A Self-Governing, De-Centralized, Extensible System - The Internet of Powered Things

Horia A. Maior[1] and Shrisha Rao[2]

***Abstract*— The Internet of Things is a term to describe networked objects not traditionally thought of as computers (e.g., cars, household appliances) which may nonetheless be connected using Internet protocols and technologies (TCP/IP, etc.). Such "things" may also be connected, for control or communication, to the traditional Internet, and may themselves also be equipped with sensors or actuators to interact with their environments. It is envisioned that an Internet of Things will be useful in particular in resource-constrained systems (e.g., with smart grids). The naive approach to an Internet of Things would use a central controller or master node of some sort to oversee the activities of all "things" in the Internet of Things. However, this has obvious drawbacks, not the least being scalability. It is therefore desirable that a "thing" govern its own actions while achieving global "self" properties (such as self-adaptivity, self-stabilization) in an Internet of Things that has to work with resource constraints (e.g., limited allowable peak electricity usage for a domestic or industrial system of many "things"). Such an Internet of Things with self-governing "things" would not require a central controller, and addition or removal of "things" would be far easier. This paper provides a theoretical model of such Self-Internet of Things, giving a set of principles and properties to the system in respect to resource-constrain and energy efficiency systems. This would involve a proposed type of behaviour for a single "thing" in such an Internet of Things, as well as the principles or protocols by which such "things" are connected with one another.**

Keywords: Internet of Things, Distributed Systems, Resource Allocation and Management.

I. Introduction and Background

A. Self-governing, decentralized, extensible IoT, connected to a shared, variable power supply. Each thing is an object.

A lot of effort is invested when comes to managing and distributing resources to large, almost implied distributed systems. A common resource of this type is the power resource; powering all kinds of "things" around us, such as households appliances (fridge, computer, toaster etc.), heating systems (e.g. of the house, of the office), etc. The Internet of Things (IoT) is a term to describe such networked "things", objects not traditionally thought of as computers (e.g. cars, household appliances), which may nonetheless be connected using Internet protocols and technologies (TCP/IP, etc.). Assuming such a network of objects, connected to a shared power supply, in this paper we are creating a framework for a Self-Governing, Decentralized, Extensible, Non-Preemptive IoT, where objects communicate and cooperate together to achieve system efficiency (in terms of power consumption).

The naive approach to the IoT would use a central controller or master node of some sort to oversee the activities of all objects of the system, and make decisions on behalf of the objects (e.g. decide when the toaster should be powered). However, the approach has got clear limitations and drawbacks, not the least being scalability. We are talking about an extensible large system, where new objects can join the system at any point. One other issue may be the reliability of the system. If *the single point of failure (SPOF)* exists, the entire system would stop working, and all objects of the system would fail to get powered.

A decentralized approach however, would better fit the needs of such IoT, overtaking the limitations mentioned above. All objects of the system would required to have a controller of some kind, which provides them with computing capabilities. This would make them capable of making own decisions, therefore let us call them self-governing objects. Liu and Zhou [1] describes such property as "autonomy Feature" of objects, where objects have the ability to reason, negotiate, understand, adapt and learn from other objects or environments. Such an Internet of Things with self-governing "things" would not require a central controller, and addition or removal of "things" would be far easier.

B. Prioritized objects, cooperating and creating individual views of the system

We described an IoT system where objects are interconnected, and in the same time connected to a power supply. Each object of the system consumes an amount of power resource, that can be different from object to object (e.g. a toast will consume less power than a washing machine). We assume this amount to be constant for each object, that means each object consumes a fixed amount of power or none, and this amount cannot change in time.

The power supply is something that supplies the system with power resource. In the real world, if we think of power suppliers, they might provide a variable amount of power budget, that means at different times there is different power resource available (e.g. thinking of a wind turbine, the provided power resource would be dependant on how much wind there is). Therefore, we assume the power resource to be variable in time and dependent on some external factors. We will also assume that there is only one power supply in the IoT system.

The total demand of system at any one time (regardless of resource), as described by Rao [2], is the sum of all resource

[1]Horia A. Maior, Mixed Reality Lab, Faculty of Computer Science, University of Nottingham, United Kingdom. psxhama@nottingham.ac.uk

[2]Shrisha Rao, International Institute of Information Technology - Bangalore, India. srao@iiitb.ac.in

demands for all processes in a system. In our case, the total demand of power resource is the sum of all power demands of all objects in the system. One question is, what happens if the total demand of the system exceeds the power resource provided by the power supply. The answer is we cannot satisfy all objects with power. We are assuming that we cannot partially fulfill an object with power. It is either powered, therefore it consumes the fixed amount of power demand it requested, or not powered and consumes nothing. Rao [2] describes that in practice, distributed processes (objects in our case) drawing some sort of resource (power resource in our case), are distinguished from one another in terms of some sort of priorities. This is because some objects are more important than others in terms of their functions and utilities, and if the power supply cannot satisfy all the objects, priority will distinguish between which objects should be powered with the available power resource. Following the presented approach, objects with higher priority will be provided with power resource first (See Figure 1)

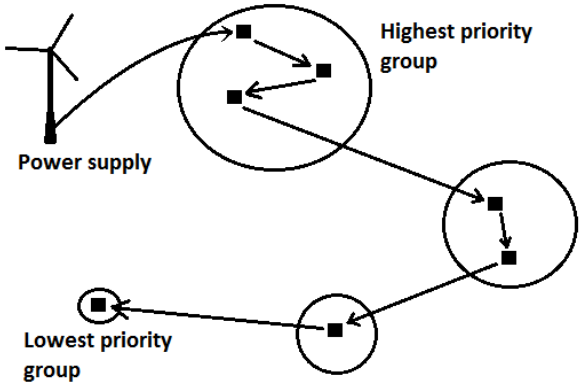

Fig. 1. The flow of powering the system after priorities.

Not having a central controller, all objects of the system need to decide when it is a suitable time for them to get powered. Therefore, they need to have a general overview of their position in the system in terms of their priority compared to other objects priorities and their power demand compared with other objects power demand. They achieve this by communicating and informing other objects about their details, while receiving information from all the other objects.

All the objects in our IoT framework are connected, networked between them such that any object in the system can communicate - send or receive a message - with/from any other object in the system. Prasad and Kumar [3] describes this particular aspect of IoT as the advance version of Machine to Machine communication, where objects are exchanging information between them without a human intervention. When sending a message, we will later describe two types of destinations: one to one message (where the message is sent from one object to another specific object), and message broadcasting (where messages are broadcasted/flooded over all other objects in the system - see Algorithm 1 in the further section). For simplicity of the model, we make abstraction of how objects are connected.

We are describing a system with non-preemptive power policy; meaning that there is no partial fulfillment. When an object was allocated power resource, it is not interrupted until it has finished its demands, or there is a change in the available power resources. In other words, if a new object with high priority joins the system at some point and a low priority object has been allocated with power, the new object has to wait until the low priority object is leaving the system, or more power is provided by the power supply. In the same time, there should be no dependencies between objects except priorities. That is when the available power resource exceeds the total demand of the system, all objects of the system shall be powered, and no object shall wait for a different object to finish its demands or leave the system.

In this section we introduced the topic and presented an overview of our interest. The following section is to describe the framework of IoT by creating a model of such system, developing some algorithms for the distributed IoT, and at the end presenting and developing some profs of correctness of our work.

II. Problem Statement and The Model

In Section I we discussed and presented our interest and in the same time the focus of this paper. In this section we will describe an abstract model of IoT within resource-constrained domain, formally stating the problem we address. We will develop algorithms working on the model developed and we will attempt to prove the correctness of the main properties of our system.

A. Definition and Notations

Let there be numerous objects o_i that demand and may consume power resource. The set of all objects, denoted as O, together with some sort of power supply constitute the whole system.

$$O = \{o_0, o_1, ..., o_{n-1}\}, \text{ the set of all } n \text{ objects.}$$

Each object consumes a non-negative amount of power resource. To simplify the system, let us consider this non-negative amount of power constant for each object in the system (an object cannot change its power demand). Let $\mathbb{R}^+$ be the set of positive real numbers, then let the demand function, $f : O \to \mathbb{R}^+ | f(o_i) = r$, give the power demand of each object o_i. This means object o_i demands r amount of power from the power supply.

The Total Demand of system at any one time, as described by Rao [2], is the sum of all power demands for all objects in the system:

$$\sum_{i=0}^{n-1} f(o_i)$$

The power supply can be described as the total amount of power resource available to share between all the objects of the system at any time. The main property of the power supply is that it is variable in time. Let the set T denote the set of all time instances, and $\mathbb{R}^+ \cup \{0\}$ be the set of

positive real numbers including 0. Then, the function $\gamma : T \to \mathbb{R}^+ \cup \{0\}$ denotes the power supply function. With $\gamma(t) = w$, w the [2] the maximum resource limit, in our case power resource, available in the system at a given time t.

We described objects having various power demands and we described a power supply providing the system with power resource. As presented in the introduction, objects also differ between them in terms of some sort of priority.

Let $\delta : O \times T \to \mathbb{R}^+ \cup \{0\}$ be the priority function, with $\delta(o_i, t) = p$, where p is the priority of object o_i at time t. As described, priority function is time dependent, that means objects can change priority in time.

B. IoT as a Distributed System

In a distributed IoT, every object is networked in some way so it is able to exchange information with all other objects in the system (for simplicity, we make abstraction of how they are physically networked). It is also assumed that every object has some sort of computing capability (we assume that computing capability has no cost what so ever), and it is in some way 'self-aware' of its current 'needs' (for example each object knows or is able to find out about its power settings). Because we are building a de-centralized system, the decision-making comes to the object itself (this means that each object in the system decides on its own when is a suitable time to consume power resource), therefore, every object is also interested in an overview of the whole system (for example each object needs to know the maximum power limit $\gamma(t) = w$ but also each object has to know about what is its priority relative to other's in the network).

When creating such a system, each object in the system is "exploring" the whole system in some way. During the exploration, objects need to find out (and memorize in some way) other objects with the same priority, and in the same time find out other priorities in the system. This way, every object will have an general overview of what is their own priority relative to other object in the system. The exploration is possible trough some sort of communication and exchange of information between objects. Making abstraction of how the communication is made, let v and u be two objects; $v, u \in \{O\}$. Let $msg(v, u, m)$ be a message, where v is the sender of the message, u is the delivery object of the message and m is the message (the message can contain any kind of information coming from the sender). Following Peleg's approach [4], a message can be also broadcasted or "flooded" to/over a network (in our case system). Algorithm 1 below, is an adapted version from [4], for broadcasting a message across all n objects of a system, from a root object o_j:

The number of messages sent by each object can be used to evaluate the complexity of the Algorithm 1 and all further algorithms developed in this paper. Each object running Algorithm 1 is sending a message to all other objects (in total $n-1$ objects); therefore the complexity of Algorithm 1 is $\mathcal{O}(n)$.

Algorithm 1: Algorithm Flood

```
Let a source o_j
for i ← 0 to n − 1 do
    if i! = j then
        Send msg(o_j, o_i, m)
    end
end
for i ← 0 to n − 1 do
    Upon receiving a message o_i
    Store the message
    Compute the message
    Send acknowledgement
end
```

Using Algorithm 1, we design an exploration algorithm that is run by all objects in the system, all trying to communicate to other objects information about their priority and their power demand. In the same time, objects receive, compute and store information about other objects. Therefore, every object will end up having an overview of their position in the system.

Let o_i be an object; $o_i \in \{O\}$. As described above, each object stores information about other priorities in the system and other objects of the same priority in the system. Let each every object o_i have to arranged lists:

- let $P_i =$ be an arranged list where each object o_i can store priorities of other objects (decreasing order); such that $P_i(0)$ is the highest priority in the set.
- let Q_i be an arranged list of objects of the same priority as o_i (increasing order); such that $Q_i(0)$ is the object with smallest demand.

Before joining the system and running the exploration algorithm, every object o_i has: $P_i = \{\emptyset\}$ and $Q_i = \{\emptyset\} \cup \{\delta(o_i, t)\}$. Please consider Exploration Algorithm (2)below.

The Algorithm 2 has complexity $\mathcal{O}(n^2)$, and this is mostly because there are n objects running the Algorithm 1, every object sending $n-1$ messages (that is $n \times n$).

Being an extensible IoT system, new objects can join at any time, and objects may no longer want to be part of the system and leave at any time. An object joins the system when it demands power resource. When joining, the new object shall run the exploration algorithm. This way, other objects will make note of the new object's details (like priority and power setting), but in the same time, the new object will create it's overview of the system trough message acknowledgements from the existing objects. An object is leaving the system when it is no-longer interested in power resource at that particular time. On leaving, the object informs all the other objects about it's intentions.

Let $B = \{T, F\}$ the set of boolean values true (T) and false (F), and function $p : O \times T \to B$:

$$p(o_i, t) = \begin{cases} \text{T} & \text{if } o_i \text{ is powered} \\ \text{F} & \text{if } o_i \text{ is not powered} \end{cases}$$

Algorithm 2: Exploration Algorithm run by all object joining the system

1 $t_0 = currenttime$
2 $o_i \leftarrow$ Flood Algorithm over G
Send: $msg(o_i, join, \delta(o_i, t_0), f(o_i))$
3 **for** $j \leftarrow 0$ ***to*** $n-1$ **do**
4 Let o_j, $(j! = i)$ receive the message
5 **Upon** receiving message:
if $\delta(o_i, t_0) = \delta(o_j, t_0)$ **then**
6 **Insert** o_i in Q_j **in order** of $f(o_i)$
7 **Send** acknowledgement $(\delta(o_i, t_0) = \delta(o_j, t_0))$
8 **end**
9 **else**
10 **if** $\delta(o_i, t_0) \notin P_j$ **then**
11 **Insert** $\delta(o_i, t_0)$ in P_j **in order**
12 **Send** acknowledgement $\delta(o_i, t_0) < \delta(o_j, t_0)$ (if so)
Send acknowledgement $\delta(o_i, t_0) > \delta(o_j, t_0)$ (if so)
13 **end**
14 **end**
15 **end**

Algorithm 3: Leave Algorithm run by every object leaving the system

1 $t = leavingtime$
2 $o_i \leftarrow$ leave the system
Flood Algorithm over G
Send $msg(o_i, leave, \delta(o_i, t), Q_i, f(o_i), powered = p(o_i, t))$
3 **for** $j \leftarrow 0$ ***to*** $n-1$ **do**
4 Let o_j, $(j! = i)$ receive the message
5 **Upon** receiving:
if $\delta(o_i, t) = \delta(o_j, t)$ **then**
6 **Remove** o_i from Q_j
7 **end**
8 **else**
9 **if** $Q_i - \{o_i\} = \emptyset$ **then**
10 **Remove** $\delta(o_i, t)$ from P_j
11 **end**
12 **end**
13 **if** $powered = T$ **then**
14 $w \leftarrow w + f(o_i)$
15 **end**
16 **end**

When an object o_i gets powered at time t, the boolean function $p(o_i, t) \leftarrow T$, becomes true. Algorithm 3 below describes exactly what happens when an object is leaving the system.

On leaving, an object sends a message to all other objects in the system. Like Algorithm 1, Leave Algorithm has the complexity $\mathcal{O}(n)$.

In Algorithm 3, the leaving object is messaging all other objects three important pieces of information:

- it's priority level $\delta(o_i, t)$ over (together with the list with other objects of it's priority Q_i)
- it's power demand $f(o_i)$.
- whether or not the leaving object was powered (function $p(o_i)$).

In the case that the leaving object has the same priority with the one receiving the message, the last one mentioned has to remove the the leaving object from the same priority objects database Q_j (see line $5, 6$ in Algorithm 3). In the case that the leaving object is the last of its priority, the whole priority level is removed from the other priorities database P_j (see line $9, 10$ in Algorithm 3). Finally, if the leaving object was powered, it's power demand becomes available for other objects (see line $13 - 15$ in Algorithm 3).

C. Prioritized objects and a happy IoT

In he subsections above, it has been described a distributed, extensible system, with objects that are able to communicate with other objects of the system, with new objects joining the system and objects maybe leaving the system. However, we did not yet mentioned how the system is powered from the power supply, that is how each individual object receives, if so, power resource.

We discussed that objects can have different priorities, and that is because some objects are more important in a way than others. In the case of not being able to satisfy all objects with power resource, higher priority objects are more likely to get power resource than lower-priority ones. We also discussed that objects can have different power demands. That is, one object might need more power resource than other ones. In the case of objects with the same priority level, the system shall always prioritize the low consumer object.

Let there be n objects $o_i \in O$, with $0 <= i <= n-1$. Each object has a power demand $f(o_i)$, a priority function $\delta(o_i, t)$, and each object has created its own overview of other objects priorities in the system (in decreasing order the list P_i; such that $P_i(0)$ the highest priority), and objects with the same priority (in increasing order of power demands the list Q_i; such that $Q_i(0)$ is the smallest demand object of $o_i's$ priority). The power supplies is denoted by the function $\gamma(t) = w$, providing the power resource available at every time $t \in T$. The following algorithm (Algorithm 4) describes how the power supplier, supplies the highest priority object with the smallest power demand first, and then how each object informs other objects about their actions and remaining power resource.

Algorithm 4 is run by all objects in the system individually. The first part of the algorithm, lines $3 - 17$, will addresses to the object with the highest priority and the lowest power demand of the system (line 3), this object having the first chance to get powered (in the case that the available power

Algorithm 4: Power Algorithm

1 $t = currenttime$
2 Let o_i the object running this code
3 **if** $\delta(o_i, t) > P_i(0)$ ***and*** $f(o_i) = Q_i(0)$ **then**
4 $\quad w \leftarrow \gamma(t)$
5 $\quad$ **if** $f(o_i) <= w$ **then**
6 $\quad\quad$ **Power** object o_i $p(o_i, t) \leftarrow T$
7 $\quad\quad$ $w \leftarrow w - f(o_i)$
8 $\quad$ **end**
9 $\quad$ **if** $Q_i - \{o_i\} = \{\emptyset\}$ **then**
10 $\quad\quad$ $nextPriority \leftarrow P_i(0)$
11 $\quad\quad$ **Broadcast** $msg(power, nextPriority, w)$
12 $\quad$ **end**
13 $\quad$ **else**
14 $\quad\quad$ $nextObj \leftarrow Q_i(1)$
15 $\quad\quad$ **Send** $msg(power, nextObj, w)$
16 $\quad$ **end**
17 **end**
18 **Upon** receiving one of the msg:
Receiving $msg(nextObj, w)$ or
Receiving $msg(nextPriority, w)$
if $(o_i = nextObj)$ ***or***
$(\delta(o_i, t) = nextPriority$ ***and*** $f(o_i) = Q_i(0))$ **then**
19 $\quad$ **if** $f(o_i) <= w$ **then**
20 $\quad\quad$ **Power** object o_i $p(o_i, t) \leftarrow T$
21 $\quad\quad$ $w \leftarrow w - f(o_i)$
22 $\quad$ **end**
23 $\quad$ **if** $Q_i - \{o_i\} = \{\emptyset\}$ **then**
24 $\quad\quad$ $nextPriority \leftarrow P_i(j) <=>$
$\quad\quad$ $(P_i(j-1) > \delta(o_i, t) > P_i(j))$
25 $\quad\quad$ **Broadcast** $msg(nextPriority, w)$
26 $\quad$ **end**
27 $\quad$ **else**
28 $\quad\quad$ $nextObj \leftarrow Q_i(j) <=> (o_i = Q_i(j-1))$
29 $\quad\quad$ **Send** $msg(nextObj, w)$
30 $\quad$ **end**
31 **end**

resource can satisfy its power demands - line $4-6$). From this point, the current object informs the next object with the same priority (if there is one - lines $13-16$), otherwise it broadcasts an information containing the new priority that shall receive power ($lines 9-12$). The second part of Algorithm 4, lines $18-31$ address all the other objects of the system, which are waiting for a message addressing them or their priority level. The smallest consumer of a priority category will always be powered first. Regardless if an object gets powered, it will always send a message to the next object or priority group. This enables small demand objects with small priority to get powered in the case of a high priority object cannot fulfil its demands with available power resource.

The complexity of Algorithm 4 in Big O notation is $\mathcal{O}(n^2)$. The worse case scenario is when all objects have different priorities, and every object is broadcasting a message once ($n \times n$).

D. Proofs of Correctness

In the previous subsections we developed a framework and created algorithms for describing our IoT model. In this subsection, we are identifying a set of properties of the system, and we are trying to prove the correctness of our proposed design by validating these properties.

Theorem 2.1: No Wastage. Guarantying that all the available power will be used by the objects of the system, and there will be not object entering a situation where there is available resource in the system meeting that objects demands but the object is not powered.

Lemma 2.2: After running Algorithm 4, $\nexists o_i \in O, (p(o_i) = F)$ **and** $(f(o_i) \leq w)$;

Proof: Let us assume that there is such a case where the Total Demand of the system cannot be satisfied by the available power resource and there exist an object starving while there is sufficient power to satisfy its demands. If the starving object o_i is the highest priority object with the smallest power demand (between objects of the same priority), then it must have get powered (line $5, 6$ in Algorithm 4). Otherwise, the starving object must have received a message from a higher priority object, and the check for enough power resource was done (line $19, 20$ in the same algorithm). Hence the contradiction. ■

Theorem 2.3: No Priority Inversion. Guarantying that high priority objects get powered first, if possible.

Lemma 2.4: $\forall u, v \in O$ with $(\delta(u, t) \leq \delta(v, t))$ **and** $(w \geq f(u)$ and $w \geq f(v))$, $\nexists$ $(p(u) = T$ **and** $p(v) = F)$.

Proof: If we assume that there exist a case where two objects $u, v \in O$, with priority of object $u \leq$ priority of object v, and the power budget can satisfy one of the two objects power demands. $p(u) = T$ **and** $p(v) = F$ is not possible after running Algorithm 4 by both objects, because the message flow (with the remaining power resource) starts from the highest priority object (line 3) towards the lowest priority object (line $9-11$). Therefore the message with the remaining power resource must have first visited object v, therefore the contradiction. ■

and Conclusion

III. Discussion, Conclusion Further Work

In Section I we described our interest and motivation for the paper. We also presented an overview of the framework specification we were to present and pointed out strengths and weaknesses of our approach. In the further section, Section II, we have described a mathematical model of the IoT, we developed four algorithms for the model, and in the last part, we developed a proof of correctness for main properties of the model/framework.

The present work has attempted to describe a framework for a real time resource-constrained problem. However, dealing with a complex system (IoT), we made a series of assumptions attempting to simplify it.

Further work in this area of research can explore issues that have not been considered in this paper, including:

- having objects that can change their power demands in time (e.g. thinking of a hairdryer machine, normally has few levels of power),
- explore what happens when the power supplier is varying the available power resource,
- and nonetheless, creating applying the framework in real life with the help of simulation.

and Conclusion

REFERENCES

[1] Y. Liu and G. Zhou, "Key technologies and applications of internet of things," in *Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference on*, 2012, pp. 197–200.

[2] S. Rao, "A foundation of demand-side resource management in distributed systems," in *Transactions on computational science VIII*. Springer, 2011, pp. 114–126.

[3] S. S. Prasad and C. Kumar, "An energy efficient and reliable internet of things," in *Communication, Information & Computing Technology (ICCICT), 2012 International Conference on*. IEEE, 2012, pp. 1–4.

[4] D. Peleg, *Distributed computing: a locality-sensitive approach*. SIAM, 2000, vol. 5.