A Distributed Control Plane for the Internet of Things Based on a Distributed Hash Table

Jaime Jiménez Bolonio[1], Manuel Urueña[2], and Gonzalo Camarillo[1]

[1] Ericsson Research, NomadicLab, Finland
{jaime.j.jimenez,gonzalo.camarillo}@ericsson.com
[2] University Carlos III of Madrid, Spain
muruenya@it.uc3m.es

Abstract. As any other communication system, the Internet of Things (IoT) requires a functional control plane. However developing such control plane in a centralized way presents a number of challenges given the multiple stakeholders, the huge number of devices distributed worldwide, their limited connectivity, and specially that most IoT devices are battery-powered and thus must be sleeping most of the time. This paper explores the possibility of employing a distributed control plane for the IoT that leverages the intrinsic scalability and flexibility of peer-to-peer Distributed Hash Tables (DHTs). In particular, it proposes using a so-called "command mailbox" resource to remotely control sleeping sensors and actuators in an asynchronous way, while also solving important issues such as device bootstrapping and security.

Keywords: Internet of Things (IoT), Peer-to-Peer (P2P), Distributed Hash Table (DHT), REsource Location And Discovery (RELOAD).

1 Introduction

The future vision of Internet of Things (IoT) is being realized by the interconnection of a wealth of heterogeneous devices [1]. Due to their ubiquity, such devices are likely to be connected to various types of networks, from NAT-based home networks to cellular or low-range wireless sensor networks. Many of them will have limited resources (i.e. computation, memory, etc.), power limitations being the most notable constrain (e.g. battery powered). Thus, it will be likely that such devices will be sleeping most of the time in order to save energy.

IoT has been defined in various ways, for our purposes we will use the definition given by [2]: *"The pervasive presence around us of a variety of things or objects which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals"*

Such cooperation is achieved by collaboration between **sensors** that gather data from their surroundings and **actuators** that interact with the physical world. A control entity, the **master**, is the one issuing the commands to the actuators and who configures both sensors and actuators. These masters are just logical entities that may be IoT devices (e.g. a light sensor that controls a shades' actuator), a simple user

D. Pesch et al. (Eds.): MONAMI 2013, LNICST 125, pp. 108–121, 2013.

application (e.g. a home automation web page), or a complex management software orchestrating many sensors and actuators (e.g. a smart city). Moreover an IoT device may be controlled by several masters simultaneously, which may be unknown at the time the device was deployed.

When analyzing the IoT traffic we consider two main types of communication: commands (i.e., sent by a master to a sensor or actuator) and data (e.g., measurements sent periodically by a sensor). Since each type of traffic has quite different communication patterns, we refer to those types of traffic using the *control-plane* and *data-plane* terms, respectively.

This paper is focused on the challenges of building a **control-plane** for the IoT, mainly the limited communication patterns of these devices, given that most IoT devices will be behind firewalls/NATs and, more importantly, that they will be sleeping most of the time in order to save battery, and thus cannot be contacted directly by their masters. A simple solution to this problem may be employing centralized servers that act as gateways between IoT devices and their masters (which may be other IoT devices themselves). However any centralized solution has a limited scalability and may complicate the multi-tenant requirements for the IoT.

Therefore this paper proposes a fully decentralized solution based on a Distributed Hash Table (DHT) that is employed as a rendezvous mechanism between IoT devices and their masters. In particular we specify how such solution may be implemented using RELOAD/Chord, although, in order to do so in an efficient and secure way, we propose a number of enhancements to the current RELOAD specification.

2 Related Work

Some works in the literature propose to connect wireless sensors with the network by using some kind of local gateway, such as a mobile device [3][4]. Some of these data-plane solutions try to decentralize those gateways, for instance by means of a distributed overlay as in [4]. Sensors then connect to peers that are equipped with both cellular and local Wireless Sensor Network (WSN) radio interfaces. The distributed gateway overlay provides functions for resource discovery, network management, storage and a rendezvous mechanism, featuring also the usual characteristics of P2P systems, such as scalability and NAT traversal.

Albeit limited, P2P systems can be implemented by constrained devices [8][9] and fit with the IoT requirements previously stated. Therefore the new enhancements we propose in this paper will be also suitable for the deployment of a fully distributed IoT scenario that does not require such local gateways, but devices will autonomously connect to the P2P IoT.

3 Background and Problem Statement

We take REsource Location And Discovery (RELOAD) [5] as the reference P2P protocol since it provides a standard, generic, self-organizing overlay network service. On top the RELOAD overlay layer different application protocols can be plugged in, such as the Session Initiation Protocol (SIP), Extensible Messaging and Presence

Protocol (XMPP) or even the Constrained Application Protocol (CoAP) [7], a lightweight client-server protocol for sensors [6] that will probably be employed in the data-plane of the IoT.

RELOAD proposes Chord [10] as its default Distributed Hash Table (DHT) algorithm to organize the overlay. It also has an integrated Network Address Translator (NAT) traversal mechanism, the Interactive Connectivity Establishment (ICE) [11]. In a distributed and heterogeneous IoT scenario, this mechanism comes very handy for interconnecting the autonomous devices, which will use whatever communication technology is available. The DHT allows for storing information in the overlay, where resources are identified by their *resource-ID*, which is usually obtained by hashing some resource's information, i.e. name, data, URI, owner ID, etc. As with other DHTs, RELOAD identifies devices by their *node-ID*, usually calculated with the same hash algorithm as the resource-ID. RELOAD supports two types of nodes: *peers* and *clients*. Peers are nodes that run the DHT algorithm, route messages, and store data on behalf of other nodes. Clients are nodes that do not run the DHT algorithm, and neither provide message routing nor storage services. Instead, they use other peers as proxies to the DHT.

Therefore, given the connectivity and resource constraints of most IoT devices, it is reasonable that they connect as RELOAD clients to a DHT composed by stable peers. However, there are some limitations in the way clients operates in the current specification, related to enabling sleeping devices and adapting current access control policies to the open and multi-stakeholder nature of IoT.

3.1 Enabling Sleeping IoT Devices in RELOAD

Although there has been a lot of research on network scalability, including P2P networks, the sleepy behavior of network devices has been considered only recently [1]. The main reason being that it changes one of the main assumptions about Internet hosts, that is, that they can be contacted at any time. Both P2P protocols, like RELOAD, and sensor protocols, like COAP, assume that nodes, either peers/clients or COAP servers/gateways, are always able to receive messages. However, this would require IoT devices to be fully awake all the time, or at least its wireless interface, which will severely limit the lifetime of any battery-power device.

Still, many do not consider this an issue, since it is assumed that wireless sensor devices just awake periodically to send one or few COAP messages with the last sensor measurement to a gateway or central server, and immediately go to sleep again [13]. Although in the data plane this client-only behavior of sensors is possible in most scenarios, this is no longer the case for the control and management planes. Although they have been overlooked by the research literature, they are essential for the correct operation of all kind of networks, including the Internet of Things. For example in order to configure a sensor with the COAP URI where to send its measurements to, the time between those measurements, or a threshold to filter unimportant events. Other example of management plane operation may be obtaining the statistics about transmitted/received packets in order to troubleshoot a problem.

To save space and cost, many IoT devices will not have a dedicated control/management interface (e.g. an USB port), but will rely on the same network interface employed for the data plane, which should be sleeping most of the time. Therefore, *it is not possible to send control or management commands to sleeping sensors or actuators*, unless some additional synchronization or rendezvous mechanism is in place. Our proposal is that low power devices can use a Distributed Hash Table (DHT) as a rendezvous mechanism to receive commands from their masters, since they may be behind a firewall or NAT and be sleeping most of the time. To do so, a sensor device creates a resource in the DHT, called *Command Mailbox*, and polls it periodically to check if it has new commands from its (potentially unknown) masters, while always-on actuators may receive these commands immediately by becoming DHT peers. Masters only need to know the node-ID of the device (either a sensor or actuator), in order to send commands to it through the DHT.

3.2 Security Considerations for the IoT Control-Plane

Although current sensor-based applications are vertically integrated, and thus a single vendor provides the whole stack, which is usually specifically designed for a given application and client, we envision the future IoT as an open and multi-stakeholder network, where interoperable devices can be provided by multiple vendors, and new applications can be deployed dynamically over the existing infrastructure. However, such open environment poses clear security challenges. In our particular case, although the commands from the master are encrypted and signed using a Message Authentication Code (MAC), the main security problem is how to allow an arbitrary number of masters (e.g. the users of a given IoT application) to write in the Command Mailbox resource of an already deployed device without using a dedicated server or directory [14].

Currently RELOAD has two control policies that can be employed to protect this kind of resource sharing: *node-based* and *list-based*, although they have some drawbacks:

- *Node-based* (USER-NODE-MATCH in RELOAD) [5] policy allows **any node of the DHT** to write one entry in any Dictionary resource applying it. However, in order to protect their own memory, peers responsible for the storage of the Dictionary resource will certainly limit its size. This could easily lead to a Denial of Service (DoS) attack, because an attacker with multiple valid certificates would be able to store a high volume of data to overflow such limit, preventing legitimate masters from storing their information. This happens because the peer storing the resource does not know which nodes are the valid masters of a device.
- *List-based* (USER-CHAIN-ACL in RELOAD) [15] policy is not vulnerable to this attack, because only the nodes **explicitly allowed** by the owner of the resource are able to write data. However, this requires the device to know the list of legitimate masters beforehand. Moreover, a device cannot be shared by an arbitrary number of users, because that would be too great a burden for the peer

managing such list, as well as all associated certificates - thus the dedicated server approaches such in [14]. In the IoT context, it implies that our sensor device should have to know beforehand the user-IDs or node-IDs of all possible masters before being deployed. This obviously complicates the management and deployment of IoT devices, especially in the cases of embedded ones that do not have a dedicated control interface. Therefore, a simpler mechanism to enable initially unknown masters to write commands in the resource of a deployed embedded device would be useful.

4 Proposed Solutions

As we have seen in the previous section, controlling sleeping IoT devices is problematic because those devices are asleep most of the time. In addition, current DHTs do not implement access control policies suitable for the large amount of devices that will be part of the IoT future scenario and their multi-stakeholder management. In this section we tackle those problems and present solutions to them.

4.1 A Command Mailbox for DHTs

Taking Chord's implementation in RELOAD [5] as a model, devices that want to use this method should be pre-configured at least with the following information:

- The node-ID of the device, which must be globally unique.
- A valid certificate from the Enrollment Server of the DHT.
- The DNS name of the overlay, or the IP addresses of Bootstrap Servers.
- An optional, randomly generated, *secret key*. The master can later change this key, but if the device is reset, it goes back to this initial secret key.

In order to issue commands to the device, its masters must know the device's node-ID and its current secret key. This information may be shared in several ways, for instance it can just be printed in the manual or in the device itself. In any case, the master should change the secret-key as soon as possible (i.e. the very first control command). Notice that it is not necessary neither that the master knows whether the device is a sensor or an actuator, nor the device has to know whom its master or masters are beforehand, which greatly simplifies the bootstrapping and the dynamic ownership of IoT devices.

The proposed command mailbox resource is to be used by sensors, actuators and masters. The initial operation procedure after being reset (see Figure 1) is the same for devices plugged to the grid (e.g. actuators), thus awake and connected all the time, and for battery-powered, or otherwise intermittent-connected, devices (e.g. sensors).

After booting, an IoT device contacts the Bootstrap Server to obtain the overlay configuration and to identify its Admitting Peer, i.e. the one responsible for the node-ID of the joining node. Then, in the case of an always-on actuator, since it has enough resources to be a full peer, it joins the DHT as a RELOAD peer through its Admitting Peer. A sensor, on the other hand, attaches to its Admitting Peer as a RELOAD client, and thus has no routing or storage responsibilities.

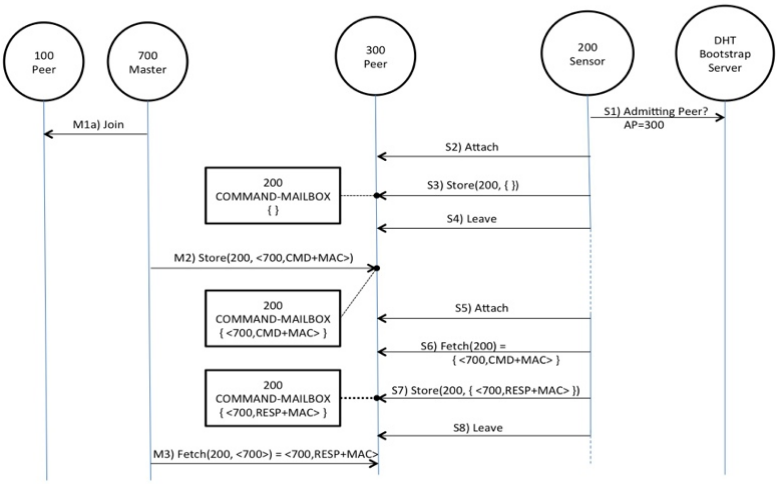

Fig. 1. Issuing a command to a sleepy client device (e.g. a battery-powered sensor)

Then the booting device creates in the DHT a Dictionary resource with the new *COMMAND-MAILBOX* kind-ID and the proposed *CLIENT-ID-MATCH* policy that enables it to store it with the same key as its own node-ID. Note that this behavior is not contemplated in current RELOAD specification because, to limit resource consumption and provide more availability, data can only be stored when the resource-ID are either a hash of the owner's user-ID or node-ID. In the case of an actuator peer, since it is the responsible for its own command mailbox resource, it is not necessary to create a real resource (e.g. reserve memory or replicate it), but just to process the DHT messages targeting it. In the case of a sensor device, the command mailbox is created in its Admitting Peer by performing a DHT store operation with the same key as the sensor's node-ID.

At this point, once the Command Mailbox resource has been created, a sensor device may leave the Admitting Peer and go to sleep for some pre-defined amount of time between command checks (e.g. 10 minutes). Conversely, an actuator remains connected as a peer in order to receive messages from its masters, requesting a store operation in its Command Mailbox resource. Then, when the sensor awakes, it attaches again as a client to its Admitting Peer and fetches its mailbox looking for any new command. Both types of devices can verify the MAC code of the incoming command, and optionally decrypt it, to guarantee that it comes from a valid master that knows its current secret key.

Therefore, when a master has to issue commands to any of the IoT devices it controls, it only has to be connected to the same DHT (either as a peer or as a client). Then it can issue control messages just by storing the encrypted and signed command into the Command Mailbox resource with the same resource-ID as the target device's node-ID. Thus, if the target device is an actuator, the command will be received directly by the actuator peer and act accordingly. Sensors on the other hand will only receive such command after checking the Command Mailbox at its Admitting Peer.

Once all commands have been executed or discarded (i.e. wrong MAC), the device overwrites the whole Command Mailbox resource. If necessary it can write a response to the command in the same mailbox, or sent it back to the master using other mechanisms, like a RELOAD message routed through the overlay, or a direct COAP message. If no response is needed, the command-mailbox entry can be just left empty.

To check that the device has processed the command, the master can just wait for a direct response message (e.g. through the overlay) or, if the master is a sleepy node itself, it may then try to fetch the same resource to check the response of the device, or that at least verify if the key is empty, i.e. not containing its own command.

4.2 Access Control Policy for RELOAD Based on Shared Keys

Previously we have discussed how to solve the problem of sending control or management commands to sleeping sensors or actuators. Now we address the issue of sharing write permissions of the command mailbox resource with other (potentially unknown) nodes in the overlay to prevent Denial of Service (DoS) attacks. To do so, we propose the use of a shared-key mechanism (see Figure 2).

In our IoT deployment scenario every embedded device can be deployed without a pre-configured list of masters, but rather just with a randomly generated secret key. Any master would only need to know the device's identifier and its initial secret key, which can be printed in the manual or in the device itself. Therefore a device can be easily controlled by an arbitrary number of masters without managing an explicit access control list.

A device that wants to create a Command Mailbox resource and share its write permissions with its (unknown) masters by means of the proposed shared-key mechanism should follow two steps:

1. Establishing a Write Key: The device sends a `Store Request` message with the same ID as the Command Mailbox resource being protected and a value identifying it as a Write Key (in RELOAD this can be done by editing part of the `StoreKindData` with an additional `key_sign_type = WRITE_KEY` attribute). The message should also define the Message Authentication Code (MAC) algorithm to be employed by other nodes, and include the **shared key** associated to that resource, and that may be derived (e.g. by hashing) from the device's secret key. The Write Key is encrypted with the public key of the peer storing the resource, so it can be securely forwarded through the overlay. Only the owner node can change the secret key or the MAC algorithm at any time by sending a new `Store Request` message with a different `write_key` field.

2. Storing in the Resource: After establishing the Write Key of the Command Mailbox, any node that knows such key is also able to store information in the shared resource. Those messages need to include a `write_sign` signature field containing the MAC value of the whole message structure including the resource-ID, by using the current Write Key. The MAC algorithm must be the one specified in the `mac_algorithm` field of the `Store Request` operation by the owner device. When considering RELOAD, other operations (i.e. Fetch, Stat, etc.) are not affected since the `write_key_sign` field only appears in Store messages.

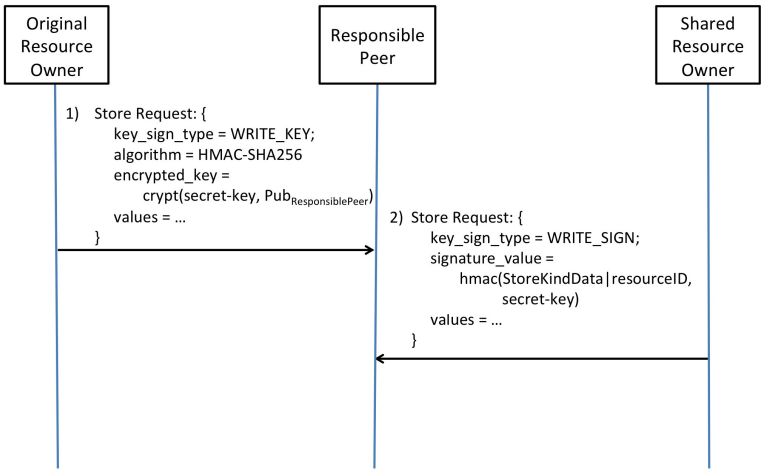

Fig. 2. RELOAD resource being shared between two owners

When the responsible peer needs to **replicate** the shared resource in one or more replicas, it should also include the whole `write_key` field in the `Store Request` message sent to the replica, and optionally encrypting the write key with the replica's public key. Usually the certificates of the replicas should be in the responsible peer's cache, but if not, the key can be obtained by sending a RELOAD ping message to the replica's node-ID.

5 Scalability Analysis

This section analyzes the scalability of the proposed DHT-based IoT control plane by comparing it with using centralized servers, as well as analyzing the effect of the proposed enhancements to RELOAD. To do so, let us define N as the total number of nodes in the Internet of Things (IoT):

$$N = N_S + N_A$$

Where N_S and N_A are, respectively, the total number of sensors and actuators in the IoT. We will assume that actuators are mains-powered and have a permanent connection (although they may be behind NATs/firewalls), so they can behave as DHT peers. On the other hand, sensors are sleeping most of the time so they only wake up periodically to check their command mailboxes. Due to this intermitted connection sensors are not full peers, but connect as RELOAD clients to their Admitting Peers.

To model the IoT control traffic, we will also define Ox as the average rate, measured in messages per time unit, of control operations issued to a given node of type x. Thus, O_s and O_a are the average rate of control operations issued to a sensor and an actuator, respectively (e.g. $O_a = 1$ means that each actuator receives on average one control operation per time unit). P_s and P_a are defined as the rate sensors and actuators poll their associated Command Mailbox (e.g. $P_s = 1$ means that each sensor polls its command mailbox once per time unit).

Now we can define M_X as the total number of control messages exchanged (either sent or received) per time unit by all nodes of type X. Then M_C is the total number of

messages sent or received per unit time by all servers in the centralized scenario, and $\boldsymbol{M'_{DHT}}$ specifies all messages generated within the RELOAD DHT. Notice that forwarding a control message, either by a central server or a RELOAD peer, involves one reception and one transmission, and thus it is accounted as two messages.

Finally, the metric employed to compare the scalability of the different scenarios is L_x, the average load of a node of type x, measured as the average number of messages exchanged by a node per time unit. It is computed as the total number of messages (M_X) divided by the number of type X nodes (N_X):

$$L_x = \frac{M_X}{N_X}$$

Next we provide an analytic model for this load metric in the three evaluated scenarios.

5.1 Centralized Scenario

In this scenario, the control plane of the IoT is provided by a set of $\boldsymbol{N_c}$ centralized servers. We will assume a perfect load balancing strategy so all servers handle exactly the same number of messages. Then, actuators have a permanent connection with one of these servers, which acts as a proxy of control commands, sending back and forth the operation requests from masters and the replies from actuators. Therefore, the total number messages exchanged by all IoT actuators is just two times (i.e. request + reply) the average control operation rate of a single actuator (O_a) multiplied by the total number of actuators (N_A):

$$M_A = 2O_aN_A$$

Sensors, on the other hand, do not have such permanent connection and thus cannot receive control commands from their masters directly. Instead these masters' operations are stored in a Command Mailbox at the central servers while the master waits for a response. Sensors poll its mailbox periodically (at P_s rate), and when it contains a new operation request, it is executed and the response is sent back to the user through the central server. Therefore the total number of messages exchanged by the sensors is the sum of the polling ones plus the two request/response messages exchange with the master:

$$M_S = 2P_sN_S + 2O_sN_S = 2N_S(P_s + O_s)$$

Then, the total number of messages exchanged by all masters is:

$$M_M = 2O_aN_A + 2O_sN_S$$

Therefore, since the central servers forward all messages exchanged among masters, sensors and actuators, the total number of messages handled by all servers (M'_C) and the average load per server (L_c) are:

$$M_C = 4O_aN_A + 2N_S(P_s + 2O_s)$$

$$L_c = \frac{M_C}{N_C}$$

5.2 Standard DHT without the Proposed RELOAD Enhancements

To avoid the extra cost and single point of failure of centralized servers, in the following scenarios the control plane is provided by a global DHT comprised by all IoT actuators that are powerful enough to act as RELOAD peers. Sensors act as RELOAD clients and are connected to the DHT through their Admitting Peers. Masters that want to issue control commands are also connected as RELOAD clients in order to write the desired operation in the Command Mailbox of the target IoT node and then, as in the previous case, wait for a direct response forwarded through the DHT. Both operations are acknowledged to provide a reliable service.

However in the second scenario we will first not consider the proposed RELOAD enhancements. Therefore the Command Mailbox resource of a sensor or an actuator is stored by some (random) peer in the DHT obtained by hashing its node-ID, and both sensors and actuators have to periodically poll it in order to check if there is any new operation request. In that case, they execute it and send the reply back through the DHT to the master (which has provided its node-ID in the command request) that finally acknowledges back to the device. Therefore the number of messages exchanged by all masters, sensors and actuators among them, as well as with the peers storing their respective Command Mailboxes are:

$$M'_S = 2P_sN_S + 2O_sN_S = 2N_S(P_s + O_s)$$

$$M'_A = 2P_aN_A + 2O_aN_A = 2N_A(P_a + O_a)$$

$$M'_M = 4O_sN_S + 4O_aN_A$$

However each of these messages, exchanged either with the peer storing the command mailbox or with the master, must be forwarded through the DHT. For a Chord ring with N_{DHT} peers, the average number of hops is $H = log_2 N_{DHT}$ [10] and, since each hop requires receiving and sending each forwarded message, the total number of forwarded messages doubles. Moreover there is certain overhead in order to maintain the DHT structure. In case of RELOAD/Chord an update message must be sent to all neighbor peers every 10 minutes (U_{Neighs} = 6 msg/hr), while a search for best fingers is performed each hour ($U_{Fingers}$ = 1 msg/hr). Given that each peer must have 16 fingers and 22 neighbors (i.e. 3 predecessors + 3 successors + 16 fingers) [5], the total number of overhead messages per time unit to maintain the DHT is:

$$M_{DHT} = 22U_{Neighs}N_{DHT} + 4H16U_{Fingers}N_{DHT} = N_{DHT}(22U_{Neighs} + 64HU_{Fingers})$$

Then, the total number of messages (exchanged among masters, sensors and actuators) sent and received by all DHT peers (M'_{DHT}), and the average peer load (L'_{DHT}) are:

$$M'_{DHT} = 2H(2P_sN_S + 2P_aN_A + 4O_sN_S + 4O_aN_A) + M_{DHT}$$

$$L'_{DHT} = \frac{M'_{DHT}}{N_{DHT}}$$

5.3 Enhanced RELOAD DHT

In the third scenario, the IoT control plane is also provided by a global DHT, but now it employs the proposed RELOAD enhancements. This way, the Command Mailboxes of sensors and actuators are not stored in remote peers, but in the sensor's Admitting Peer or at the actuator itself. This greatly reduces polling overhead because allows actuators to receive commands immediately, and polling messages from sensors are not forwarded several hops away through the DHT, but just target the local Admitting Peer. Therefore the total number of messages exchanged among masters, sensors and actuators in this distributed scenario are:

$$M''_A = 4O_aN_A$$

$$M''_S = 2N_S(P_s + O_s)$$

$$M''_M = 4O_sN_S + 4O_aN_A$$

Now only the messages among masters and actuators, sensors or their Admitting Peers have to be forwarded in the DHT (still with H hops on average), while sensors' polling messages reach the Admitting Peer directly, leading to the following total number of messages (also considering the M_{DHT} overhead) in the DHT and the average load per peer:

$$M''_{DHT} = 2H(4O_sN_S + 4O_aN_A) + 2P_sN_S + M_{DHT}$$

$$L''_{DHT} = \frac{M''_{DHT}}{N_{DHT}}$$

6 Evaluation

In order to evaluate the scalability of the three analyzed scenarios, we should compare them with similar parameters. Therefore let us assume that the control operation rate for sensors is just one operation per node and day (O_s = 1 op. per node and day = 1/24 op. per node and hour) while actuators receive one operation per hour (O_A = 1 op. per node and hour). The Command Mailbox polling rate is also the same for sensors and actuators, but faster to reduce the response time (P_s = P_a = 1 poll per node every minute = 60 polls per node and hour). To do not favor any device type, let us assume that half of the IoT devices are sensors and the other half actuators (N_s = N_a = $N/2$). Moreover, all actuators form the IoT control-plane DHT (N_{DHT} = N_a).

Figure 3 shows the scalability of the three proposed scenarios. Each curve represents the average load per node measured in messages per hour of each solution. That is, L_c, L'_{DHT} and L''_{DHT} that were defined in the previous section. Since the average load depends on the number of nodes, the centralized solution has four curves for different number of servers, ranging from one to 10,000. Notice that both axes are in log scale.

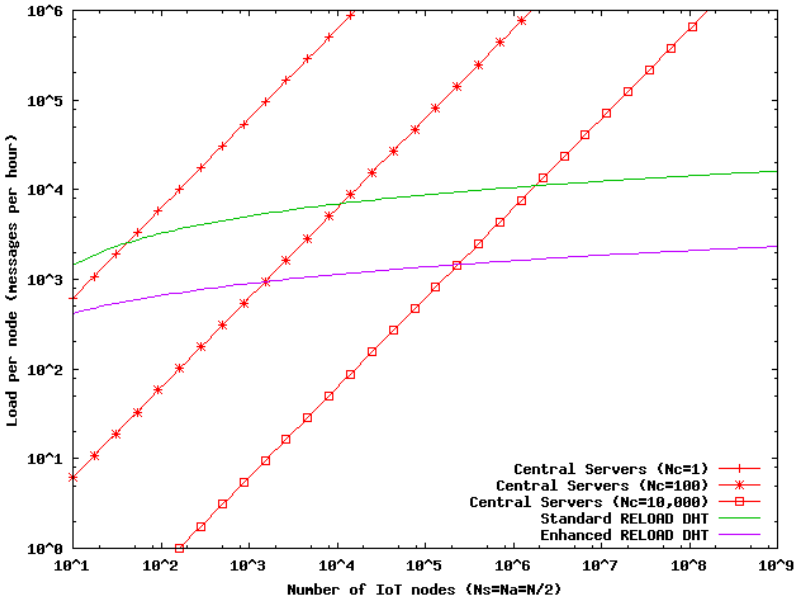

Fig. 3. Load per node in the three evaluated scenarios (L_c, L'_{DHT}, L''_{DHT})

It can be clearly seen that the centralized solution requires deploying a high number of servers in order to maintain the same average load as the DHT-based solutions, which is low enough (2,351 msg/hr = 0.65 msg/sec with one billion - 10^9 - IoT nodes) to be supported even by constrained devices. Moreover, the proposed enhancements for RELOAD reduce one order of magnitude the total number of messages in the DHT by avoiding polling remote Command Mailbox resources, which constitute a high share of the total traffic.

However, due to the overhead to maintain the DHT, a distributed IoT control plane does not make sense for small IoT deployments that may simply rely on few centralized servers. On the other hand, a DHT composed by IoT actuators does not require deploying and maintaining additional infrastructure, and continuously upgrades as the IoT grows, given that the new actuators will help providing the IoT control plane.

7 Conclusion

This paper provides the foundation of a fully distributed control plane for the Internet of Things (IoT). It provides an asynchronous mechanism to send control plane commands (e.g. CoAP, SNMP or custom ones) to intermittently connected devices. Moreover this solution enables the IoT to be managed by different stakeholders, which do not need to deploy its own infrastructure but may rely on the existing IoT devices themselves.

For implementation purposes, we focus on the IETF standard P2P protocol: RELOAD. Our enhancements would require minimal modifications to the current specification. Since our mechanism is fully distributed it takes advantage of the inherent properties of distributed systems such as scalability, NAT traversal and autonomous operation. Moreover, since the IoT control and management traffic is not concentrated in a single point of the network, but it is distributed among all peers of the DHT, it does not have a single point of failure as centralized solutions. Our estimations show that the overall traffic in the DHT is quite low since most messages are sent towards their final destinations (i.e. actuator peers), or through a direct connection (e.g. sensors polling their Admitting Peers). Moreover, our analysis shows that the proposed DHT solution has remarkable scalability properties when compared with a centralized solution, since the growth of the IoT as a fairly limited impact on the control-plane load of DHT peers.

We have also provided a simple access control policy to enable a RELOAD resource, such as the proposed Command Mailbox one, to be shared by an arbitrary number of nodes, without requiring an explicit list of allowed devices as it happens now with access list-based policies. Moreover the proposed resource sharing policy does not allow external nodes (i.e. that do not know the shared key) to write any data in the shared resource, and thus is not vulnerable by design to the Denial of Service (DoS) attacks against node-based policies. This is a final requirement for an open, but secure, distributed management of IoT devices.

Acknowledgments. The authors would like to thank Heikki Mahkonen and Petri Jokela for their help with this work and for reviewing the final document.

References

[1] Mazhelis, O., Luoma, E., Warma, H.: Defining an Internet-of-Things Ecosystem. In: Andreev, S., Balandin, S., Koucheryavy, Y. (eds.) NEW2AN/ruSMART 2012. LNCS, vol. 7469, pp. 1–14. Springer, Heidelberg (2012)

[2] Atzori, L., Iera, A., Morabito, G.: The Internet of things: A survey. Computer Networks 54(15), 2787–2805 (2010)

[3] Hu, F., Rajatheva, N., Latva-aho, M., You, X.: Sensor Integration to LTE/LTE-A Network through MC-CDMA and Relaying. VTC Spring, 1–5 (2012)

[4] Mäenpää, J., Bolonio, J.J., Loreto, S.: Using RELOAD and CoAP for wide area sensor and actuator networking. EURASIP Journal on Wireless Communications and Networking 2012(1), 121 (2012)

[5] Jennings, C., Baset, S., Schulzrinne, H., Lowekamp, B., Rescorla, E.: REsource LOcation And Discovery (RELOAD) Base Protocol. In: 2013 IETF Internet-Draft, Intended status: Standards Track

[6] Shelby, Z., Hartke, K., Bormann, C.: Constrained Application Protocol (CoAP). In: 2013 IETF. Internet-Draft, Intended Status: Standards Track (2013)

[7] Jimenez, J., Lopez-Vega, J.M., Maenpaa, J., Camarillo, G.: A Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD). In: 2013 IETF. Internet-Draft, Intended Status: Standards Track (2013)

[8] Mäenpää, J., Bolonio, J.J.: Performance of REsource LOcation and Discovery (RELOAD) on Mobile Phones. In: 2010 IEEE Wireless Communications and Networking Conference (WCNC). IEEE (2010)
[9] Tozlu, S., Senel, M.: Battery lifetime performance of Wi-Fi enabled sensors. In: 2012 IEEE Consumer Communications and Networking Conference (CCNC). IEEE (2012)
[10] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup protocol for internet applications 11, 17–32 (2003)
[11] Rosenberg, J.: Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245 IETF (2010)
[12] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making gnutella-like p2p systems scalable. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 407–418. ACM (2003)
[13] Vial, M.: CoRE Mirror Server. Draft-vial-core-mirror-proxy-01, IETF Internet-Draft, Intended Status: Standards Track (2013)
[14] Sturm, C., Dittrich, K.R., Ziegler, P.: An access control mechanism for P2P collaborations. In: Proceedings of the 2008 International Workshop on Data Management in Peer-to-Peer Systems. ACM (2008)
[15] Knauf, A., Schmidt, T.C., Hege, G., Waehlisch, M.: A Usage for Shared Resources in RELOAD (ShaRe). Draft-ietf-p2psip-share-01, 2013 IETF Internet-Draft, Intended status: Standards Track (2013)