A Process-Based Internet of Things

Socrates Varakliotis, Peter T. Kirstein
UCL Computer Science
Gower Street, London, WC1E 6BT UK
Email: {S.Varakliotis, P.Kirstein}@cs.ucl.ac.uk

Antonio Jara, Antonio Skarmeta
Computer Science Faculty
University of Murcia
Email: jara@ieee.org, skarmeta@um.es

***Abstract*—The Internet of Things (IoT) is envisaged as a unified network of 'smaller' networks that live on the fringes of the Internet, such as systems that monitor and control buildings, industrial plants, the power grid, etc. The Internet Protocol (IP) has been promoted as the transport glue that implements this vision. Although we observe a converged view from the various application domains that their gateways should adopt IP, it is clear that the path to full adoption is long. The gateways may impose very specific requirements of security, resilience to failures, ease of maintenance and upgrade. We present the design goals of an enhanced architecture for the IoT that can reduce such concerns, paving the way for faster adoption of IP in the IoT. We argue that the functions one needs to perform in IoT networks of various application domains can be summarised in a small group of basic request-response operations, which traverse gateways and act as the transport layer. We describe how more complex functions, abstracted into server-based processes, can then be executed by the basic transport operations. With this approach we aim to reduce the technology-specific functions of IoT gateways, which now even become transparent, at best.**

I. Introduction – Problem Statement

In the Internet of Things (IoT) a key role is played by the gateways and the services that they must perform. In many projects, and in particular in the EC FP7 Project IoT6 [1], an architecture has been defined which distinguishes between the general Internet, a ServiceNet – in which the services specific to the domain are carried out, and a DevNet – which deals with the actual physical devices (see discussion on the Architecture in Section III and Fig. 1). Because of the large number of devices that will be required in the IoT, and the benefits that would accrue from recent protocol advances, it is clear that IPv6 would be desirable particularly in the ServiceNet. The processing power of the devices themselves is also advancing, but because their applicability covers so wide a spectrum of activities, it will be some time before we can expect them all to be IPv6 enabled. Many will continue to use the legacy networked systems, like KNX [2], BACnet [3], X10 [4], in the Buildings Automation Systems domain. The prevalence of Internet technology is encouraging suppliers to provide Internet interfaces to such systems to allow them to be controlled remotely – but normally these use the Internet just to carry the control and monitoring functions for remote operation. There is a growing tendency to have sensors and actuators operate over wireless rather than cabling – partly for flexibility and partly to reduce installation costs. Many of the recent developments of Internet protocols are directed at standardising the above procedures.

The nature of the environments in which IoT is starting to be deployed is giving an increasing importance to having the control and monitoring functions run as services that are related to the application domain, and are agnostic to the technology of the devices. In addition, there are very important requirements of security, resilience to failures, ease of maintenance and upgrade. These functionalities contribute significantly to the complexity of the gateway functions that must be performed. At the same time, the very prevalence and nature of the devices makes it important that they be as resource-efficient and power-efficient as possible.

The trend in many projects and products has been to incorporate most of the functionality needed into the gateways – or to relax some of the facilities for security and re-configuration to allow less powerful hardware to be used in the controllers/gateways. At the same time there have been very significant developments in Internet protocols to help simplify and standardise that portion of the communications architecture. Of particular importance are the following:

- The application interface of CoAP [5], which can optionally operate in a secure and reliable fashion between entities at the datagram level with reduced resources. It has been accompanied by the notion of CoAP proxies, which allow functionality to be carried out in other HTTP- or CoAP-enabled systems on behalf of the original requestor.
- The wireless communications systems based on IEEE 802.15.4 [6], 6LoWPAN [7], RPL [8], which allow operation over low-power wireless networks – even when some nodes are deliberately powered off to reduce power consumption (low duty-cycle systems).
- Web Services are based on streams over HTTP; this is now very prevalent, and can be configured to provide secure and reliable service.

Finally, there have been two other developments. There is a large growth in Cloud Computing, which allows computing processes to be run remotely, and is a very energy-efficient way of having powerful servers available in a resilient cost-effective way. Secondly, there has been a realisation that the original concept of the Domain Name Service (DNS) that is at the heart of the Internet could be extended to distributed clusters of repositories. These can hold securely replicated data relevant to IoT deployments. The repositories can have both a local secure updating of their data, and yet a universal secure access to this data.

This paper shows how these different threads can be brought together to provide IoT systems that can make maximum use of on-going developments to provide cost-efficient, power-efficient, secure and maintainable systems. The key

978-1-4799-3459-1/14/$31.00 ©2014 IEEE

to the methodology is to *reduce the functionality needed in the device gateways themselves to a minimum, while putting much of the technology-dependent functionality in general-purpose servers – which could even be part of cloud-computing clusters*. We argue that it is possible to define the whole gateway functionality as a number of separate processes, and to carry these out either in the same or different processors. It is only necessary to have one of the processes accessible 'from outside' by HTTP or CoAP. All others can then be accessed securely via CoAP proxies.

II. Relevant Literature

Many authors have made contributions to the subjects treated in this work. We highlight in the following paragraphs the pertinent approaches to gateways[1]:

Server-based logic for the IoT has been discussed by Kovatsch et al. [9]. They propose an IoT architecture which decouples infrastructure from applications. Then the main logic of the devices is implemented on servers in a cloud system. This approach respects nicely the *end-to-end Internet principle*. We adopt this approach and build our IoT enhanced architecture on same design principles, emphasising on the network communications.

In a complementary approach, Ericsson Labs [10] proposes a gateway framework for legacy device integration, called Generic Device Access, which encapsulates various *base drivers* at the lower layers to terminate the specific protocol stacks "southbound" towards the devices. They then expose the specific device capabilities to applications with *protocol adapters*, written as a set of general Java APIs, in an OSGi environment.

The strength of both papers above is that the authors did not limit their vision to a single application domain (i.e. buildings, or smart grids, etc.) but realised that a generic, but complete, communications stack can be built that directly integrates sensor and actuator networks (including RFID, PLC, etc.) into the Internet, based on lightweight RESTful web services.

Jung et al. [11] allow the interconnection of heterogeneous legacy technologies by use of "multi-protocol gateways" with IPv6 in the network layer. The novelty of their design lies in the utilisation of EXI-compressed messages to bind well-identified functions of Buildings Automation Systems (BAS), called *contracts*, to CoAP. The approach is tied to the well-defined BAS management framework oBIX and therefore limits the reach of such gateways to the BAS domain, which is rather dominated by legacy systems. Furthermore, Jung's approach could be considered an instance of Ericsson's gateway with an *oBIX protocol adapter*.

Use of the Handle system as a resource resolver for the IoT has been proposed [12]. While an initial assessment of the Handle system and protocol [13] revealed that it was not IPv6 ready for direct integration with *IoT services*, as defined in Section III, this has since been remedied. Further aspects of such integration and the relevant functions are detailed in Sections IV and V.

[1]A comprehensive review, including various data representations and protocols (EXI, JSON, REST, oBIX, CoAP, etc.), exists in [18]

The above key literature has laid a solid foundation of design principles, components and technologies one could employ in the design of an efficient, scalable, secure and maintainable IoT architecture. In the following sections we gather the best elements of all above works and illustrate these in the implementation of an advanced architecture for the IoT, which is based on processes and (clusters of) servers. This way we decouple the application from the infrastructure, where possible, and build generic transparent gateways where the application-layer protocols are agnostic to the underlying network and vice versa, reminiscent of the gateway-to-router evolution in the Internet over the past 30 years. Furthermore, we are not limiting the gateway design to one application domain. We assume, however, that in all domains, the devices are capable of profiling their constrained capabilities in CoRE fashion [14], or by adopting the IPSO Application Framework [15], or otherwise. Such profiles can then be resolved by systems like Handle.

III. IoT Architecture and Functions

Our concept of the generic architecture of the IoT is shown in Fig. 1 and has been introduced previously in [16].

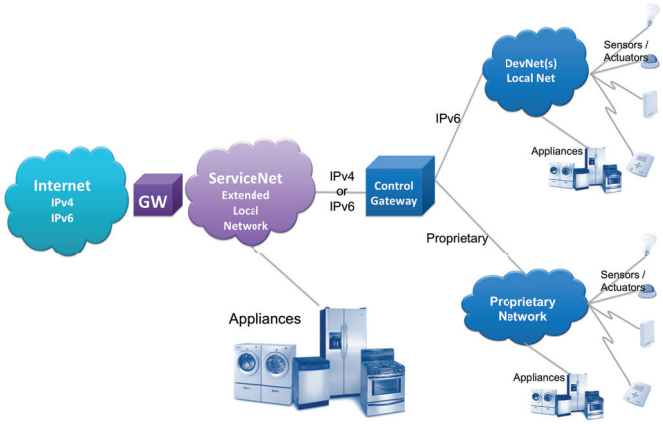

Figure 1. Schematic of the Internet of Things.

Here the primary purpose of this paper is to consider the architecture of the gateway functionality between the DevNet and the ServiceNet. We identify below a set of functional components, which we will call *IoT services*, that we believe should serve as the fundamentals of the IoT infrastructure in the ServiceNet.

- A name-to-address **resolver**, which uniquely maps a generic name ('handle') to an ID representing an IoT resource
- A **configuration manager**, which is the key location where IoT devices could look up, retrieve and upload their configuration status
- We have assumed that IP (IPv6) is the key network addressing protocol of the end-devices, therefore we expect to see a universal mapping between IPv6 addresses and resources/devices. For IP-enabled objects the IPv6 addressing is a given. For legacy technologies IPv6 addressing can be achieved through some **IP addressing** proxy mechanism [17]

- We want all of the above operations to be performed in an as **secure** an environment as possible, between a device/resource and an IoT server

In the following sections we describe the proof-of-concept mechanisms of an enhanced architecture with the *basic IoT services* described above (resolver, configuration manager, secure operation and IP addressing proxy). In ongoing work [18], we describe how **group communications** (e.g. multicast) and **translators** (such as multicast-to-unicast, or IPv4-to-IPv6 bridges) can be fitted in as *enhanced IoT services*. The resulting enhanced architecture is analysed later (Fig. 4, Section V.)

In theory, the IoT services described above could be implemented anywhere[2]. In fact it may well be possible to locate some of the services in the Internet itself. Which part of the Internet would be an implementation optimisation that we will not consider in detail in this paper. In contrast, we maintain here that frequently the more appropriate location to run such services is in the network vicinity of the ServiceNet, or the gateway to the DevNet. (This makes the whole advanced architecture compatible with other architectures.)

We show DevNets of two types:

- The current generation of legacy networks which may adopt proprietary protocols, or ones that are specific to a particular domain (like the BACnet or KNX used for buildings management).
- The emerging generation of wireless networks, designed for the IP-enabled IoT environment, using IEEE 802.15.4, 6LoWPAN, RPL, etc.

We will show that gateways to both sets of DevNets can be improved, but that the improvement is much greater if the devices have embraced the IPSO concepts, which advocate the use of IPv6 in Smart Objects, for use in energy, consumer, healthcare and industrial applications.

IV. Basic Gateway Services

The IoT covers such a broad range of applications that it is very difficult to propose single solutions that cover them all. A typical exemplar IoT application that we consider in our prototype work is the monitoring and control of buildings. This domain might contain a number of subsystems that can be monitored and controlled, e.g.:

> *Lighting systems, Heating, ventilation and air-conditioning (HVAC) systems, Electricity supplies, Water supplies, Window blinds, Fire alarms, Computer systems, Door access systems, Personnel records (roles and badges), etc.*

For each of these there must be configuration profiles. While each has its own idiosyncrasies most will want to monitor its state or give a reading. Many will require some form of actuation. Many will have their own proprietary control system[3] – with its own set of addressing, security

[2]In the remaining text we may use the term *service* and *process* interchangeably. The rationale is that IoT services may be implemented in one or more processes *off the gateway host*.

[3]Hence, we distinguish the *gateway's* function from that of the *controller*.

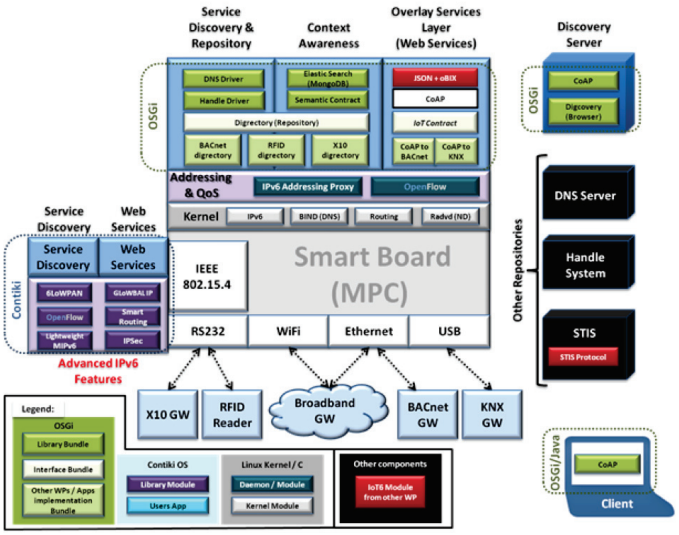

Figure 2. View of IoT system with integrated Application-Level Gateways.

and functions. Up to recently, the trend was to install a large number of unconnected systems. There has now been a further trend to have the sensor and actuators able to be controlled remotely, with the communications to the controllers being some sort of Internet technology – even if individual sensors or devices are not yet so addressable. While it used to be the practice to program the configuration of each controller separately, it is more common now to have a repository of device configuration profiles for each subsystem, and to have the capability of loading any profile remotely into the controller. Of course there would be a similar one of these for each subsystem. It is possible that the monitoring data from the different subsystems is brought together in one monitoring server. The servers may all be on the ServiceNet of Fig. 1, though some may even be on the general Internet.

When one looks at the sort of technology that goes into the controllers of the various subsystems in Fig. 1, there is now a trend to augment their functions, and to make them more like the *IoT6 gateway component* that appears in Fig. 2 (see the inner rectangle, labelled '*Smart Board, multi-protocol card*').

The gateway of Fig. 2 is quite complex. In practice it is an Application-Level Gateway, with translations occurring at most of its functional layers: networking, transport, application. When the need to provide secure access to the devices through this gateway is added, with role-based authorisation for individual operations, the complexity is clearly formidable. For wired controllers, it would even be difficult to replicate the device, and it might have to be maintained while in real-time operation. Moreover a different version of these would be necessary for each subsystem.

While all the functions of Fig. 2 are necessary, we are experimenting with a much more resilient version of such a system. At its most extreme, all the technology-dependent translations are carried out in **server processes**. These include the following:

(i) A *Building Management System* (BMS) that requests sensor data and sets devices according to its model of the different subsystems of the buildings. This includes

the set of entities that are requesting to carry out specific operations. This system will send messages requesting a sensor reading/actuator system to read/set the state of a specific set of devices

(ii) The BMS most commonly nowadays is a browser-based application, hence it might require to access the IoT subsystems via an *HTTP-CoAP proxy*.

(iii) A *resolver* that provides secured access and returns the location of the configurations of the set of devices. It also provides the locations of the server that can translate the read/write operations into the exact data stream(s) required by the relevant controller, and the location of the controller. Finally, it contains the Public Key certificates of the entities authorised to carry out all the possible operations.

(iv) One or more *repositories* that hold the configuration data of the devices

(v) One or more servers that can carry out the *address translation* processes needed by this set of devices.

(vi) An enhanced *controller* for each subsystem

A typical sequence of operations would be as follows.

- The BMS (i) will issue a series of secured messages, using web services over HTTPS to a resolver system (iii) like Handle.
- The resolver (iii) will check that the relevant operations are duly authorised. If the operations are authorised relevant configuration information locations (iv) will be returned.
- A set of secured messages encapsulated into CoAP will be sent by the BMS (i) (or its proxy (ii)) to the translators (v).
- The translators (v) will then send secured strings of data, in the exact syntax and semantics of the legacy device controllers to the gateway controller (vi). The controllers (vi) are enhanced only by including a CoAP module that incorporates DTLS. The reason for this last is discussed below.

V. Enhanced Gateway Functionality: A Distributed Protocol Stack

The previous section indicated a preliminary breakdown of core functions of a gateway (with controllers) to smaller components with specialised functionality. With the above basic IoT services all the technology-dependent operations are already being carried out as data streams with the current mode of working; exactly the same streams can be generated in processes upstream of the gateway, passed through as data via CoAP – just as the present generation of controllers expect.

With this observation in mind we summarise in Fig. 3 a typical generic IoT layering model, showing on the left-hand side available technologies and applications. Unlike the majority of previous work we identified in the literature, this layering is not constrained to one application domain. On the right-hand side we propose a potential protocol stack, with the functions of interest to us shown as smaller blocks in the various layers. These smaller blocks provide functions that are very often required, but need not all be co-located. Some could be distributed for execution as processes on servers elsewhere. We will refer to this stack as the **"distributed protocol stack"**.

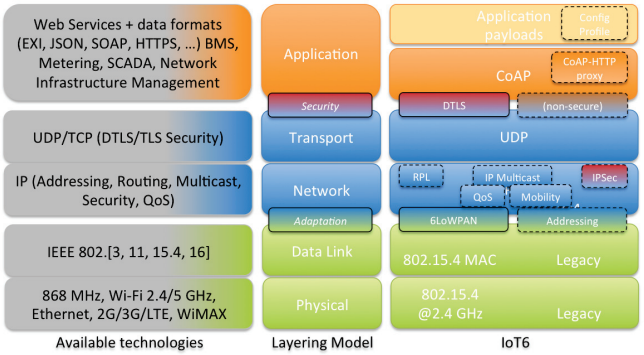

Figure 3. Typical IoT layering model and potential protocol stack.

The building management application normally adopts some abstraction of the building, which is quite technology independent; thus it may include the subsystems listed in Section IV (HVACs, lighting systems, temperature gauges etc.) with some abstract model of their configuration, but none of their technology. Just as in the single gateway case, a generic information model (description of methods and objects) tailored to buildings, like oBIX, may be used in a server process, with support from data representations like JSON.

In our enhanced architectural view, shown comprehensively in Fig. 4, this requesting application (i) will access gateway servers by web services using HTTPS, via a name resolution service like Handle (iii). This resolver already contains all the infrastructure required to authorise the transaction – including duplication protection against replay. If the transaction is not authorised, either a refusal will be returned or no action taken. However if the transaction is authorised, the response will include a one-time security token, signed by the resolver. At this level, the gateway process will access the resolver with information on the symbolic operation and the requester. The resolver will respond with the location where the proxy (ii) for that technology is carried out, and the repository where the configuration data is located (in our diagram the Configuration Manager is collocated with the resolver (iii), but other data may be configured in a separate repository (iv)). All the communication between processes from here-on can be via CoAP using DTLS (vi). The technology-dependent translation process will access the configuration repository for real address bindings – this may include the sort of addressing proxy (v) defined in [17]. The ensuing messages are then sent via the CoAP on the physical controller (vi) in the syntax and semantics it understands. The controller need only verify that the eventual data stream it has received includes the secure token from the name resolver; no other authorisation information is required. Of course in some application domains other considerations may require additional verification or audit trails. Successful actuation will be acknowledged signed by the controller private key. Requests for monitoring data will be sent to specified data repositories signed with the controller key.

In future work we will address comprehensively the advanced IoT services of Mobility and Group Communications. For completeness their processes are shown in the 'Translators Cluster' (v) of Fig. 4.

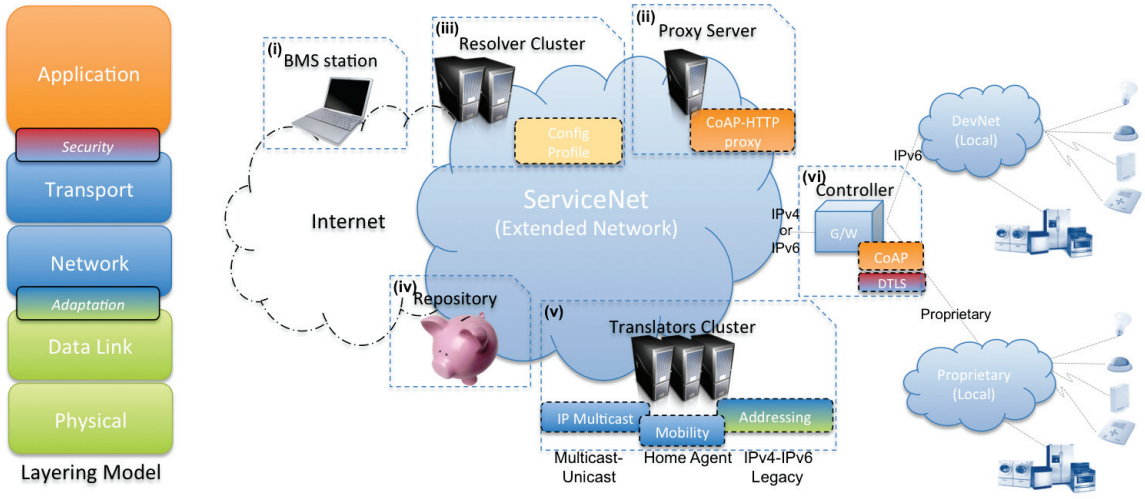

Figure 4. Schematic of a typical 'gateway' controller with a distributed protocol stack.

All the above has removed most of the technology-dependent extension out of the gateway controllers and into servers. It has also located the resource-hungry, and very tricky, security operations to a location where proper security assessment can be made. Moreover, this security operation is largely unrelated to the technology of IoT being employed. This has significant advantages. The added functionality of CoAP will not increase the processing and memory load on the controller. Moreover, the CoAP module itself will be a well-known and tried software module. Hence the probability of destabilising the supplier's controller is minimised. All the other processes are in servers that can be replicated. This permits all maintenance, trouble-shooting and upgrades to be carried out off-line. Depending on the type of processing and the quality of the networks, one can choose which operations should be carried out near the controller on the ServiceNet, and which can be carried out further up-stream – even in processor clouds.

VI. IoT Security

There are many aspects of security. We have discussed already in Section V how we propose to ensure the use of powerful security procedures without overloading the gateway controllers. Another aspect is the assurance that once the transaction has been authorised, the subsequent chain of transactions does not appreciably introduce new vulnerabilities – though they can introduce delay. The vulnerabilities caused by physical tampering with the IoT components are potentially serious, but are beyond the scope of this study. One of the places where CoAP provides only a subset of services is in its provisions for security. While HTTP allows for the use of many forms of security, CoAP has been defined only with the use of DTLS [19] over UDP, and that only with unicast links.

The fact that only unicast links are supported between a client and server does not proscribe the use of multicast addresses; it is just that only a unicast link is protected. If a multicast address is used, a multicast-to-unicast bridge can be provided as a translator service. However, the protocol protects for message integrity, confidentiality, replay, source/destination and some Denial of Service. Since Public/Private key protection is used, a gateway need accept messages only from a trusted source – indeed in our model this may be restricted to ones carry the security token signed by the resolver.

The impact of the above is that it will not require much resource to provide full security in a gateway. An up-stream server can provide full role-based checking on the authorisation for any operations; the gateway may then keep a cache of the public keys of the trusted servers. Only authorised monitoring or activation requests will be transmitted on to the gateway. Thus the gateway need merely check the integrity and source of the datagram; no further authorisation checking is required. This acts also as a partial protection from DoS attacks on the gateway. Any other messages received can be discarded.

Thus the burden of full authentication and authorisation has been passed up a chain where previous servers can provide full role-based authorisation requests with, if necessary, full audit trails. In some cases this security functionality can also be located in technology-dependent translation servers. Often, however, the outside request will come through a resource directory like Handle, which has built in this infrastructure for all its requests. The choice of these is application dependent.

Of course it will often be necessary for the devices themselves to have in-built *safety* mechanisms, to ensure that they will protect the subsystems from communications failure or other incorrect operations

VII. Technical Activity

All the gateway components, i.e. those of Fig. 2, have been integrated in the IoT6 project. Most are installed into a single-processor gateway. This gateway is expected to be able to provide a link into a number of different device controllers or even complete monitoring and control subsystems – that may even not be IP-enabled. Detailed results will be presented in a follow-on and associated publications [18]. Suffice to say that we have identified existing modules that provide all the functionality needed – either from modules in the single-node IoT6 gateway, or developments elsewhere.

While many existing name/location resolvers could be used, we evaluated the Handle system as being particularly

suitable for a "proof of concept" pilot. Its resolver is available locally as part of the global Handle infrastructure, and has been accessed experimentally from our system. Separate security modules exist in the different implementations of Handle (HTTPS) and Contiki (mostly as IPSec in the IP-enabled end-nodes). The use of oBIX/JSON to describe certain configurations has been completed with the gateway of Fig. 2 and specific Use Cases in IoT6.

We have re-constituted the integrated process of Fig. 2 into the new vision of Section V. We have validated a suitable DTLS module for Contiki to encompass all gateway operations as described. We have set up and validated an HTTP-CoAP proxy. In the next phase, we will be re-engineering some service modules (IP Addressing Proxy, a multicast bridge and a Home Agent for network mobility) running on the integrated gateway for some legacy technologies like X10 and/or KNX to now execute on a server process, on behalf of the gateway.

VIII. Conclusions - Future Work

We have outlined a generic approach to the development of IoT gateways, which is particularly relevant to one connecting in legacy automation systems. It should lead to a new architecture of "gateway processes", which can be better tailored to specific application domains to provide more maintainable systems that are operationally simpler. It has been shown that it can use state-of-the-art mechanisms for representing a specific domain of building automation, and incorporating one of the latest versatile operational object location resolvers. It has shown how it is possible to incorporate strong security into the IoT operations, without incurring unacceptable overhead onto the real-time components of the system.

There has been a temptation to build "universal gateways", meaning that the same hardware, and much of its software, can be used with different device controllers. In that concept, there is a significant modification of the gateway functionality for each different application domain and equipment. The approach suggested here is almost the opposite. Here we advocate adding a minimal upgrade of the normal operation of each technology system gateway – so that it can be used in any environment that is appropriate. The different monitoring and actuation processes are expressed in virtual form. The configuration information is also expressed in configuration profiles stored in systems that could be in local – but sometimes even more distant clouds. The secure binding, and the introduction of the technology-dependent aspects, is carried out in servers that can be replicated. The complex security operations are carried out in specific, potentially sophisticated, processes. Some of the more complex variants, like support for mobility and multicast, can be carried out also in such server-based processes. For legacy systems, this implies minimal change by the supplier for the equipment. For the more modern IP-based IoT sensor and actuator systems, the gateways may become as simplified over the current concept as the router gained over the application gateway in the early Internet. Exactly which segmentation of function is desirable – or even feasible – is application dependent. In the pilot system we are developing, we will gather data on comparison of response times due to unified and distributed gateway implementations. This should allow more informed decisions on the optimum deployment of gateway(s) in specific environments.

Acknowledgment

The authors would like to thank Vint Cerf and Bob Kahn for early discussions on the enhanced architecture and integration with Handle. This work has been supported by EC grant FP7-ICT-2011-7-288445 (IoT6).

References

[1] IoT6 European Project. "Universal Integration of the Internet of Things through an IPv6-based Service Oriented Architecture enabling heterogeneous components interoperability" (2012).

[2] KNX: The Worldwide standard for Home and Building Control, http://www.knx.org/knx/what-is-knx

[3] S. T. Bushby. "BACnet: a standard communication infrastructure for intelligent buildings." Automation in Construction 6.5 (1997): 529-540.

[4] X10 Industry Standard, http://en.wikipedia.org/wiki/X10_(industry_standard)

[5] Z. Shelby, K. Hartke, and C. Bormann. "Constrained application protocol (CoAP)" (2013).

[6] IEEE 802.15.4. "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf, 8 September 2006.

[7] Z. Shelby and C. Bormann. "6LoWPAN: The wireless embedded Internet". Vol. 43. Wiley, (2011).

[8] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur and R. Alexander. "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", IETF RFC6550.

[9] M. Kovatsch, S. Mayer and B. Ostermaier. "Moving Application Logic from the Firmware to the Cloud: Towards the Thin Server Architecture for the Internet of Things". 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 751 - 756. Palermo, 4-6 July 2012.

[10] Ericsson Labs, blog post by Jan Holler, 15/11/2012. "Having a headache using legacy IoT devices?" https://labs.ericsson.com/blog/having-a-headache-using-legacy-iot-devices.

[11] M. Jung, J. Weidinger, C. Reinisch, W. Kastner, C. Crettaz, A. Olivieri and Y. Bocchi. "A Transparent IPv6 Multi-protocol Gateway to Integrate Building Automation Systems in the Internet of Things". IEEE International Conference on Green Computing and Communications (GreenCom) 2012, pp. 225-233. Besancon, 20-23 Nov. 2012.

[12] D. Standeford, Washington Internet Daily, Vol 13, No 166. "IoT Naming System Said 'Critical' for Network of Connected Devices But Which One Unclear". http://www.cnri.reston.va.us/papers/wwid082712.pdf

[13] S. Sun, L. Lannom, and B. Boesch. "Handle system overview". RFC 3650, November, (2003).

[14] Z. Shelby and M. Vial. "CoRE Interfaces". Internet Draft draft-shelby-core-interfaces-05. 26/8/2013.

[15] Z. Shelby and C. Chauvenet. "The IPSO Application Framework". IPSO Alliance 'draft-ipso-app-framework-04', 24 August 2012. http://www.ipso-alliance.org/wp-content/media/draft-ipso-app-framework-04.pdf

[16] V. Cerf and P. Kirstein. "Gateways for the Internet of Things, An Old Problem Revisited". Accepted in Globecom 2013.

[17] A. J. Jara, P. Moreno-Sanchez, A. F. Skarmeta, S. Varakliotis and P. Kirstein. "IPv6 Addressing Proxy: Mapping Native Addressing from Legacy Technologies and Devices to the Internet of Things (IPv6)". Sensors 2013, 13, 6687-6712. doi:10.3390/s130506687

[18] S. Varakliotis and P. Kirstein. "A process-based Internet of Things". Technical report, October 2013. http://www.cs.ucl.ac.uk/staff/P.Kirstein/IoT_Korea_Long.pdf

[19] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC6347, January 2012.