN but only if the camera is not currently following another object; *Graph (GR)*, the decision is based on the learnt neighbourhood relations; *Received calls (RE)*, a camera follows the object with the least requests. All experiments where conducted over $T = 1000$ time steps with the goal of covering all objects with at least $k = 3$ cameras. Each scenario has



(a) Scenario 1 (b) Scenario 2

(c) Scenario 3 (d) Scenario 4

(e) Scenario 5 (f) Scenario 6

Fig. 4. Performance of our self-aware response model (SA) in comparison to non-self-aware response models. Performance for $k = 1$ is illustrated in yellow, for $k = 2$ in turquoise, and for $k \geq 3$ in blue.

between 8 and 19 objects to be followed.

Figure 3 shows the average percentage of covered objects accumulated over all objects and the entire simulation time as given in Equation 1. This network-wide performance of covering important objects is shown exemplary for Scenario 2 with and without network-level SA. In direct comparison we can see the benefits of introducing network-level self-awareness where cameras (i) know the available cameras in the network and only contact those, and (ii) cameras only request help until k cameras agree to provision the object. It becomes apparent that network-level SA can improve k-coverage by up to 15%. However, this is not achieved for approaches using the *RA* communication strategy. We speculate this is due to the fact that the Network SA reduces the number of potential communication partner drastically. Interestingly, our networks achieve much better 1- and 2-coverage when we employ network-level SA with performance increase of up to 25%.

3.2. Self-aware response model

We also introduce an new *Self-aware* response model in Algorithm 1. The *Self-aware* response model is based on the *Available* response, were cameras provision the object that is within their FOV or has been requested help for and is closest to their own location. However, if the camera observes multiple object at the same time, the Network SA is employed to make local decisions. If more than k cameras provision the object the observing cameras will iteratively switch to the other object with the lowest number of cameras currently provisioning. This is continued until the number of cameras for each object equals k over time. Again, we rely on the information about primarily provisioned objects in help requests from other cameras. Furthermore, if a camera observes multiple objects and the currently provisioned object is covered by at least one more camera, it will switch to another object that is not provisioned by any other camera.

Algorithm 1: Self-aware response model

```
Perform for each camera c_i at each time step t:
foreach o_a ∈ O & cov(o_a, c_i, t) do
    if kcov(o_a, j, t), j > k then
        Provision object o_b where kcov(o_b, l, t), l < k
    else
        if (kcov(o_a, l, t), l > 1 & cov(o_b, c_i, t) &
          kcov(o_b, l, t), l == 0) then
            Provision o_b
        end
    end
end
```

While we expected that this will reduce the network-wide k-coverage, Figure 4 indicates that this reduces performance only in rare cases for k-coverage and never more than 3% of a non-self-aware response model. However, we can observe an increase in network-wide 1-coverage throughout all direct comparisons. The performance of the self-aware response model is highlighted in red.

4. CONCLUSION

We presented a new architecture for self-aware IoT applications. The architecture explicitly considers the difference between knowledge determined on the node and information received from other devices. We presented the benefits of this approach in a case study featuring autonomous smart camera systems tasked to achieve a common goal. Furthermore, we introduced a new self-aware approach for this problem, making decisions based on both, node and network self-awareness. This allowed the network to increase the network-wide performance even further and shows the benefits and applicability of our novel architecture. In future work we will study the impact of the amount of exchanged information among the cameras for network-level self-awareness. Furthermore, we will investigate the impact of more sophisticated learning techniques on goals that can only be achieved by collaborating devices in the network.

5. REFERENCES

[1] L. Esterle and R. Grosu, "Cyber-physical systems: challenge of the 21st century," *e & i Elektrotechnik und Informationstechnik*, vol. 133, no. 7, pp. 299–303, Nov 2016.

[2] P. R. Lewis, M. Platzner, B. Rinner, J. Torresen, and X. Yao, Eds., *Self-aware Computing Systems: An Engineering Approach*. Springer, 2016.

[3] S. Kounev, P. Lewis, K. Bellman, N. Bencomo, J. Camara, A. Diaconescu, L. Esterle, K. Geihs, H. Giese, S. Götz, P. Inverardi, J. Kephart, and A. Zisman, "The notion of self-aware computing," in *Self-Aware Computing Systems*, S. Kounev, J. O. Kephart, A. Milenkoski, and X. Zhu, Eds., 2017, pp. 3–16.

[4] N. Dutt, A. Jantsch, and S. Sarma, "Toward smart embedded systems: A self-aware System-on-Chip (SoC) perspective," *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 2, pp. 22:1–22:27, Feb. 2016. [Online]. Available: http://doi.acm.org/10.1145/2872936

[5] H. Hoffmann, J. Holt, G. Kurian, E. Lau, M. Maggio, J. E. Miller, S. M. Neuman, M. Sinangil, Y. Sinangil, A. Agarwal, A. P. Chandrakasan, and S. Devadas, "Self-aware computing in the Angstrom processor," in *Proc. Design Automation Conference*, June 2012, pp. 259–264.

[6] A. Agne, M. Happe, A. Lösch, C. Plessl, and M. Platzner, "Self-awareness as a model for designing and operating heterogeneous multicores," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 2, pp. 13:1–13:18, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2617596

[7] M. Platzner, "On-the-fly computing: Self-aware heterogeneous multi-cores," in *Proc. Int. Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2016, pp. 1–2.

[8] Z. Guettatfi, P. Hübner, M. Platzner, and B. Rinner, "Computational self-awareness as design approach for visual sensor nodes," in *Proc. Int. Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, July 2017, pp. 1–8.

[9] L. Esterle, J. Simonjan, G. Nebehay, R. Pflugfelder, G. F. Domínguez, and B. Rinner, "Self-aware object tracking in multi-camera networks," in *Self-aware Computing Systems*. Springer, 2016, pp. 261–277.

[10] B. Rinner, L. Esterle, J. Simonjan, G. Nebehay, R. Pflugfelder, G. F. Domnguez, and P. R. Lewis, "Self-aware and self-expressive camera networks," *Computer*, vol. 48, no. 7, pp. 21–28, July 2015.

[11] K. Dabčević, L. Marcenaro, and C. Regazzoni, "Security in cognitive radio networks," *Evolution of Cognitive Networks and Self-Adaptive Communication Systems*, pp. 301–335, 2013.

[12] M. Möstl, J. Schlatow, R. Ernst, H. Hoffmann, A. Merchant, and A. Shraer, "Self-aware systems for the internet-of-things," in *Proc. Int. Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2016, pp. 1–9.

[13] S. Spinner, "Self-aware resource management in virtualized data centers," Ph.D. dissertation, Julius Maximilians University Würzburg, Germany, 2017. [Online]. Available: https://opus.bibliothek.uni-wuerzburg.de/frontdoor/index/index/docId/15375

[14] M. Salama, A. Shawish, and R. Bahsoon, "Dynamic modelling of tactics impact on the stability of self-aware cloud architectures," in *Proc. Int. Conf. on Cloud Computing (CLOUD)*, June 2016, pp. 871–875.

[15] A. Iosup, X. Zhu, A. Merchant, E. Kalyvianaki, M. Maggio, S. Spinner, T. Abdelzaher, O. Mengshoel, and S. Bouchenak, *Self-awareness of Cloud Applications*. Cham: Springer International Publishing, 2017, pp. 575–610.

[16] J. Schlatow, M. Moostl, R. Ernst, M. Nolte, I. Jatzkowski, M. Maurer, C. Herber, and A. Herkersdorf, "Self-awareness in autonomous automotive systems," in *Proc. Design, Automation Test in Europe Conference Exhibition*, March 2017, pp. 1050–1055.

[17] T. N. Mundhenk, J. Everist, C. Landauer, L. Itti, and K. Bellman, "Distributed biologically based real time tracking in the absence of prior target information," in *Proc. of SPIE on Intelligent Robots and Computer Vision*, vol. 6006, 2005, p. 12.

[18] C. Landauer and K. L. Bellman, "An architecture for self-awareness experiments," in *Proc. Int. Conf. on Autonomic Computing*, July 2017, pp. 255–262.

[19] L. Esterle and P. R. Lewis, "Online multi-object k-coverage with mobile smart cameras," in *Proc. Int. Conf. on Distributed Smart Cameras*. ACM, 2017, pp. 1–6.

[20] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: A review," *IEEE Transactions on Cybernetics*, pp. 1–12, 2017, in print.

[21] L. Esterle, P. R. Lewis, H. Caine, X. Yao, and B. Rinner, "Camsim: A distributed smart camera network simulator," in *Proc. Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshops*, Sept 2013, pp. 19–20.