Integrating Constraint-based Planning with LwM2M for IoT Network Scheduling

Uwe Köckemann[1], **Nicolas Tsiftes**[2], **and Amy Loutfi**[1]
[1] Center for Applied Autonomous Sensor Systems, Örebro University, Sweden
[2] RISE SICS, Sweden

Abstract

This paper describes the design and implementation of a network scheduler prototype for IoT networks within the e-healthcare domain. The network scheduler combines a constraint-based task planner with the Lightweight Machine-to-Machine (LwM2M) protocol to be able to reconfigure IoT networks at run-time based on recognized activities and changes in the environment. To support such network scheduling, we implement a LwM2M application layer for the IoT devices that provides sensor data, network stack information, and a set of controllable parameters that affect the communication performance and the energy consumption.

1 Introduction

The *Internet of Things (IoT)* is an enabling technology for assisted living of the elderly at home. One such system is *E-care@home*, which combines IoT networking with health sensors, ontologies, and artificial intelligence [Alirezaie *et al.*, 2017]. In each home supported by the E-care@home system, an IoT network consisting of environmental sensors and on-body health sensors are used to monitor the activities of older adults in their home. With such a system comes certain challenges, and it is important that the purchase cost and maintenance cost be kept low if it is to be deployed on a mass scale and for a non-expert user group. Hence, low-cost hardware with considerable resource constraints is typically used, and the energy consumption must be kept low to extend the lifetimes of those IoT devices that are battery-driven. Additionally, the network communication must be dependable and efficient when low-power radios are used, while the network can be challenged by external interference.

In such contexts, we argue that it is important to enable reconfiguration of each individual IoT device at run-time based on the discovered activities and environmental changes in a home. This would enable a system to improve the performance at run-time further, as opposed to only supporting a single pre-deployment configuration of system parameters. Furthermore, as the system is deployed, network conditions may be subject to change during its lifetime and may require further (re)configuration. Additionally, different activities in a home and the degradation of health may change the requirements for data collection. Such requirements can encompass the set of sensors to sample, the sensor data fidelity, and the sensing period. For example, when the system notices that a person leaves the home, it may decide through a configuration policy that a limited set of sensors should be sampled at a lower rate until it detects that the person has returned home. Similarly, the IoT devices may experience packet losses due to external interference, which reduces the attainable data rates. Thus, it is beneficial to change the sampling frequency during run-time in order to obtain better performance while achieving an acceptable amount of data for medical diagnosis, activity recognition, and detection of emergencies.

In this paper, we present a network scheduling prototype for the E-care@home system. The main novelty of the paper is that we combine a constraint-based planner with IoT networking, and extend the language of the constraint-based planner to include the possibilities to directly control individual IoT devices. Since we are using the standard Lightweight Machine-to-Machine (LwM2M) protocol for the application-layer data, our prototype design can be used in other systems as well. The constraint-based task planner that we use is called SpiderPlan, and is available as open source[1]. This component allows us to integrate existing work on configuration planning [Köckemann and Karlsson, 2017], context-awareness and pro-activity [Köckemann *et al.*, 2015], as well as social acceptability [Köckemann *et al.*, 2014]. In the future, we plan to use the presented approach to dynamically configure the IoT network based on information taken from the smart home ontology developed in E-care@home [Alirezaie *et al.*, 2017].

The remainder of this paper is structured as follows. In Section 2, we discuss related work. In Section 3, we describe the software support required for network scheduling, covering both the client and server sides of the communication. Section 4 gives an overview of the constraint-based planner used to control network schedules according to a specified policy. Section 5 demonstrates how context-dependent configuration can be achieved using the configuration planner and a set of environmental sensor devices in an experimental E-care@home IoT network. We conclude the paper and discuss future work in Section 6.

[1] `http://spiderplan.org/`

2 Related Work

Automatic configuration is the process of determining which sensors and other devices and programs to activate and how to connect them in order to collect and process the data required in a specific context. Devices that are not needed for this purpose can be put on stand-by. In [Perera *et al.*,] configuration planning was identified as one of the fields which benefitted most of integrating IoT with context-aware computing. In this work it was denoted as automated configuration of devices where applications should be able to understand the sensors capabilities. This work claimed that still the modelling languages such as sensor markup language and sensor ontologies were too immature to consider automated configuration. We argue that in fact, if automated configuration is seen as a planning problem, the context models can be exploited to find an optimal configuration of devices. The constraint-based planner described in Section 4.1 was described by [Köckemann, 2016] who used it to provide solutions for several challenges of human-aware planning. Constraint-based planning [Frank and Jónsson, 2003; Mansouri and Pecora, 2014; Fratini *et al.*, 2008] covers approaches to hybrid planning that integrate a variety of constraints (usually temporal constraints and resources) to model complex domains.

In other works, [Lundh, 2009] describes a configuration planning approach based on an extension of hierarchical task-network planning [Nau *et al.*, 1999]. In this extension, functionalities represent information dependencies and the configuration planning problem is a combination of the HTN problem and finding a configuration that satisfies all information dependencies. [Di Rocco *et al.*, 2013] integrate information dependencies with causal, temporal and resource reasoning. Information availability is not considered in the form of temporal intervals. [d. C. Silva-Lopez *et al.*, 2015] considered multiple, partially-ordered preferences for configuration planning. The configuration planning approach used in this work [Köckemann and Karlsson, 2017] is capable of considering information availability with temporal intervals. We connect the planner used by this approach to the IoT network through CoAP/LwM2M. To our knowledge this is the first attempt to tightly integrate the context of a constraint-based planner with the resources of an IoT network.

3 Network Scheduling Support

For network scheduling support, we designed and implemented a software infrastructure based on the Open Mobile Alliance (OMA) Lightweight Machine-to-Machine (LwM2M) protocol to support network scheduling. Our software infrastructure has three main components: a LwM2M client that runs on resource-constrained environmental sensors, a LwM2M server that runs on a gateway server or in the cloud, and a configuration planner called SpiderPlan, which is a separate application that typically runs on the same machine as the LwM2M server. In the following, we will first give a brief introduction to LwM2M, and the proceed with describing the design and implementation of the aforementioned components in our software infrastructure.

3.1 LwM2M Overview

LwM2M is a standard IoT communication protocol with support for device management, an object model with resources, and reliable end-to-end transmissions with data encryption. LwM2M is based on the Constrained Application Protocol (CoAP), which is an IETF standard protocol for resource-constrained communication in the IoT. It is similar to HTTP, but is based on lightweight communication on top of UDP. Additionally, LwM2M makes it possible to write values to resources, which is important for network scheduling since it requires configuration value updates to be transmitted to sensor devices over a wireless IoT network at run-time.

LwM2M consists of three main types of entities: a server, a bootstrap server, and one or more clients. In E-care@home networks, each sensor device is a LwM2M client, whereas the gateway typically runs as a LwM2M server. The server manages registrations of newly started clients and handles incoming sensor data. It can also update clients' configuration and perform various actions. Optionally, one can also employ a bootstrap server to configure clients with a server's URL and security credentials, but we do not currently do this in our prototype system.

Each LwM2M client contains a number of objects with a specific set of resources. Furthermore, each object can have several instances. The objects can be accessed from any software capable of communicating using the CoAP protocol. Resources can be accessed as URLs of the form `coap://<ipaddress>:<port>/<objectID>/<instanceID>/<resourceID>`. We will show examples of how specific resources such as sensor values can be read through the constraint-based planner in Section 5.

3.2 LwM2M Client

In this work, we designed and implemented a LwM2M client for the Contiki-NG operating system [Contiki-NG, 2017]. Contiki-NG is focused on standard IoT protocols and has an active release cycle. Additionally, Contiki-NG has ample support for LwM2M, which fits our needs for device management, sensor readings, and reconfiguration for network scheduling. The LwM2M implementation in Contiki-NG is itself built on top of Erbium CoAP, which is designed for low-power IoT devices [Kovatsch *et al.*, 2011].

Sensor Objects. Each IoT device running the LwM2M client software in an E-care@home network deployment can provide a set of sensor objects. These sensor objects provide remotely readable resources for sensor values, measurement units, and value ranges. Each physical sensor chip on the hardware platforms used for environmental sensing in E-care@home has a unique LwM2M object. In the cases where it is possible, the sensor objects conform to standard sensor objects in OMA's registry [OMA LwM2M Object and Resource Registry, 2018]. Some types of sensors that are needed in E-care@home deployments are not defined in this registry, however, and therefore we define our own LwM2M objects for these sensors. These objects have IDs in the range of 32769-42768, which is reserved by IPSO for private use by vendors.

Configuration Object. The E-care@home configuration object can be used to control the configuration of each E-

Table 1: E-care@home Configuration Object.

Resource	ID	Type	Operations
Sensor name	1	String	Read
Object ID	2	Integer	Read
Activation status	3	Boolean	Read & Write
Sampling frequency	4	Integer	Read & Write
Update threshold	5	Integer	Read & Write

Table 2: Networking Configuration Object.

Resource	ID	Type	Operations
Target Duty Cycle	1	Integer	Read & Write
Leaf Node Predicate	2	Boolean	Read & Write
Destination IP address	3	String	Read & Write

care@home sensor on a LwM2M client. This object has ID 41914, and it contains five different resources as shown in Table 1: sensor name (ID 1), sensor object ID (ID 2), activation status (ID 3), sampling frequency (ID 4), and update threshold (ID 5). For network scheduling purposes, the sampling frequency and the update threshold are most relevant. The sampling frequency directly affects the periodicity of update messages sent to observes of a certain resource. Similarly, the update threshold controls whether updates should be sent to observers when a value has been changed by a specified amount. Hence, these parameters have a direct influence on the amount of network traffic generated by a client, and consequently its energy consumption.

Networking Configuration Object. The networking configuration allows constraint-based planner or a system operator to control a set of important parameters that affect the performance of the network. Table 2 shows our selection of initial parameters as a set of LwM2M resources. Currently, this set of resources is limited because this software is a prototype, but it is possible to add new resources as a more complete network scheduler is implemented.

The target duty cycle (ID 1) instructs the system to try to maintain a certain radio duty cycle for a specific sensor device. The number represents the percentage of time that the radio should be in either reception or transmission mode. Since the radio of the hardware platform that we use for environmental sensors typically consumes a current that is an order of magnitude higher than the microcontroller, this parameter will have a major influence on the battery lifetime. The leaf node predicate (ID 2) determines whether the sensor device should operate as a leaf node in the routing topology of an E-care@home IoT network. Leaf nodes can run using less energy in some cases, but could limit the routing options for other sensor devices that could have preferred to use the leaf node as the parent in the routing topology. The destination IP address (ID 3) can be changed during run-time in order to distribute the network load over multiple paths as the network evolves with new sensor devices and gateways.

Networking Performance Object. The networking performance object shows the statistics of a number of different parameters of importance for network scheduling. As shown in Table 3, it contains four resources. The transmission duty cycle (ID 1) shows the percentage of time that the client device's radio is in transmission mode. The reception duty cycle (ID 2) shows the percentage of time that the device's radio is in reception mode. When TSCH is used, these values are determined by the TSCH schedule in use and the amount of network traffic. The average power consumption (ID 3) is in turn highly affected by the aforementioned duty cycles, as the radio is typically the most energy-consuming chip on a resource-constrained IoT device, but it can also be affected by usage levels of the micro-controller and sensor chips. Finally, the packet reception rate (ID 4) shows how reliable the communication is. A low packet reception rate can be caused by a number of different factors such as high external or internal interference, under-provisioned TSCH schedules in relation to the traffic, or too small packet queues.

Table 3: Networking Performance Object.

Resource	ID	Type	Operations
Transmission Duty Cycle	1	Integer	Read
Reception Duty Cycle	2	Integer	Read
Average Power Consumption	3	Integer	Read
Packet Reception Rate	4	Integer	Read

3.3 LwM2M Server

The E-care@home LwM2M server is an application written in Java that is built on the Eclipse Project's Leshan software. Leshan provides an implementation of LwM2M that uses the Californium CoAP Framework [Kovatsch *et al.*, 2014]. The main tasks of the server are to (1) handle client registrations, (2) read sensor data, and (3) insert the sensor data into the E-care@home database.

We also make it possible to control LwM2M resource directly from the constraint-based task planner SpiderPlan, which controls various settings on the sensor devices in an E-care@home IoT network according to a configuration policy. In the following, we will describe this aspect in more depth.

4 Configuration Planning

Our main motivation for integrating a constraint-based planner with LwM2M is configuration planning, which is a combination of task planning with information dependencies and information goals. Task planning operators may require information to be applicable (e.g., a robot may need a map to move through the environment) and information goals may require task goals to be achieved (e.g., to use a camera for activity recognition it needs to be pointed at the object of interest). Information dependencies (as described in [Köckemann and Karlsson, 2017]) can be represented by information links of the form

$$I \xrightarrow[TR]{c} O \tag{1}$$

that require a set of information inputs I in order to provide some output information O at a cost c per time unit that the

link is used. The set TR contains any non-information subgoals (*task-requirements*) that need to be achieved to use the link. Basic information links allow to produce information directly from sensors ($I = \emptyset$). In previous work [Köckemann and Karlsson, 2017], we discussed two alternative ways of using constraint-based planning (see next section) for configuration planning. In the *goal-based approach* information is a direct effect of operators and treated no different than a regular planning problem. In a second approach we defined information processing chains from sensors to information goals as a separate sub-problem that can be solved, for instance, with a *Constraint Processing* solver. The language extension we discuss here can be applied to both approaches.

4.1 Constraint-based Planning

We use a constraint-based planner that uses flexible temporal intervals to describe the environment and allows to integrate a variety of other types of knowledge via different types of constraints. A solution to the overall problem is composed out of solutions to individual sub-problems (posed by each constraint type). A sub-problem for a constraint type can have three different outcomes: *satisfied*, *unsatisfiable*, or *flaw resolution*. In the latter case, the planner's search space is expanded with a set of choices of possible resolvers. Solvers for sub-problems use well established methods.

All information relevant to the planner is contained in a *Constraint Database (CDB)* Φ, which is a set of constraints of different types. Constraint types considered in this paper are statements, temporal constraints, and interaction constraints.

A statement $(I\ x\ v)$ models a state-variable x having value v during a flexible temporal interval I. Statements can be used to model context for the task planner (initial state, preconditions, and effects), as well as information availability, or human activities. As we will see later, we use statements to establish links to read and write LwM2M resources.

Temporal constraints decide when temporal intervals start and end by imposing, e.g., flexible durations, release times or precedence constraints. We use a quantified version of Allen's interval relations [Allen, 1984; Meiri, 1996] between statements in CDBs. We also add the unary temporal constraints *release*, *deadline*, *duration* and *at*. When writing a LwM2M resource, we determine the time at which we change the resource's value via temporal constraints. When reading values from a LwM2M resource, we will add temporal constraints to reflect *when* that value was read and for how long it persisted.

Interaction Constraints (ICs) can be used to describe complex situations as flaws and provide a choice of resolvers. For an $IC\ ic$, this situation is described as its condition CDB $Condition(ic)$ and we refer to the sequence of its resolver CDBs as $Resolvers(ic)$. If $\Phi \cup Condition(ic)$ can be satisfied, at least one $r \in Resolvers(ic)$ must be applied. This can be used in a variety of ways, such as, imposing rules for human-awareness on generated plans, modeling pro-activity, formulating context-dependent goals, or for context-awareness. In this paper we use *ICs* to impose a change on the configuration of IoT devices based on data received from them (i.e. pro-activity). There are many more constraint types available in the planner that we use (such as goal constraints or Prolog constraints). A more complete description of available constraint types can be found in [Köckemann, 2016].

Algorithm 1 Constraint-based planning

```
Require: Φ - a CDB, Θ - a seqence of constraint types
 1: function CB-PLAN(Φ, Θ)
 2:     for t ∈ Θ do
 3:         PREPROCESS-t(Φ)                          ▷ Optional preprocessing
 4:     return RESOLVE-ALL(Φ, Θ)
 5: function RESOLVE-ALL(Φ)
 6:     for t ∈ Θ do
 7:         R ← TESTANDRESOLVE-t(Φ)
 8:         if R = Failure then
 9:             return Failure                       ▷ Φ cannot be fixed
10:         if R ≠ {Φ} then
11:             for Φ' ∈ R do
12:                 Φ'' ←RESOLVE-ALL(Φ', Θ)
13:                 if Φ'' ≠ Failure then
14:                     return Φ''                   ▷ Solution from recursive call
15:             return Failure                       ▷ No working resolver
16:     return Φ                                     ▷ No flaws: Solution found
```

To solve the constraint-based planning problem we use flaw resolution search. Flaw resolution search integrates a set of dedicated solvers that return *satisfied*, *unsatisfiable*, or *a choice of resolvers*. If, given a CDB Φ, every solver returns *satisfied* then Φ is a solution. If any solver adds a choice of resolvers the search space is expanded and we start from the beginning (every solver must approve the changes). Finally, if any solver returns *unsatisfiable* the search backtracks on the last choice. If not choices remain the problem has no solution. Algorithm 1 implements this approach. It also includes an optional preprocessing step for each constraint type (lines 2 and 3). This preprocessing can be used to satisfy static constraints (e.g., Prolog background knowledge) only once before the actual search. Θ is the order in which constraint types are checked (usually sorted by complexity of the underlying solver since we want to fail as early as possible). TESTANDRESOLVE-T implements the dedicated solver for each constraint type $t \in \Theta$. A more detailed discussion of this approach and its domain definition language can be found in [Köckemann, 2016].

5 Context-Dependent Configuration

In this section we describe how we bring together the LwM2M application layer and the constraint-based planner to facilitate configuration planning. We do this by providing a language extension for the planner described in the previous section. This extension connects the statements used by the planner to LwM2M resources in two ways. First, it allows to read LwM2M resources into statements of the planner. This makes it possible for the planner to take the state of the IoT network into account when making decisions. This could influence the planner's decisions in a particular situation or lead to the addition of context dependent goals. The second direction allows the planner to write into resources of the IoT network. As a result, the decisions made by the planner can be directly applied to control the network.

The following discussion assumes that the planner works

online, meaning that it is executed at a regular frequency to resolve any flaws that may have appeared due to information collected from outside sources (such as the LwM2M resources). If the planner operates at 1Hz, Algorithm 1 is called once per second. Execution management (e.g., reading/writing LwM2M resources as described below) is not part of Algorithm 1, but changes its input Φ. Execution management happens with the same frequency as planning and is carried out before. We refer to the combination of both execution management and Algorithm 1 as *execution updates* and the preferred frequency of updates as the *execution update frequency*. If we execute at 1Hz and the planner takes too long, we may miss several updates.

The domain definition language used by SpiderPlan allows to model CDBs as lists of expressions together with their constraint type in the form

```
(:type exp1 exp2 exp3)
```

Our extension simply links CoAP addresses to state-variables of the configuration planner. To read a resource into the planner we use an expression of the following form.

```
(:coap (get variable coap-address))
```

The configuration planner will read the value found under the specified LwM2M address and write it into a statement using the given variable. If the value is the same as the previously read value, an existing statement will be extended (by updating corresponding temporal constraints). Otherwise, a new statement will be created.

Example 1 *The following expression leads the planner to read the given LwM2M resource into a state-variable* temperature.

```
(:coap (get temperature "coap://[<IP>]:5683
    /3303/0/5700"))
```

If the first value read by the planner at time step 0 *is* 30.0 *it will create the following constraints.*

```
(:statement (I_coap_1 temperature 30.0))
(:temporal (release I_coap_1 [0 0]) (deadline
    I_coap_1 [1 1]))
```

Suppose the planner reads the same value for another 5 time steps and then gets a new value of 30.5*, the resulting set of constraints would be as follows.*

```
(:statement (I_coap_1 temperature 30.0)
  (I_coap_2 temperature 30.5))
  (:temporal (release I_coap_1 [0 0]) (deadline
      I_coap_1 [5 5]) (release I_coap_2 [5
      5]) (deadline I_coap_2 [6 6]))
```

Writing resources works almost in the same way:

```
(:coap (put variable coap-address))
```

Given such an expression, the constraint-based planner will write the current value of the given variable into the resource found under the given address. This will occur at each execution update.

Example 2 *The following CDB controls a LwM2M resource of a LED. The first statement has the value* 0 *and is used from time step* 0 *to* 10*. At time step* 10*, the value changes to* 1 *for 10 time steps and then to* 0 *again for another* 10 *time steps. The date time reference expression is used to link real-world time to the planner's internal time-steps and to determine how time is represented to the user. So time step* 0 *is the current time (now), and one time step represents one second (indicated by the timespan).*

```
(:coap (put led "coap://[<IP>]:5683
    /3311/0/5850"))
(:statement (A1 led 0) (A2 led 1) (A3 led 0))
(:temporal (at A1 [0 0] [10 10]) (at A2 [10 10]
    [20 20]) (at A3 [20 20] [30 30])
  (date-time-reference "YYYY-MM-DD␣hh:mm:ss.
    fff" (datetime now) (timespan (s 1))))
```

Now we can bring together both ends and use the planner for dynamic configuration of LwM2M resources.

Example 3 *The example below reads a motion sensor and maintains* home *and* away *configurations. It uses the two LwM2M resources shown below. The resource* sampling-frequency *uses the configuration object (see Section 3.2).*

```
(:coap (get motion "coap://<IP>:<PORT
    >/41916/.../...")
  (put sample-frequency "coap://<IP>:<PORT
    >/41914/0/4")))
```

The following Interaction Constraint (IC), states that if the current configuration is away *and motion is detected, the configuration is set to* at-home *and the sampling frequency is set* 1.0*. The intersection constraint is satisfied when there is a possible overlap between the two intervals $?C1$ and $?C2$. The question-mark symbol indicates a variable term, so this constraints will be applied for all pairs of intervals using the state-variables configuration and motion). The* meets *constraint asserts that the second interval starts when the first interval ends.*

```
(:ic (home-mode) (:condition (:statement
  (?C1 configuration away) (?C2 motion 1) )
  (:temporal (intersection {?C1 ?C2}) ))
 (:resolver (:statement (?R1 configuration at-home)
  (?R2 sample-frequency 1.0) )
  (:temporal (meets ?C1 ?C2)
   (meets ?C1 ?R1 [0 0]) (equals ?R1 ?R2) )))
```

The following IC does the opposite of the previous one. If the configuration is at-home *and no motion has been detected for 600 or more seconds* $[600inf]$*, it sets the configuration to* away *and lowers the sampling frequency.*

```
(:ic (away-mode) (:condition (:statement
 (?C1 configuration at-home) (?C2 motion 0))
 (:temporal (intersection {?C1 ?C2})
  (duration ?C2 [600 inf]) ))
 (:resolver (:statement (?R1 configuration away)
  (?R2 sample-frequency 0.1))
  (:temporal (meets ?C1 ?C2)
   (meets ?C1 ?R1) (equals ?R1 ?R2) )))
```

6 Conclusions & Future Work

We illustrated how high-level automated planning can be combined with IoT networks through LwM2M for context-awareness and automatic configuration. The main benefit of this approach is that the configuration planner can take into account a variety of different knowledge types. At the server side, we have connected the constraint-based planner SpiderPlan with the deployed IoT devices through LwM2M to enable dynamic network scheduling based on a set of rules.

Possible applications for this approach include activity recognition (e.g., using Interaction Constraints), configuration planning (dynamically configure a network of IoT devices to provide necessary information) and pro-active configuration of an IoT network based on sensed information (see previous section).

As this work is in the prototype stage, there are some clear steps for future work. One would be to create and evaluate more complex domains controlling a larger number of nodes. A second direction for future work will be to integrate the work done in this deliverable with our work on planning with ontologies in order to control IoT networks by considering information stored in the smart home ontology developed within E-care@home. The work presented in this paper provides an important stepping stone in this direction.

Acknowledgments. This work was financed by the distributed environment E-care@home, funded by the Swedish Knowledge Foundation.

References

[Alirezaie *et al.*, 2017] M. Alirezaie, J. Renoux, U. Köckemann, A. Kristoffersson, L. Karlsson, E. Blomqvist, N. Tsiftes, T. Voigt, and A. Loutfi. An ontology-based context-aware system for smart homes: E-care@home. *Sensors*, 17(7), 2017.

[Allen, 1984] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[Contiki-NG, 2017] The Contiki-NG Operating System. Web page, 2017. https://www.contiki-ng.org/ (visited 2018-03-23).

[d. C. Silva-Lopez *et al.*, 2015] L. S. d. C. Silva-Lopez, M. Broxvall, A. Loutfi, and L. Karlsson. Towards Configuration Planning with Partially Ordered Preferences: Representation and Results. *KI*, 29(2):173–183, 2015.

[Di Rocco *et al.*, 2013] M. Di Rocco, F. Pecora, and A. Saffiotti. When robots are late: Configuration planning for multiple robots with dynamic goals. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5922–9515. IEEE, 2013.

[Frank and Jónsson, 2003] J. Frank and A. Jónsson. Constraint-Based Attribute and Interval Planning. *Constraints*, 8:339–364, 2003.

[Fratini *et al.*, 2008] S. Fratini, F. Pecora, and A. Cesta. Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. *Archives of Control Sciences*, 18(2):231–271, 2008.

[Köckemann and Karlsson, 2017] Uwe Köckemann and Lars Karlsson. Configuration planning with temporal constraints. In *Proceedings of the 31st Conf. on Artifical Intelligence (AAAI 2014)*, 2017.

[Köckemann *et al.*, 2014] U. Köckemann, F. Pecora, and L. Karlsson. Grandpa hates robots - Interaction constraints for planning in inhabited environments. In *Proceedings of the 28th Conference on Artificial intelligence (AAAI)*, volume 3, 2014.

[Köckemann *et al.*, 2015] U. Köckemann, F. Pecora, and L. Karlsson. Inferring Context and Goals for Online Human-Aware Planning. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2015.

[Köckemann, 2016] U. Köckemann. *Constraint-based Methods for Human-aware Planning*. PhD thesis, Örebro university, 2016.

[Kovatsch *et al.*, 2011] M. Kovatsch, S. Duquennoy, and A. Dunkels. A Low-Power CoAP for Contiki. In *Proceedings of the Workshop on Internet of Things Technology and Architectures (IEEE IoTech 2011)*, Valencia, Spain, October 2011.

[Kovatsch *et al.*, 2014] M. Kovatsch, M. Lanter, and Z. Shelby. Californium: Scalable cloud services for the Internet of Things with CoAP. In *IEEE Int. Conf. on the Internet of Things (IOT)*, Cambridge, MA, USA, October 2014.

[Lundh, 2009] R. Lundh. *Robots that Help Each Other: Self-Configuration of Distributed Robot Systems*. PhD thesis, Örebro University, School of Science and Technology, 2009.

[Mansouri and Pecora, 2014] M. Mansouri and F. Pecora. More Knowledge on the Table: Planning with Space, Time and Resources for Robots. In *IEEE International Conference on Robotics and Automation (ICRA 2014)*, 2014.

[Meiri, 1996] I. Meiri. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. In *Artificial Intelligence*, pages 260–267, 1996.

[Nau *et al.*, 1999] D. S. Nau, Y. Cao, A. Lotem, and H. Muñoz-avila. SHOP: Simple Hierarchical Ordered Planner. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 968–975, 1999.

[OMA LwM2M Object and Resource Registry, 2018] OMA LwM2M Object and Resource Registry. `http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html`, 2018.

[Perera *et al.*,] C. Perera, P. P. Jayaraman, A. B. Zaslavsky, P. Christen, and D. Georgakopoulos. Dynamic configuration of sensors using mobile sensor hub in internet of things paradigm. In *2013 IEEE Eighth Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, Australia, April 2-5, 2013*.