A Distributed Web-based Naming System for Smart Buildings

Gérôme Bovet*‡ and Jean Hennebert†‡
*LTCI
Institut Mines-Telecom, Telecom ParisTech, 46 Rue Barrault, 75013 Paris, France
Email: gerome.bovet@telecom-paristech.fr
†DIUF
University of Fribourg, Bd de Pérolles 90, 1700 Fribourg, Switzerland
Email: jean.hennebert@unifr.ch
‡iCoSys
University of Applied Sciences Western Switzerland, Bd de Pérolles 80, 1700 Fribourg, Switzerland

***Abstract*—Nowadays, pervasive application scenarios relying on sensor networks are gaining momentum. The field of smart buildings is a promising playground where the use of sensors allows a reduction of the overall energy consumption. Most of current applications are using the classical DNS which is not suited for the Internet-of-Things because of requiring humans to get it working. From another perspective, Web technologies are pushing in sensor networks following the Web-of-Things paradigm advocating to use RESTful APIs for manipulating resources representing device capabilities. Being aware of these two observations, we propose to build on top of Web technologies leading to a novel naming system that is entirely autonomous. In this work, we describe the architecture supporting what can be called an autonomous Web-oriented naming system. As proof of concept, we simulate a rather large building and compare the behaviour of our approach to the legacy DNS and Multicast DNS (mDNS).**

I. Introduction

In the 21st century, rising energy costs and the impact of natural disasters due to climate change have made people realize that it is now necessary to better manage our energy consumption. In this context, buildings represent the main contributor by consuming up to 40% of the overall energy, which is even greater than industry or transportation [1]. Based on this observation, it is obvious that buildings account for a large energy saving potential. Building management systems (BMS) aim at reducing the energy consumption by optimizing the management of Heating, Ventilation and Air conditioning (HVAC), as well as lighting and other appliances. Following this direction, novel BMS approaches tend to migrate from a centralized management to a distributed one following the *Internet-of-Things* (IoT) paradigm, where agents shall coordinate between themselves [2]. In the foreseeable future, building management systems relying on a tremendous variety of sensors and actuators will democratize in private households, as for example the sen.se Mother and Cookies [3]. However, those systems will only be accepted by common people if they require no special IT knowledge, being easy installable in a plug-and-play manner.

Targeting a distributed and autonomous building management requires the establishment of an underlying compatible architecture, especially in IoT scenarios. Today's legacy network infrastructures rely on the DNS [4], which aim is to map IP addresses to human readable names. Although the DNS has been largely sufficient until now, its suffers from being a heavy infrastructure only deployable and manageable by expert people, that weakly supports the mobility and dynamism of everyday objects. This is especially true for smart buildings applications where pervasive devices can appear and disappear, as well as move inside the building. Additionally, the legacy DNS has to be set up before deploying those devices, which is preventing a plug-and-play installation. Although some distributed naming systems like mDNS [5] exist, they are not specifically tailored for the IoT nor for specific constraints of *smart buildings*. We believe that the future naming system for the IoT should offer solutions to following issues: *scalability, autonomy and fault tolerance, resource orientation, pervasiveness, backward compatibility and efficiency*.

In this paper, we present the design and architecture of a novel naming system at the convergence of the aforementioned requirements. We base our design following the principles induced by the *Web-of-Things* (WoT) paradigm [6] that is gaining momentum in sensor network applications. Everyday objects as well as sensors are exposing their capabilities in form of RESTful APIs inheriting and extending key properties of the Web, such as a strong interaction style allowing communications between various hard-/software platforms, a scalable architecture supporting billions of resources, as well as a total independence to network topology and medium. We now seek to exploit those benefits to provide a naming system that is loosely-coupled, and making an intensive use of Web technologies thus allowing a transparent integration in future Web-based building management systems. Since the purpose of the paper is to present our view of an autonomous naming architecture for Web-of-Things oriented networks, we will focus on its design guidelines and inherent concepts, rather than on its performance.

The remainder of this paper is organized as follows. Next section summarizes some related work. In Section III we provide a list of requirements that a naming system for smart buildings must fulfil. Section IV describes our proposed autonomous Web-based naming architecture, and details the zone decomposition that hierarchically organizes the name space. Its decomposition in RESTful APIs is showed in

Section V. The backward compatibility with the legacy DNS is presented in Section VI. In Section VII, we test our architecture by performing simulations with a prototype implementation. Finally, Section VIII concludes our paper and provides insights on further research.

II. Related Work

The traditional IP-based Domain Name System (DNS) which has been a key enabler of the Web is today often criticized, although numerous enhancements have been proposed to face its shortcomings. One of these enhancements is the Multicast DNS [5], widespread in IoT applications, which fully distributes name servers among the network, relying on a multicast group. Contrary to the DNS which is hierarchically organized in zones, the mDNS works with a flat name space where each server can hold records for any domain. While the approach is fully distributed, it does not ensure that there is no single point of failure as records may not be duplicated on other servers. Additionally, it requires a strong mechanism for detecting record conflicts. In the same way as for mDNS, MOSS [7] proposes a distributed architecture using a flat naming that has to be defined prior to deployment as each domain is manually associated with a predefined multicast group. Before sending a query, the resolver checks its table of domains to retrieve the associated multicast group. This has the advantage to only target servers that have entries for the requested domain. However, MOSS makes the assumption that every device is acting as a resolver and server for its own entry, which is possibly not feasible on constrained devices. Meanwhile this eliminates the need for a replication system as a name entry is managed by its own device. If a resolver does not receive a response means that the concerned device is currently down.

Partially distributed systems are like the classical DNS approach decomposing the name space into zones regrouping contiguous domains. Hosts can register and update their naming information (name and IP address) at the server managing their zone. This decomposition allows each zone working separately from each other in case of a link break between them. This concept is applied in [8] to a military network where the zones are decomposed according to the units of action of the troop. A master and a slave for each zone are synchronizing over DNS zone transfers for ensuring replication. An hybrid naming systems taking benefit of fully and partially distributed named is proposed in [9]. The system automatically adapts its working according to the underlying ad hoc network topology by choosing to work either with multicast groups in fully distributed mode or with unicast requests sent to zone servers.

Peer-to-peer (P2P) networks are gaining momentum in areas related to naming. Virtual overlay networks of DNS servers are composed over a structured or unstructured P2P [10]. Structured P2P approaches offer the advantage to have a predictable performance due to the fact that the overlay topology and the placement of the resources is controlled. The classical flat P2P has since been extended to hierarchical P2P where super-peers are elected among peers to form the overlay network [11]. Each super-peer has the capacity to regulate how many ordinary peers can connect to them before promoting additional super-peers. This concept is reused in our approach as it allows to dynamically react to load variations and to restrict the number of name servers to the minimum, thus reducing the need for synchronization.

III. Naming Requirements in Smart Buildings

The context of smart buildings is an interesting playground for IoT applications. The field of building automation is currently moving towards integrating some key concepts of the IoT, especially with Wireless Sensor Networks (WSN). New pervasive IP-enabled devices are acting as sensors or actuators. While the IoT proposes several technologies for allowing devices talking to each other from a network point of view, it lacks of standardization for the upper layers, especially regarding the naming. In this section we analyze the different issues that a naming system for smart buildings must meet.

1) **Scalability:** Office and factory buildings can be endowed with thousands of devices. If we push our vision further looking at future smart cities, millions of devices will sense and interact with the city. The underlying naming system must thereby be highly scalable and support an unlimited number of devices. In addition, the number of devices should only little affect the performance of the system.
2) **Autonomy and fault tolerance:** One major drawback of the DNS is that it has to be installed and maintained by professional people with expert knowledge. This drawback is one of the reason why it can not be considered for IoT applications. An automation system will only be accepted by users if their contribution for installation is reduced to the minimum, targeting a plug-and-play approach. This means that users should not have to set up a DNS server and zone delegations as part of the installation. The naming system should configure itself and being autonomous regarding the variations in size of the network. Additionally, the naming system should be insensitive to single points of failure, meaning that the lost of servers is automatically compensated.
3) **Resource orientation:** We can identify a clear trend of pushing RESTful APIs down to field devices in smart homes and smart buildings [12]. According to the Web-of-Things, each device exposes its capabilities as REST services in form of Web resources, thus standardizing the application layer. Existing classical building automation networks as KNX and EnOcean working with specific stacks have already been made compatible with the WoT by mapping their devices to RESTful APIs [13] [14]. By following this trend, there is only a small step from considering a Web-based naming system, exposing name entries as Web resources.
4) **Pervasiveness:** With the advances made in pervasive computing, sensing devices are becoming smaller and affordable. The limited available computational power limits the tasks that can be run as well as the protocols that can be understood by those devices. As consequence of this, the naming system has to be lightweight and compatible with constrained devices having only little memory and CPU.
5) **Backward compatibility:** We target our naming system to be compatible with the legacy DNS. As

the legacy DNS is on the heart of the Internet and the current Web, dissociating our architecture from it would limit the interaction with IoT devices as data and automation algorithms are traditionally hosted on dedicated computer or in the cloud. Our proposed smart building naming system should therefore allow a transparent compatibility of requests issued from the legacy DNS or from the smart building's naming system.

6) **Efficiency:** Finally, working in the context of smart buildings targeting energy savings, the underlying "smart" system has to be itself optimized in terms of energy consumption. The naming system should therefore limit the number of messages it needs for synchronization and data replication.

IV. The Distributed Web-based Naming System Architecture

In this section we illustrate the architecture of our autonomous Web-oriented naming system for smart buildings. The design choices and underlying technologies have been selected according to the aforementioned requirements. It basically relies on the concepts of partially distributed naming and P2P where we dynamically create zones that will be managed by some sort of super-peers. In our vision, sensors and everyday objects having enough computational capacities can run server agents, which avoids the need for dedicated name servers. Our reference implementation, although not necessary finding the best solution ensures finding an acceptable one within a defined time.

A. Entities

In the same way as for the legacy DNS, the name space is divided in domains and zones. *Domains* represent a logical entity of the name tree, and especially in our case a physical part of the building. We motivate to use a name tree according to the buildings floor plan because of the logical structure that can be found in buildings (i.e. building ->floor ->room ->device) matching with humans representation, as illustrated in Figure 1.

Those domains can be grouped in *zones* forming a virtual entity of contiguous parts of the domain tree, that are usually managed by a master and a slave server residing in the zone. In our approach a zone represents the grouping of near distance rooms. We here rely on the functioning of distributed automation systems where rooms form independent entities that can run disconnected from each other. This allows to ensure that the automation of a room will continue working even if the backbone network of the building suffers from disruptions.

Server agents have the possibility to act as resolver and to be promoted either to master or slave of a zone in which they reside. *Master* servers differ from slaves in that they can decide whenever their own zone should be partitioned or a child zone should be merged with its own. *Slave* servers are continuously synchronized with their respective master in order to ensure an up-to-date replication of data.

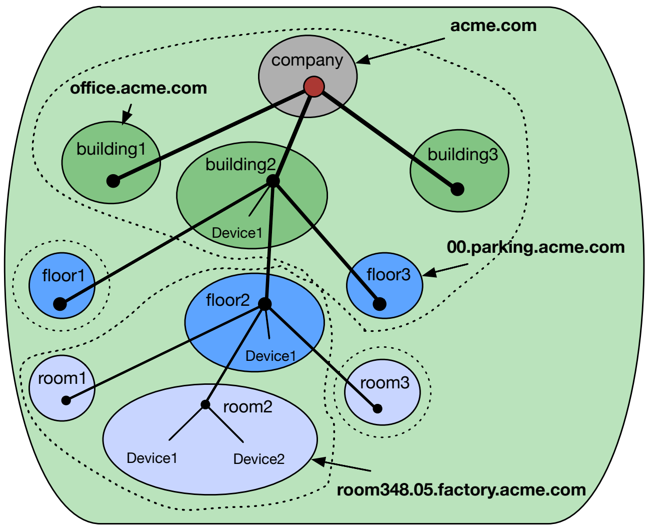


Fig. 1. Naming tree decomposition according to the buildings room plans and composition of zones

B. Zone Composition Strategy

The number of zones influences the availability and scalability of the network. Having more zones increases the need for higher bandwidth for supporting the synchronization between those zones and the replication of data between master and slave servers. However, this approach allows to decouple rooms. Fewer zones will allow to lessen the synchronization between parent and children zones, and require less servers for managing the name space. This choice is left to the developer. Meanwhile, our algorithm tries reducing the number of zones in order to minimize network traffic. The partitioning of the tree and thus the number of zones depends on several variables which are: the maximum number of name entries for a zone and the number of desired slave servers. While a brute force approach would result in having the best decomposition of a tree, its CPU and memory requirements would be too high for constrained devices. We here explain the most important steps of our algorithm that are also depicted in Figure 2:

1) In the first step, we ensure that all leaves of the tree have at least two servers. The algorithm will recursively merge leaves with there parent until each zone holds at least two servers.
2) Once each leaf has the required number of servers, the algorithm will merge non-leaf zones with there children so that each one holds at least two servers and does not exceed the maximum number of entries. We minimize the number of zones by trying every possibility to merge non-leaves with there children. At the end we obtain a new tree where each zone will be managed separately.

This process of repartitioning a zone is performed in two distinct cases:

- A zone exceeds the maximum number of entries it can handle. The master of the zone will partition its own zone and delegate the management of new ones to other servers. This operation is resulting in at least

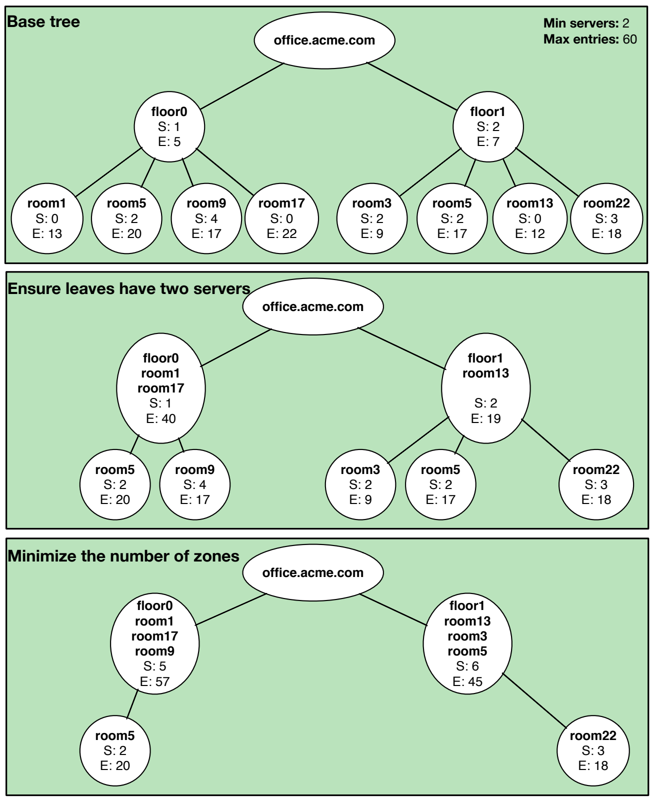


Fig. 2. Partitioning of a name tree with following constraints: minimum 2 servers and maximum 60 entries per zone.

one new zone.

- A zone can be merged with one of its children when the sum of entries of both zones is lower than the maximum it can handle. In this scenario the parent will repatriate the child's zone and merge it with its own. This operation is resulting in one zone less.

C. Name Server Selection Strategy

As indicated above, the name servers are not manually configured like it is the case for the legacy DNS, but are elected among server agents. This largely contributes to the autonomy of the naming system as there is no need for a human in the loop. Each zone is managed autonomously and has to ensure that it has at least one master and one slave name server, thus ensuring replication. Server agents can either promote themselves to name servers or be designated by others depending on certain conditions. A server agent always starts up with no specific role. His first step is to inquire for a name server managing the domain it resides in. If it receives no response, it will then promote itself as master for his domain.

There are also other ways for a server agent to become a name server:

- The server agent is promoted to a slave of the zone by the master. This ensures that the zone has always at least one slave if possible.
- A slave server promotes itself as master as soon as it detects that the master is no more responding and will then designate a new slave. This ensures that there is always a server that has the possibility to repartition the zone with at least one slave for replication.
- After the partitioning of a zone, new masters will be designated for the resulting new zones. Each new master has then the responsibility for designating slaves.

The criteria when promoting server agents to slave name servers can influence the overall performance of the naming system, especially for ad-hoc wireless networks. Two main criteria are relevant when selecting new servers: the closeness to the master for slaves and the number of servers of the domain in which they reside for new masters. For the first one, the locations of the master and the slave will influence the costs for replication and the ability to overcome connection breaks. While close located master and slaves (numbers of vertices in the name tree) allows to minimize replication costs, it induces some potential difficulties if the rest of the zone gets disconnected from the servers. Meanwhile we recommend choosing a slave server that is the closest to the master server as we want to reduce the communication costs. Another argument advocating to choose servers based on their closeness is that if one part of the zone is disconnected from the servers, other server agents will temporarily promote themselves as name server until the connection is recovered. Regarding the selection of new masters after partitioning, we put more importance on selecting a name server located in a domain having the highest number of server agents. We motivate this choice in order to comply with the aforementioned criteria, as there will be more probability to select a slave residing in the same domain and thus optimizing the replication costs.

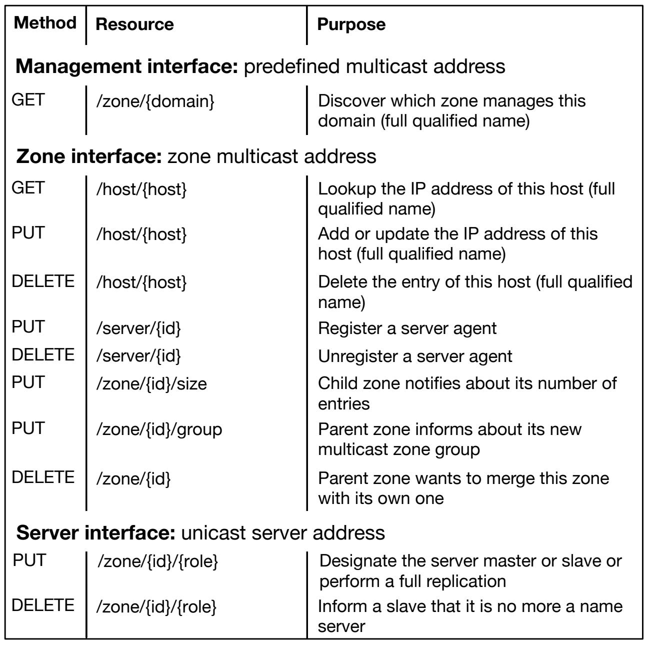
D. Data Replication Strategy

Data replication on at least one slave is mandatory for avoiding the single point of failure in case of the master failing or disconnecting. Data replication can be seen from several points of view. In the legacy DNS, the zones are periodically exchanged which creates a delta if the master fails. This delta can be reduced by replicating zones on a very regular basis, although increasing communication costs. In mDNS, data replication is ensured through the multicast messages that are processed by each host and added to their knowledge. This has the advantage to avoid full replication of zones between hosts. However, some information could be lost if a host misses a multicast message, and would result in a lost of knowledge.

In our architecture we aim at combining the strengths of both approaches. Master name servers will periodically fully replicate zones on their slaves, depending on a configurable value which can be time based or a number of updates. In order to lessen as much as possible full replications and the related communications costs, we propose to associate a different multicast group to each zone. This multicast group is used by hosts for updating records (i.e. announcement, update and deletion) and making lookups instead of having an unicast communication with the master. Following such an approach allows to bring two key foundations of our architecture regarding data replication and autonomy. First, one multicast message will have as effect that both the master and slave will update their records simultaneously, thereby reducing full replications. Secondly and not less important, using a multicast group for lookups and record updates is decoupling the zone's management from its physical name servers. Indeed hosts have no clue with which physical server they exchange as they communicate over multicast. In the case of a master name server failing and being replaced by a slave will have no influence on the hosts as they do not communicate directly with servers.

V. The Web-Oriented Naming System

In this section, we present our naming system relying on common Web technologies for managing the naming system. As previously indicated, our naming system has to be easily integrable in existing Web architectures. This is the reason why our protocol is based on RESTful APIs. In order to limit the amount of exchanged data, we opted for CoAP [15] being much more lightweight than HTTP. More specifically, we rely on the CoAP group communication for proposing APIs working over multicast [16]. Regarding the payload format, JSON outperforms XML because of requiring less memory, processing time and having less overload [17]. In addition, JSON has the advantage to be natively supported by all modern Web browsers. Following the paradigm of the Web-of-Things, we represent domains, zones and name entries as Web resources offering CRUD (Create, Read, Update and Delete) interaction.



Method	Resource	Purpose
Management interface: predefined multicast address		
GET	/zone/{domain}	Discover which zone manages this domain (full qualified name)
Zone interface: zone multicast address		
GET	/host/{host}	Lookup the IP address of this host (full qualified name)
PUT	/host/{host}	Add or update the IP address of this host (full qualified name)
DELETE	/host/{host}	Delete the entry of this host (full qualified name)
PUT	/server/{id}	Register a server agent
DELETE	/server/{id}	Unregister a server agent
PUT	/zone/{id}/size	Child zone notifies about its number of entries
PUT	/zone/{id}/group	Parent zone informs about its new multicast zone group
DELETE	/zone/{id}	Parent zone wants to merge this zone with its own one
Server interface: unicast server address		
PUT	/zone/{id}/{role}	Designate the server master or slave or perform a full replication
DELETE	/zone/{id}/{role}	Inform a slave that it is no more a name server

Fig. 3. APIs of the Web-oriented naming system (without payload and response information)

We decompose the RESTful API in three distinct interfaces depending on the scope of those, as visible in Figure 3. The *Management interface* serves as entry point where server agents and hosts can discover which zone manages a specific domain. Once knowing the zone's multicast group, the communication is switched to the *Zone interface* where hosts can announce themselves and perform lookups. Finally, the unicast *Server interface* offers services for replicating zones as well as for designating new master and slave name servers. Because of space constraints the JSON payloads are not showed in this paper.

The process of a lookup is decomposed in two parts. First, the resolver needs to discover which zone is managing the sought domain over the Management interface. The obtained response can then be cached by the resolver in order to skip this step for future requests. The resolver then performs a lookup of the host over the Zone interface and receives the response via unicast by the master of the zone. To summarize, at most two multicast messages are required to obtain the IP address of a host, which is less than the legacy DNS if the preconfigured server is not authoritative for the sought domain.

VI. Ensuring Compatibility with the Legacy DNS

Ensuring a transparent compatibility with the legacy DNS is a constraint that can not be overstepped. However, our proposed autonomous naming system relying on Web technologies is naturally not compatible with the DNS. This can be bypassed by introducing a gateway that will interconnect both worlds. From the classic Internet point of view, it will act as a standard DNS server being authoritative for the domains that are managed by our naming system. It will transparently forward requests to our naming system using the Management and Zone interfaces, sending back the response over the standard DNS protocol. On the other side the gateway will act as master server for domains that are not managed by our naming system, forwarding the queries to legacy DNS servers. Both modes are illustrated in Figure 4.

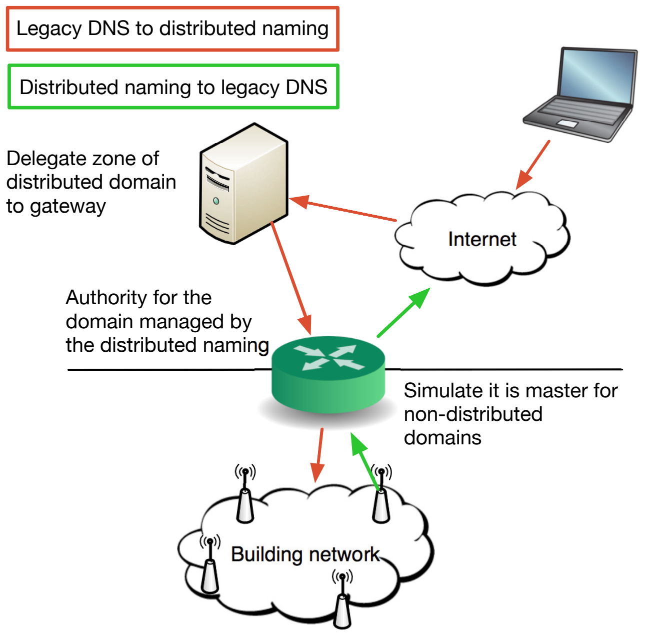

Legacy DNS to distributed naming
Distributed naming to legacy DNS
Delegate zone of distributed domain to gateway
Internet
Authority for the domain managed by the distributed naming
Simulate it is master for non-distributed domains
Building network

Fig. 4. Flow of messages between the legacy DNS and distributed naming

VII. Evaluation

In order to evaluate our Web-based naming system, we developed the server and client agents in Java. We chose as simulation playground three buildings of the College of Engineering in Fribourg, Switzerland. The buildings are composed of office, meeting and class rooms spread over five floors per building, giving a total of 180 rooms. Our simulation model randomly makes hosts and servers appear and disappear according to predefined ranges for each room type. Each repeatable simulation generates a total of 5000 operations which are of following type: add host, remove host, add server agent and remove server agent. For each simulation, we varied the maximum size of zones and observed the number of packets also as the data size (UDP + CoAP + payload) due to data replication. From the Figure 5 we can see that the number of packets and the amount of data are closely related. However, defining small zones will have as effect a more important number of zones and thus a substantial amount of synchronization data. However, both metrics tend to become steady as soon as the zone size can cover at least two rooms, which is approximately 45 hosts in our simulation. More generally we observe that in our situation 5.6kB in average are needed for one replication, while the total number of exchanged data is 608kB in the best case.

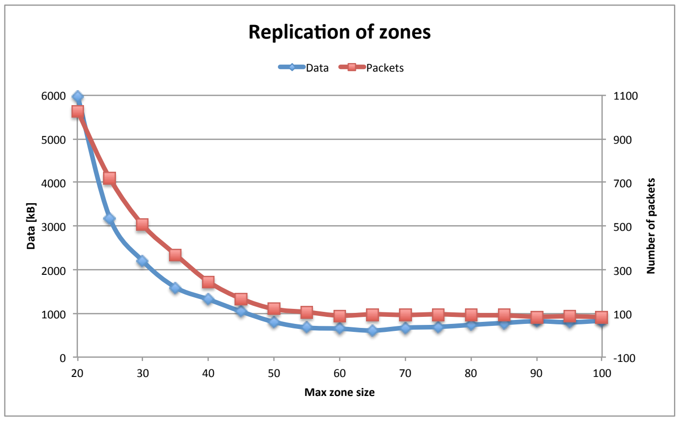


Fig. 5. Flow of messages between the legacy DNS and distributed naming

Compared to the legacy DNS and mDNS, our naming system outperforms in terms of number of packets and exchanged data, thus being more efficient. We performed the same simulations using legacy DNS servers as proposed in [8] and using mDNS [5]. Both approaches gave us a higher value of total exchanged data for replication, namely 1220kB (+100%) for DNS and 1040kB (+71%) for mDNS. This is explained by the DNS protocol itself having a relative consequent overhead compared to the combination of REST and JSON.

VIII. Conclusion and Future Works

It is likely that Web technologies will gain more importance in pervasive application scenarios and especially in smart buildings. They indeed allow to standardize from an application layer point of view the interaction mechanism between various devices. The underlying naming system can largely benefit from taking a Web approach representing domains, zones and host entries as Web resources thus removing the need for constrained devices implementing the DNS protocol. Even more important, naming systems must now evolve to self-organizing services in contrast with the legacy DNS requiring professionals throughout its operation, therefore preventing a deployment in private households.

In this paper, we showed our vision of an autonomous and Web-oriented naming system applied to sensor networks with a particular emphasis on smart buildings. Unlike mDNS, we keep the hierarchical organization of the legacy DNS while taking part of multicast for reducing the amount of data exchanged during replications. Our naming system requiring no human in the loop for deployment or maintenance makes a step towards fully plug-and-play sensor networks, which will largely contribute to their acceptance by people without IT knowledge. Future developments intend to even more reduce the replication data by compressing the JSON payload by using a shared schema. The security aspect is also an important concern that has to be solved in the near future.

Acknowledgement

The authors are grateful to the Swiss Hasler Foundation and to the RCSO grants from the HES-SO financing our research in this exciting area of smart buildings.

References

[1] L. Perez-Lombard, J. Ortiz, and C. Pout, "A review on buildings energy consumption information," *Energy and Buildings*, vol. 40, pp. 394–398, 2008.

[2] S. Abras, S. Ploix, and S. Pesty, "A multi-agent home automation system for power management," *Informatics in Control Automation*, 2008.

[3] "sen.se mother and cookies," https://sen.se/.

[4] P. Mockapetris, "Domain names - implementation and specifications," RFC 1035, 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1035.txt

[5] S. Cheshire and M. Krochmal, "Multicast dns," RFC 6762, 2013. [Online]. Available: http://tools.ietf.org/html/rfc6762

[6] D. Guinard, "A web of things application architecture - integrating the real world into the web," Ph.D. dissertation, ETHZ, 2011.

[7] Y. Gottlieb, R. Chadha, and K. Cheng, "Moss: Gathering names in networks of mobile nodes," in *Proc. of the 2007 Military Communications Conference (MILCOM 2007))*, 2007.

[8] R. Morera and A. McAuley, "Adapting dns to danamic ad hoc networks," in *Proc. of the 2005 Military Communications Conference (MILCOM 2005))*, 2005.

[9] M. Nazeeruddin, G. Parr, and B. Scotney, "An efficient and robust name resolution protocol for dynamic manets," *Ad Hoc Networks*, 2010.

[10] L. Huang, "Virgo p2p based distributed dns framework for ipv6 network," in *Proc. of the 4th International Conference on Networked Computing and Advanced Information Management*, 2008.

[11] R. Fahra and A. Leon-Garcia, "Peer-to-peer naming architecture for integrated wireline/wireless networks," in *Proc. of the IEEE GLOBECOM 2007*, 2007.

[12] A. Kamilaris and V. T. andA. Pitsillides, "The smart home meets the web of things," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 7, pp. 145–154, 2010.

[13] G. Bovet and J. Hennebert, "A web-of-things gateway for knx networks," in *Proc. of the European Conference on Smart Objects, Systems and Technologies (Smart SysTech 2013)*, 2013.

[14] ——, "Offering web-of-things connectivity to building networks," in *Proc. of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013)*, 2013.

[15] Z. Shelby, K. Hartke, and C. Bormann, "Constrained application protocol (coap)," draft-ietf-core-coap-18, 2013. [Online]. Available: http://tools.ietf.org/html/draft-ietf-core-coap-18

[16] A. Rahman and E. Dijk, "Group communication for coap," draft-ietf-core-groupcomm-18, 2013. [Online]. Available: http://tools.ietf.org/html/draft-ietf-core-groupcomm-18

[17] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, "Comparison of json and xml data interchange formats: A case study," *CAINE*, 2009.