Bridging the Devices with the Web Cloud:
A Restful Management Architecture over XMPP

Miguel Almeida[1] and Alfredo Matos[2]

[1] Nokia Siemens Networks
[2] Instituto de Telecomunicações
miguel.almeida@ua.pt, alfredo.matos@av.it.pt

Abstract. In this paper we deal with the interactions between different types of devices and a SaaS (Software as a Service) Management System. It is our goal to provide a generic way by which users interface with their devices in terms of getting information and actually being able to communicate with them. Our effort in this proposal is the establishment of these interactions while assuring a set of requirements such as privacy, authentication, association of multiple devices to a user, etc. We provide the architectural means to support this view and ensure the communication of the IoT (Internet of Things) devices with a Cloud of Web Services, while maintaining the M2M (Machine 2 Machine) vision. To do so we define an entity, the Cloud Bridge Server, which uses the Extensible Messaging and Presence Protocol (XMPP) to interact with the devices, and which provides a Representational State Transfer (REST) API for 3rd party Web Services. We present results on how our approach performs when facing other alternatives and the main advantages of using it.

Keywords: Reporting, Actuation, XMPP, REST, SaaS, Service Management.

1   Introduction

As we evolve towards the Future Internet (FI), we expect the number of devices to grow, along with the interactions between them. The widespread appearance of small and embedded devices drives the Internet of Things (IoT) as a branch of the FI. In such a scenario, being capable of adequately managing these devices implies a critical success factor, where scalable and secure interactions will pave the way for ubiquitous adoption of these paradigms.

It has become clear that information generated by the users' multiple devices and sensors, needs to be collected and evaluated in a scalable way. Many of these devices gather information that serves as input for future decisions, in which case, commands need to be sent back to the devices. We consider reporting and acting as the two important management components for emerging networks, especially considering the IoT paradigm in on Machine to Machine (M2M) context.

We face a lack of solutions that provide a scalable and integrated proposal for device management. These solutions do not take into account emerging requirements that stem from the FI and IoT, energy efficient environments, and the need to integrate with the omnipresent web environment.

The web component is translated into the sweeping trend of service oriented approaches, resulting in paradigms like Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS), which compose the Cloud. Such paradigms provide a greener approach by maximizing the resources' efficiency with scalable infrastructure, reusable platforms and distributed services that meet the end-user needs. It becomes apparent that a bridge must be established between the Cloud paradigms, embodied by SaaS and PaaS architectures, and the Future Internet, in the form of IoT. The Cloud requires a strong interaction between web services, and to make it a part of the FI, this cooperation must be extended to services which are not web oriented – the devices operating in a M2M environment as part of IoT.

Many ongoing efforts attempt to integrate both of these paradigms, but a seamless approach is yet to be accomplished. It must be assumed that any modern management architecture must cope with a cloud driven approach to enable reporting and acting. Therefore, it must integrate multiple domains and variable security environments, where the devices are user driven and might even exist on user premises.

In this paper we define a platform that interacts with different devices associated with users, supporting the M2M vision. This platform is presented as a twofold ecosystem where a cloud service interacts with an Extensible Messaging and Presence Protocol (XMPP) enabled device, through web like primitives. It provides asynchronous and secure reporting/acting via a well defined set of commands, maximizing resource usage on the device layer. This architecture is extended through a consistent Representational State Transfer (REST) interface that ensures the cloud integration by connecting a common web transaction model with the interfaces exposed by devices, managed within in the cloud.

By using a consistent addressing scheme based on the XMPP's Jabber ID (JID) and a bridge entity between the two defined environments, we attempt to address the problem caused by the complexity derived from the heterogeneity of terminals, sensors and networks, which is one of the requirements for the future of device management. Moreover, as each person uses several devices, we allow the possibility of associating devices to the users' identity while assuring a secure channel across domains.

Having in mind that a big part of the considered devices will feature power processing and autonomy constraints, we focus on a solution that does not require excessive resource consumption. We show that all the added value introduced with XMPP does not compromise performance, as discussed in Sec. 5.

The rest of the paper is organized as follows: Section 2 details the background and existing proposals in the literature. Section 3 details the XMPP driven reporting and acting architecture, exploring the requirements to allow exchange of information at the transport level. Section 4 shows how communication is established between the devices and the cloud, introducing the bridge between REST and XMPP and thus enabling the integration of the device information multiple web services in a SaaS approach. Section 6 concludes the paper.

2   Background and Requirements

When integrating devices with SaaS and PaaS oriented solutions, we must consider two main communication aspects: communication between the devices and the Cloud, and in-between the services within the Cloud. The chosen protocol which allows devices to interact with the cloud needs to consider several aspects, especially information transport.

For device interaction, we highlight the Simple Network Management Protocol (SNMP) [1], where information is collected from agents installed at the managed devices. The information can be polled on the device, and follows an explicit metadata scheme defined by the Management Information Bases (MIBs). When using SNMP to deal with the exchange of metrics and commands, we need to consider that while providing a performing environment, it lacks in authentication, identity management and any type of web integration, especially considering its binary oriented approach. Traps try to mend the lack of asynchronous core approach in SNMP, but only manage to do so in one direction, i.e., the client can report asynchronously, but to receive a command (or action as we call it) it needs to check the Network Management System (NMS) for information. Due to the aforementioned reasons we need to find a suitable transport alternative that fits the IoT approach, concerning resource efficiency (no pooling) aspects, and that, at the same time, allows a feasible integration with Web Services (WS).

The typical model for communicating with the cloud, or web service guideline, is based on REST interfaces over HTTP. REST provides a clear interaction model that enables powerful and flexible solution through simple interfaces, in a scalable environment. However, HTTP is synchronous (uses workarounds such as SMTP) and requires additional mechanisms to support authentication and identity management.

Integrating IoT devices and HTTP environments has been proposed over SOAP [2]. REST and SOAP are associated to HTTP as means to convey information that the cloud understands. SOAP does not provide any value beyond objects over HTTP. Having these objects transport reporting or acting information can be considered an orthogonal issue. In any case, SOAP faces a scalability issue because it usually requires a large amount of technology to establish bidirectional invocation: it usually requires an HTTP web server, coupled with an application server (e.g. Tomcat) to enable the WS environment. Moreover, current trends resort to REST over HTTP due to its simplicity and ease of usage given the mapping with of the HTTP methods (*GET, PUT, DELETE* and *POST*). It also provides a long-lasting interface that is not coupled with the business logic behind the interface. Where the WS approach has proven to be too complicated for mass scale deployment, we have seen REST Application Programming Interfaces (API) take hold, quickly extending to an increasing number of service providers (like Web 2.0 companies, such *Facebook* or *Twitter*) deploying SOAP-incompatible REST approach.

We then turn to a technology which has been slowly gaining traction, as it broadens its applicability domains. XMPP offers good conditions as a transport protocol for applications within the web services' scope since it offers reliability, synchronous and asynchronous deliver of messages and does not require a complexity of features such as WS-Routing and WS-Referral to ensure identity trace back [3] within private domains, since addressing is not only IP based.

XMPP was conceived as an alternative Instant Messaging protocol but has been evolving to a broader concept. Given the fact that it is open and XML based it became easy extendible and became an IETF standard. According to [4] there are three Core Stanza types defined by XMPP: The *<message/>*, *<presence/>* and *<iq/>*. The fist works as a push mechanism to immediately send messages if the destination is online. Presence relies on publish-subscribe mechanisms through which nodes inform the server of their availability (e.g.: *online*, *away*, *do not disturb*) and is usually distributed among the other nodes in the roster. The last one is a stanza responsible for entities making requests and receiving responses (hence Info/Query) from each other for management, feature negotiation and remote procedure call invocation.

One of the biggest advantages of XMPP is the fact that addresses can be associated with people or devices such as computers, mobile phones, sensors, routers or cellular network elements (3GPP RAN and Core Network Elements). This is achieved by the formation of the *JID* (Jabber ID), a uniquely addressable ID, which is a valid URI [5], that is created according to the following format: *person@domain/resource*, where *person* usually represents the user entity; *domain* represents the network gateway or "primary" server to which other entities connect for XML routing and data management capabilities; and *resource*, which is of special interest since it allows to identify a specific device associated with the *person*. Security can be achieved by using Transport Layer Security (TLS) for channel encryption, while authentication is achieved through Simple Authentication and Security Layer (SASL). Regarding the portability and interoperability requirements, XMPP uses the "*over-IP*" approach and allows the binding of resources to streams for network-addressing purposes. This feature also allows us to perform Identity Management via the relationships of the user and of the resource.

Another requirement we want to fulfill is the communication across multiple domains (e.g. across two operators). XMPP allows multi domain management using [6], while making use of server-to-server communication. It also allows the capabilities' exchange and location awareness features via the presence stanzas. Regarding the efficiency of the protocol, an important matter as m[entioned above, several activities are being conducted to improve XMPP performance, namely new lightweight version such as [7], but the major performance issues drive from the presence signaling which can be optimized, but is not the scope of this work. That concern made us consider that proposals like Soap over XMPP [8], would even enhance the performance concerns, since XMPP and SOAP are two XML based protocols, running one on top of the other. XMPP has been previously used in device connection, but only as a protocol capable on interconnecting sensors in an asynchronous wireless environment [9], falling short of the IoT potential it carries. We see a clear opportunity for XMPP and REST as IoT drivers, as explained in the following sections.

3   Managing Devices with XMPP

Future IoT environments will impose that networks deal with an increasing number of devices. It is crucial that the paradigms start shifting towards optimized security and event driven approaches for M2M scenarios.

We propose a framework using XMPP as the core technology for device management. Its base specification enables a plethora of features in terms of security, simple federation mechanisms, and considering its kernel use of XML, it enables a seamless integration path towards cloud technologies, one of the emerging networking paradigms. We use XMPP beyond its transport features, by leveraging the core components such as the JID and the XMPP federation model to build an IoT-aware environment that supports multiple devices.

3.1   User and Device Identification

XMPP provides the usage of a JID that uniquely identifies a user at a specified resource. The user ID is a unique identifier within the operator domain. Each user authenticates at its designated server, providing secure authentication towards the XMPP provider (see Figure 1 and Figure 2).

john.doe@operator.com/mobile-phone

UserID   Resource

Fig. 1. JID example

This creates a strong trust relationship between provider and user, even on devices which are not supplied by (and bound to) the operator. The JID can also convey the device identification, by specifying the resource, associated with the registered user.

By aligning the user identification in the JID with the user profile at the operator, it is easy to perform Authentication, Authorization and Accounting (AAA) in the operator structures, using a unique identifier, coupled with the appropriate resource information. Reporting user and device related information becomes simpler, due to the implicit identity linkage.
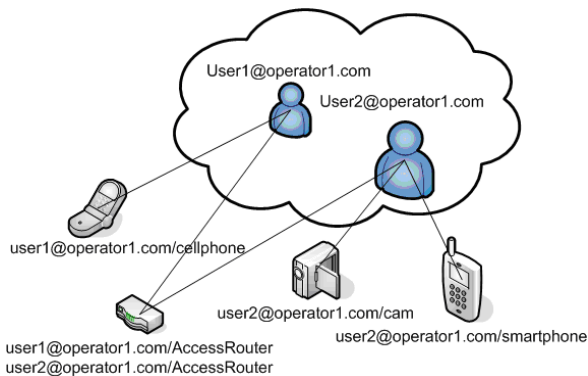

Fig. 2. Multiple Resources linked to one account

The described JID semantics constitutes a lightweight identity approach, providing a unique user identifier and associated authentication and authorization credentials. However, one of the key advantages of XMPP is that this user identification can be

easily mapped with Identity Management (IdM) architectures such as SAML 2.0 based frameworks, like Liberty Alliance [10], or other approaches such as OpendID [11], where the user identity is a URL (and the JID is a valid URL). The XMPP resource becomes part of the user identity, i.e. mapped to identity attributes, and a more secure framework, now tied to the XMPP identity and to the network's resources.

In this paper we focus on information aggregation as a function of the SaaS layer, by using the XMPP JID as the central structure, gathering information stemming from several user devices bound under one user profile.

3.2   Network Operation with XMPP

Within the XMPP domain of the proposed solution we must define two important steps: 1) how the connections are established between the network entities (devices and servers) and 2) how the information is exchanged between them.

Interacting with Devices

As we previously stated, we take advantage of the relationship between the users and their operator accounts. To support this, the following steps are taken:

1. User has a pre-enrolment phase with the Customer Service Provider (CSP): XMPP account pre-registered at the operator, with associated credentials;
2. User is authenticated at the CSP domain through the certified devices. Authentication with the provider is done through a secure TLS channel provided by XMPP, by which the user proves to be the owner of the XMPP identity;
3. User claims the resource: XMPP uses a lightweight mechanism to identify resources;
4. We request that the server provides a valid certificate establishing two-way authentication, establishing a scenario of mutual authentication, protecting the user's security and privacy;
5. Allow multiple concurrent XMPP connections for the same account, towards the same server, but with different resources, allowing concurrent usage of different devices, and using the secure and authenticated channels for the user information specified in the following section.

Information Exchange: Reports and Commands

Once the secure and authenticated communication channels are established, we can start the information exchange process over them. However, we first segment the nature of the information to understand what types of transport mechanisms (Stanzas) are necessary. The terminal collects the devices' characteristics, the *Performance Metrics* and the behavior *Actions*. The performance metrics are related with a multilayer analysis considering equipment performance, network performance and application performance, but given that we focus on the framework to convey these metrics rather than on the metrics themselves, we consider this component out of

scope since it has been covered extensively in existing literature [12]. Moreover, the actions of the user may also be of interest, which leads us to consider tracking that information as well (e.g. mobility or user interaction with applications).

As shown in Figure 3, the XMPP server, upon initial registration of the user, subscribes to all the events coming from the user. This will allow access to the reports stemming from the user. This information can later be relayed towards the cloud, particularly the Web Services that operate on a SaaS paradigm, as described in Section 5. This is done through a RESTful interface, where the operator's XMPP server acts as a proxy for the required information.
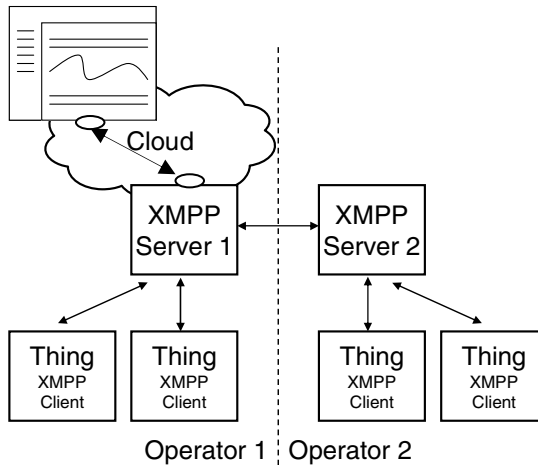

Fig. 3. Network Diagram

The choice of using the *message* stanza was taken considering that when comparing it to the *iq* stanza, it allows bare *JIDs* (user@host), while the *iq* requires full *JIDs*. Also the message stanzas can be sent later if the resource is offline thus ensuring liability [4].

We define the following additional Message Types in <message/> stanza: *Report* and *Command.*

Reports are sent from the devices to the XMPP server, where the final targets are the mentioned Cloud services. Similarly, *Commands* originate at the Cloud service towards the XMPP server, which conveys them to the appropriate device.

Then we specify the methods in the subject as shown in Table 1, ensuring Web Service integration. The usage of these methods can be extended as far as the Web Service understands them. Our XMPP server does not require any specific adaptation over this matter.

In the stanza *subject* we use the method invocation. It is worth mentioning that we re-use the *subject* field to convey the required operations so that the message if fully XMPP compliant, without the need do draft a XMPP extension Protocol (XEP). The field however, is fully understood at the *XMPP/Cloud Bridge Server (Cloud Bridge Server)*, which performs the necessary operations with the message. We purposely

align the method invocation towards HTTP primitives to clearly align the XMPP interactions with the REST interfaces that will be supported as part of the cloud integration discussed in Section 4.2, which specifies the interactions between the devices and the Cloud. We support the following methods, as specified by HTTP1.1: *GET, POST, PUT and DELETE*.

The *body* conveys an XML object indicating the name of the metrics and their values. The definition of this XML implies a creation of an Object Class with well defined Classes on the Service's Data Model. We refrain from detailing that specification in the scope of this work. By using this approach the XMPP Servers at the other domains are completely transparent and do not require additional extensions, since the stanzas are fully compliant with legacy XMPP.

Table 1. Example of a Message stanza with the Report type

```
<message
    to='romeo@example.net'
    from='juliet@example.com/balcony'
    type='Report'
    xml:lang='en'>
  <subject>PUT</subject>
  <body>910298301928301298310938</body>
</message>
```

3.3   Inter-domain with XMPP

Given the XMPP Federation support defined in XEP [6], communication between servers is tightly integrated into the base protocol. Through different federation levels, it enables secure communication, with full authentication when required, between different domains. This property allows users registered on different XMPP domains to communicate, while still retaining the security and authentication properties of a federated approach. Trust is preserved by using Certification Authorities (CA) and public certificates in the TLS exchanges, as defined in the base specification.

By using the described model, it is possible for nodes that operate in different domains, to relay information in a secure and trusted environment. This setup introduces a hierarchical concept that enables multiple operators to use the same cloud service, with different federation levels as explained in [6]. This view is presented in Figure 3 where, two federated domains, resort to the same Cloud service.

4   Bridging Devices into the Cloud

As services begin to shift into the cloud, SaaS is becoming the driver of a web-oriented architecture. Any solution that wishes to provide cloud interaction must take into account that, as mentioned before, SaaS is being built around REST interfaces.

A REST interface facilitates the interoperability of the management solution and SaaS services.

We leverage our XMPP based device management architecture to provide Cloud support through a REST interface towards the devices, sustained by the *Cloud Bridge Server*. The added infrastructure is fully aware of the M2M environment, and simultaneously collaborates with federated web services to achieve a cloud model, built around SaaS. We define a REST API that properly exposes the XMPP properties towards web services, facilitating cloud interaction.

4.1   Cloud Interaction

The *Cloud Bridge server* is a central point of the architecture. It provides the device management interfaces and functions for the registered devices. Therefore, given its administrative control over the devices, it exposes a REST API that enables interacting with the registered devices (Figure 4).

By functioning as the proxy or bridge to the M2M environment, it handles user authentication, which put it in a position to also aggregate the control over the exposed device information towards the cloud. The proposed exposure is done in a federated web services model, through the definition of the REST API that drives actuation and reporting towards the SaaS services.
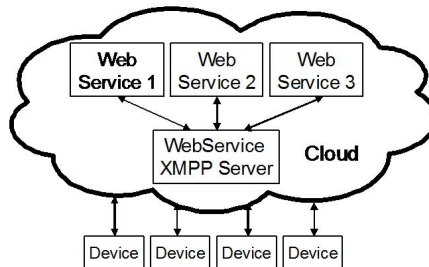

Fig. 4. inCloud Interactions

The architecture can be summarized as a two interface system: 1) the XMPP server exposes an interface that enables the 3rd party service to register as information receivers, and to act as command providers, always referencing the appropriate JID and going through the XMPP server's defined authentication mechanisms; 2) The web services expose an interface that allow information to be asynchronously supplied, and commands requested, when necessary due to network management operations. The two-way system allows outsourcing not only information storage, but also network control and management.

4.2   Aligning XMPP with REST Services

The alignment between the REST services and XMPP domain is assured in full by the REST/XMPP Bridge server. However, to assure a seamless integration we must assure that the naming, through the JID, is consistent in both domains and that there is

a secure control over the provided access in information flowing between the two domains.

The abovementioned scenario, where services interact seamlessly through the *Cloud Bridge Server,* is only possible because there is a direct re-usage of the JID on both XMPP and Cloud domain, which guarantees addressing consistency in both domains. However, the REST paradigm mandates a stateless interaction, where resources are properly identified. Even though the JID is already a referable URL, it requires the definition of a state to use at a HTTP server, given that it implies the concept of a user at a domain. To maintain compliancy with current web deployment trends, we use a transformation of the JID (the key identification mechanism in the XMPP domain) as the primary resource identifier in the web domain. The transformation is provided according to Figure 5.

john@doe.com/smartphones

doe.com/user/john/resource/smartphone

Fig. 5. JID Translation example

With this simple translation mechanism provided by the *Cloud Bridge Server*, we can guarantee the required consistency between the two domains, enabling the exposure of XMPP resources to the REST interface, which can now be fully controlled on a per-user/per-device policy determined by the bridge.

Given the fact that the JID is URI compliant we can use it to back trace the node of origin of the information. All messages are sent to the Cloud seamlessly, given that any web like environment can support RESTful primitives. Similarly, the selected stanzas require mapping between the two domains. Given the defined XMPP operations – GET, PUT, UPDATE, DELETE – which are 100% HTTP compliant, and the core of REST functionality, integration is assured with the JID transformation. This non-naïve selection of operations creates a perfectly aligned environment with minimal effort on the *Cloud Bridge Server*.

As the entry point towards the Cloud, the *Cloud Bridge Server* enables the communication between the devices and SaaS taking on a vital role for authentication and authorization. It provides the means for the devices to send reports and receive commands in a secure and trusted environment. We extend the properties stemming form the XMPP environment to the Cloud by forcing a federation environment between REST-enabled web services. Each service must register, define a SLA, and authenticate to gain access to information pertaining to the devices. Given the control over the information, the XMPP server is able to define a granular access control to the information exposed, and even to exert a parsimonious filter on the commands received from the SaaS services. Ultimately this decision can be left up to the device, by having the server convert the REST API call to an XMPP request, and forwarding it appropriately.

However, given that there is already a notion of identity on the *Cloud Bridge Server*, provided by XMPP through the JID, it can define any number of configuration on the access allowed between the identities' resources and the with the 3rd Party Web

Services. This requirement enables access control to the devices and their information on a *per service basis*.

This secure setup even allows customizing the devices towards a specific web service. If the devices are configured accordingly they can opt to send reports to the custom Web Service URLs. The XMPP signaling will transport the report in the message stanza to the XMPP Server and then the Web Service will perform a HTTP POST on the destination Web Service. The Web Services need to be aware of the data model being communicated. We do not intend to detail the Object Classes for each data type in this paper, as we simply want to cover the transport and communication procedures.

5   Discussion and Evaluation

We first start by comparing the different approaches in terms of feature support. Using Table 2 as a reference, we can see that for the purposes of integrating the devices with a web environment, SNMP is the most inadequate since HTTP based solutions are already web based (because they use HTTP and hence this analysis is Non Applicable for them) and XMPP is XML based which makes the transformation of the objects direct.

Regarding the security features, XMPP creates a secure unique channel using TLS an evolved solution of SSL. Earlier versions of SNMP present serious security constraints and that in fact has been an issue widely addressed in SNMP v3, but still the common applications use IPSec bellow the SNMP communication.

Table 2. Feature Comparison

Feature:	XMPP	HTTP+ REST	HTTP+ SOAP	SNMP
Security	TLS	SSL	SSL	IPSec
Reliability	YES	NO	NO	YES
Authentication	SASL	NO	NO	NO
P2P Support	YES	YES	YES	YES/Traps
Easy to Integrate with Web	YES	YES	YES	Can Be Done
Easy to Integrate with Operators	EASY	N/A	N/A	Complex
Identity Management	YES	NO	NO	NO
Bi-Directional Communication	YES	NO	NO	YES*
Overhead	HIGH	HIGH	HIGHEST	LOWER

SNMP also allows authentication to verify that the message is from a valid source, but we want to support authentication of an Identity and merge that with accounting information which is not possible. None of the other solutions also allow this. That brings us to the patter of Identity Management. In fact, XMPP is the best proposal to link several devices to someone's identification. And if such a platform is to be deployed at a Network Operator's site, then it would be good to allow the cross

matching of accountings with the operator's database, only feasible using the possibilities offered by XMPP.

Being able to support asynchronous communication allows the deployment of an important feature for fault management functions, which is sending alarms in a Near Real Time way. There are a lot of applications which take advantage from this possibility and XMPP is the best approach to deal which this type of events, also it allows bidirectional communication without the need to be running a web server in the devices, which is the only way to support bidirectional communication using HTTP with REST or with SOAP. Also XMPP allows P2P (and hence device to device) communication, which opens the possibility for nodes to self configure and exchange capabilities as well as performance information in future work.

In terms of overhead and performance comparison, we evaluated how the access links that connect the devices to the cloud would behave. As can be seen in Figure 6, SNMP outperforms the XMPP approaches. This happens mainly because in XMPP we defined an Object Class and used the objects within an XML to make the transactions which we measured. The results presented for the XMPP approach were obtained from experimental evaluation, while in the SNMP related results, we computed the overhead induced by raw data transportation, when conveying binary information. This information would require post processing at the NMS.
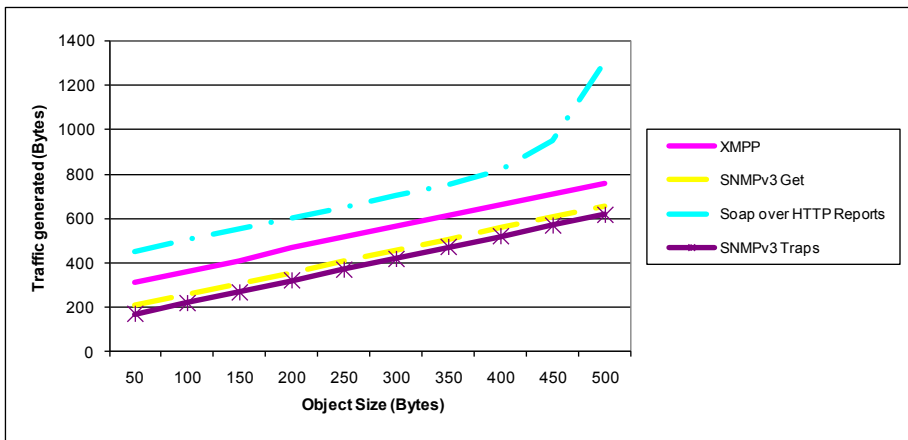

Fig. 6. Overhead and Traffic vs #Objects

In SOAP over HTTP, we must consider the overhead introduced by the signaling and also the headers of the IP, TCP (with timestamps), SOAP and, SOAP envelope. The calculated overhead was considered determined is presented in expression (1). As the object size increases, packet segmentation occurs at the TCP layer, thus increasing the size of generated traffic significantly.

$$Length_{Soap} = header_{IP} + header_{TCP} + header_{HTTP} + header_{Soap+envelope} + object_{size} . \qquad (1)$$

Taking [13], we define the header size of SNMPv2c as approximately 25 octets. The overhead of SNMPv3 is given by expression (2), where the Header Data of the SNMP is given by expression (3) as 17 octets. This means that SNMPv3 adds a minimum of

17 octets to SNMPv2c. Considering the signaling generated by both versions, the total generated traffic is given by expression (4) SNMPv3. SNMPv3 traffic was generated using ASN.1, and *Get* processes (*Request + Response* messages) were taken into account for overhead measurement purposes.

$$Traffic_{SNMPv3} = HeaderData + SecurityParameters + ScoopedPDU \ data \qquad (2)$$

$$HeaderData_{SNMPv3} = SNMPVersion + MSGID + MaxSize + Flags + Security_{Model} = 17 \ Octets \qquad (3)$$

$$TrafficSNMPv3 = 88 + n \ (10 + 2.Object_{length} + Value) \qquad (4)$$

When considering the trap-based approach, we determine the traffic of SNMP in expression (5) for SNMPv2. Then, we calculate the minimum traffic generated by SNMPv3 and add the additional minimum overhead (considering $Community_{size}$=6 octets and $Trap_{length}$=1 octet) to get expression (6) for the traffic generated by SNMPv3 traps (OID refers to Object Identifier). As can be seen in Figure 6 and Figure 7, Traps reduce the generated traffic especially when a many events are being generated.

$$Traffic_{TrapSNMPv2c} = 63 + Community_{size} + TrapOID_{size} + n.(3 + OID + Value) \qquad (5)$$

$$Traffic_{SNMP \ v3 \ Trap} = 87 + (3 + OID + Object_{size}).numObjects \qquad (6)$$

In Figure 7 we show the required bit rate in function of the number of nodes connected to a server.
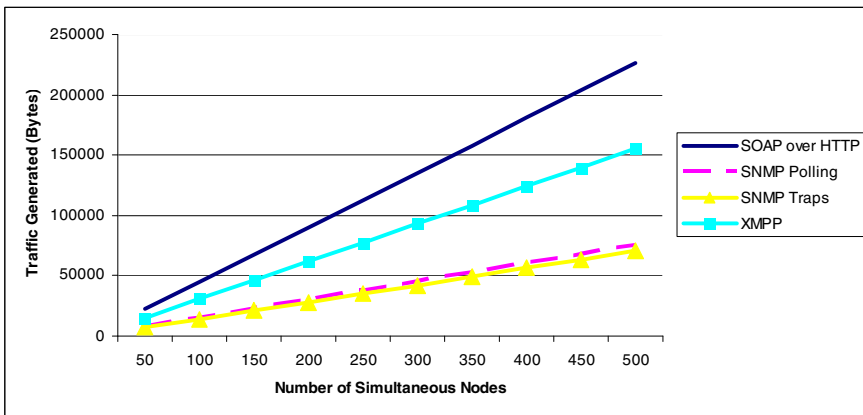

Fig. 7. Overhead and Traffic vs #Nodes

It represents a scalability analysis on the performance of the XMPP in terms of throughput required per number of nodes. We see that, once again, SNMP outperforms the XMPP approach, for the same reasons as explained above, but using SOAP over HTTP would still be much worse. However with this comparison, we want to show that the server's requirements will scale proportionally to the number of nodes.

Nevertheless the number of features supported by XMMP cannot be neglected and the ease of integrating the devices with a web environment makes it the most attractive solution. We consider that the weight of the overhead is greatly compensated by the provided features especially when considering Identity Management possibilities and authentication procedures. Also, we cannot neglect the flexibility allowed by the asynchronous and M2M oriented communication.

6   Conclusions

We have presented an architecture that addresses the integration of devices in a Cloud scenario, taking advantage of asynchronous and secure technology, such as XMPP, to drive a resource-maximizing solution, proven to be the greener approach in the long run. Through analysis of XMPP as a transport technology and as the core driver for interactions within the IoT domains, we consider it the best approach when envisioning the integration of end user devices such as terminals, gaming consoles, cell phones, IP enabled sensors, etc, with web environments and also with the operator's infrastructure. The number of features supported allows authentication using the account identification of the devices' owners within the operator's Charging Gateways, allowing the easy deployment of charging per usage. This, lightweight identity mechanisms can also be extended into Future Internet IdM scenarios, placing the proposed solution on the cutting edge of user-centric technologies.

The integration with the cloud environment provided through REST interfaces, allows the interaction with 3rd party web services, increasing the possibilities of applicability and revenue. By providing common and consistent interfaces to acting and reporting on devices within IoT, we enable a new array of business relationships and opportunities that put the telecom and infrastructure operator back in the driver seat of the network, while enabling a clear interaction with the Cloud world, a feature which has been profoundly lacking from the operators portfolio. We believe that these paradigms will be a key revenue system where both operators and service providers can capitalize by using the adequate tools, such as REST, XMPP and the *Cloud Bridge Server* to unite the common approaches.

We consider that the weight of the overhead, when compared to binary approaches such as SNMP, is greatly compensated by the features it provides especially when considering the strong security environment that enables a new level of trust borough onto M2M interactions. If we consider the Identity Management possibilities on top of enabled authentication procedures, then we face a new ecosystem will benefit all the parties in the network.

Also we cannot neglect the plethora of possibilities facilitated by the proposed asynchronous paradigms, and publish-subscribe mechanisms that will drive M2M interactions, with resource maximization concerns fro greener technology. These kinds of features represent the major requirements for solutions in the fields of eHealth, eEnergy, Domotics and lifestyle analysis.

Future work will focus on performance improvements, considering that better results can be achieved with a binary approach. By reusing the XMPP paradigms that include security through TLS, notifications, publication-subscription and inter-domain considerations, we are in a position to enable an efficient IoT architecture that fits the primary requirements of the Future Internet.

References

1. Case, J., et al.: A Simple Network Management Protocol (SNMP). RFC 1157 (May 1990)
2. Liu, L., Gaedke, M., Koppel, A.: M2M Interface: A Web Services-based Framework for Federated Enterprise Management. In: ICWS, pp. 767–774 (2005)
3. `http://xmpp.org/extensions/xep-0072.html`
4. IETF RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core, `http://www.ietf.org/rfc/rfc3920.txt`
5. Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396 (August 1998) (TXT, HTML, XML)
6. XSF XEP-0238: XMPP Protocol Flows for Inter-Domain Federation, `http://www.xmpp.org/extensions/xep-0238.html`
7. Hornsby, A., Bail, E.: μXMPP: Lightweight implementation for low power operating system Contiki. In: International Conference on Ultra Modern Telecommunications & Workshops, ICUMT 2009, October 12-14, pp. 1–5 (2009), doi:10.1109/ICUMT.2009.5345594
8. XEP-0072: SOAP Over XMPP
9. Hornsby, A., Belimpasakis, P., Defee, I.: XMPP-based wireless sensor network and its integration into the extended home environment. In: The 13th IEEE International Symposium on Consumer Electronics- ISCE 2009 (May 2009)
10. Project Liberty website, `http://www.projectliberty.org/`
11. OpenID website, `http://openid.net/`
12. Almeida, M., et al.: Cross layer design approach for performance evaluation of multimedia contents. In: IWCLD, Palma de Maiorca (July 2009)
13. de Lima, W.Q., et al.: Evaluating the Performance of SNMP and Web Services Notifications. In: IEEE/IFIP Network Operations and Management Symposium (April 2006)