Exploring the Use of DNS as a Search Engine for the Web of Things

Andreas Kamilaris, Koula Papakonstantinou and Andreas Pitsillides
Networks Research Laboratory, Department of Computer Science
University of Cyprus, Nicosia, Cyprus
Email: kami@cs.ucy.ac.cy, koula.papakonstantinou@gmail.com, andreas.pitsillides@ucy.ac.cy

Abstract—Sensor technology is becoming pervasive in our everyday lives, measuring the real world around us. The Internet of Things enables sensor devices to become active citizens of the Internet, while the Web of Things envisions interoperability between these devices and their services. An important problem remains the need for discovering these devices and services globally, ad hoc in real-time, within acceptable time delays. Attempting to solve this problem using the existing Internet infrastructure, we explore the exploitation of the Domain Name System (DNS) as a scalable and ubiquitous directory mechanism for embedded devices. We examine the feasibility of this approach by performing a simulation involving up to one million embedded devices, to test system performance and scalability. Finally, we discuss practical issues and the overall potential of this approach.

Keywords—Service Discovery, Search Engine, Web of Things, Domain Name System, DNS, Environmental Services, Embedded Devices, Sensors.

I. INTRODUCTION

Embedded computing is becoming ubiquitous in our lives. Tiny sensor devices around the world are measuring the physical environment continuously with high precision, recording ambient conditions such as wind, dust, radiation etc.

Lately, the Internet is penetrating slowly-slowly into the real world of physical objects. The introduction of IPv6 and the efforts for porting the IP stack on embedded devices [1], enable the vision of a global Internet of Things (IoT) [2].

It is already happening that physical devices exceed the human population on the Internet. Cisco predicts that by 2015, 25 billion devices will be connected to the Internet, and 50 billion devices by 2020\(^1\).

While the IoT enables communication between heterogeneous devices at the network layer, the Web of Things (WoT) envisions interoperability at the application layer, by reusing well-accepted and understood Web standards [3], [4]. Thus, embedded sensors are becoming fully integrated to the Web, directly by embedding Web servers on them [5] or indirectly by means of gateways [6].

An increasing number of governments, private companies and organizations, aiming to offer real-world services to the public, tend to expose the capabilities of their sensor devices as open Web API, in order to become easily re usable by developers and end users. These Web API are better structured and understood when they conform to the REST architectural style [7], [8], which is a core element of the WoT.

In general, there do not exist yet standardized, scalable and flexible ways to globally discover Internet-connected embedded devices, based on their characteristics and capabilities. Some early efforts either require additional infrastructure or they do not scale for the World Wide Web. We believe that service discovery of sensor devices needs to be ubiquitous to the users of the Web. The proposed solution must comply with existing Internet standards and should not require major changes to the existing technical equipment and protocols. Users should be able to discover environmental services simply by typing related keywords in their favorite Web browser. In this way, discovery of physical devices may be similar to the way we discover Web sites through search engines.

In this paper, we propose to exploit the existing Internet infrastructure to achieve real-time discovery of embedded devices and environmental services. We investigate the utilization of the Domain Name System (DNS) as a scalable, pervasive, global meta-data repository for embedded devices, and its extension for supporting location-based discovery of Web-enabled physical entities.

The rest of the paper is organized as follows: Section II identifies related work and Section III describes the general approach for real-time service discovery through DNS. Then, Section IV explains our implementation to show the feasibility of this concept and Section V presents a small technical evaluation of the system. After, Sections VI and VII discuss practical issues and the overall potential of the approach while Section VIII concludes the paper.

II. RELATED WORK

The absence of standardized discovery methods for Web-enabled physical devices led to the development of online, global sensor directories. The most well-known directories available today are Xively\(^2\) (former Pachube or Cosm) and SenseWeb [9]. UrbanRadar [10], [11] is an interesting mobile application that uses Xively as a platform for discovering and interacting with sensor devices through the Web. Evrything\(^3\) was recently established aiming to develop online, social presence for physical entities. These infrastructures allow people

\(^{1}\)http://share.cisco.com/internet-of-things.html
\(^{2}\)https://xively.com/
\(^{3}\)http://www.evrything.com/
to share, discover and monitor in real-time environmental data from sensors connected to the Internet around the world.

A key feature of these online directories is that they provide open Web API supporting the development of third-party applications. The main drawback is that they are centralized, with a single point of failure. Decentralized approaches have also been proposed, such as IrisNet [12], which uses a hierarchical architecture for a worldwide sensor Web. G-Sense [13] is a peer-to-peer system for global sensing and monitoring. These approaches, although more robust and scalable, have not been largely adopted yet by the public. Sensor discovery in these cases is still a challenging issue.

More recent approaches towards real-time discovery of physical entities include Snoogle [14] and Dyser [15], [16]. Snoogle is an effective information retrieval system for wireless sensor networks, but it is questionable whether it could scale for the World Wide Web. Dyser follows a different approach, focusing on entities than on sensor devices, e.g. whether a classroom is occupied or not. To achieve this, the authors exploit the periodic nature of people-centric sensors by using appropriate prediction models. However, these techniques are computationally-expensive and there are no guarantees for periodic behavior of entities. Besides, Dyser requires additional Internet infrastructure such as sensor gateways.

On the other hand, microformats\(^4\) suggest ways for making HTTP data available for indexing and searching, but their use would increase our dependency on commercial search engines.

The limitations of the aforementioned related work, encouraged us to examine the possibility of exploiting the DNS system for service discovery. Applying an existing technology for discovery of physical services, especially one that has been ubiquitous for decades, offers many advantages, including well-defined support, easy configuration, experienced developers and users, and availability of open-source implementations.

In general, the overall idea of using DNS for service discovery is not new. DNS-based Service Discovery (DNS-SD) [17] proposes using standard DNS programming interfaces, servers and packet formats to browse a network for services. Similarly, Multicast DNS (mDNS) [18] provides the ability to perform DNS-like operations on a local network in the absence of any conventional unicast DNS server.

Even though these two protocols have been originally designed for device/service discovery in local networks, DNS-SD has been extended to provide wide-area service discovery. However, this functionality for service advertising is domain-centric, meaning that users are only able to be informed about services offered in some particular domain. In contrast, our proposal is service-centric, envisioning to enable global, ad hoc, real-time location-based discovery of pervasive services, offered by Web-enabled sensor devices. We try to conform to the DNS-SD protocol, where possible.

III. SERVICE DISCOVERY THROUGH DNS

DNS is a hierarchical, distributed naming system for computers. Its main functionality is the translation of domain names meaningful to humans into IP addresses meaningful to machines. DNS distributes the responsibility of assigning domain names and mapping those names to IP addresses by specifying authoritative name servers for each domain. Authoritative name servers are responsible for their particular domains, and in turn can assign other authoritative name servers for their sub-domains. This mechanism allows the DNS to be distributed and fault-tolerant. Domain names consist of labels, concatenated and delimited by dots, such as www.webofthings.org. The right-most label indicates the top-level domain, which is org in the previous example.

In the following subsections, we examine how to leverage the organization of DNS to support location-based, real-time discovery of environmental services.

A. Service Discovery

We propose the inclusion of a new top-level domain at the DNS, for example the .env domain from the word environment. Service discovery begins when a user types in his Web browser a URL ending with the .env label. We believe that this procedure shall be trivial to the user, who could be able to be informed about a particular service, offered at location, just by typing the following URL in his browser:

\[
\text{service.location.env}
\]

(1)

In this way, the user can receive a list of all sensor devices offering service, deployed in location, along with some general characteristics or capabilities of them, if available. The devices will be uniquely identified by a sensorid. The user can then select a particular device, and construct the following URL:

\[
\text{sensorid.service.location.env}
\]

(2)

By typing this URL in his Web browser, the DNS will translate it to the actual IPv4/IPv6 address of the corresponding sensor device, and the user will be able to interact with it, using any environmental services offered by this device. Service discovery may involve not only humans but also machines, for automated M2M communication. In this case, machines could discover useful services on demand, taking decisions automatically based on current environmental conditions.

B. Device Registration

As mentioned before, the .env domain is intended to support all embedded devices and environmental services which are enabled to the Internet and registered to the DNS. Specialized authoritative name servers may be responsible for this domain, allowing real-time registration of physical devices and their services through Dynamic DNS (DDNS) [19].

Hence, whenever a sensor device becomes available on the Internet, it would create a request to the .env DNS server, asking for registration. In this request, the device must specify its name, location and the services it offers. The .env DNS server could offer a Web API, allowing sensor devices to POST their discovery details in HTTP requests. In case all

\(^4\)http://microformats.org/
information is provided, the .env DNS server would acknowledge the request, assigning a fully qualified domain name to the device, registering it in its records.

A sensor device may offer various environmental services. Thus, multiple hostnames will be created for this device, each for a different offered service. For example, the device tempsensor123 located in Barcelona, offering temperature and humidity services, would get the domain names tempsensor123.Barcelona.temperature.env and tempsensor123.Barcelona.humidity.env.

Since the WoT proposes a resource-oriented architecture \cite{7,8}, services could be described using uniform resource identifiers (URIs), avoiding ambiguities when users construct environmental URLs in their Web browsers.

C. Interaction with Devices

When the IPv4/IPv6 address is resolved and the request is forwarded to the appropriate sensor device, the device needs to respond with a description of its functionality. This is necessary in order to understand the semantics of the device (e.g. indoor/outdoor, degree of accuracy, measurement unit) and how to interact with it in order to get informed about environmental conditions. Thus, a description language must be defined, which declares the device semantics, but also explains the interaction possibilities with it.

Example description languages that could be adopted are Extended Environments Markup Language (EEML)\cite{5} and SensorML\cite{6}. These are languages that describe the capabilities of sensor devices, but not the interaction with them.

Therefore, we propose to employ the Web Application Description Language (WADL)\cite{7}, which is a XML-based language providing a machine-readable description of HTTP-based applications. It can be considered as the RESTful equivalent of Web Services Description Language (WSDL)\cite{8}, which is a standard for describing SOAP-based Web services. WADL is intended for applications based on the Web architecture, and is a platform-independent way of describing services.

After the user receives the WADL description, he can construct the appropriate request, query the sensor device and get informed about the environmental conditions in the selected location, in a well-known format such as XML or JSON. The whole procedure can be observed in Figure 1.

D. Freshness of Information

The idea of online directories for Web services (e.g. UDDI\cite{9} for WS-*\cite{9}) never worked, mainly because of information inconsistency and unavailability. Through DNS, these inconsistencies can be avoided.

In general, the DDNS service allows the DDNS server to allocate a static hostname to a physical device, and whenever the device is allocated a new IP address, this is communicated to the DDNS provider by software running on the device.

Furthermore, Dynamic DNS Update Leases \cite{20} constitutes a method of extending DDNS to contain an update lease life, allowing a DNS server to perform DNS dynamic updates with an attached lease time, which is automatically deleted unless renewed before the lease expires.

In other words, the name server “forgets” after some time interval the registration of the sensor devices. Devices need to state frequently their operability to the DNS server, by re-registering, e.g. every some hours. In this way, dynamic IP address assignment could be supported and device unavailability or failure could be identified in a relatively small delay.

IV. IMPLEMENTATION

We used BIND\cite{10}, which is an open-source DNS software for Linux, to implement a .env DNS server in a local environment.

\cite{5}http://www.eeml.org/
\cite{6}http://www.opengeospatial.org/standards/sensorml
\cite{7}https://wadl.dev.java.net/
\cite{8}http://www.w3.org/TR/wsd1
\cite{9}http://uddi.org/pubs/uddi_v3.htm
\cite{10}http://www.isc.org/software/bind
To achieve our task following the operation of DNS as much as possible, we exploited DNS zones, which are subsets, often single domains, of the hierarchical domain name structure of the DNS. Zones contain mappings between domain names and IP addresses. We divided the .env domain name server into various unique zones, where each zone is defined by the service it supports and its location, hosting the hostnames and IP addresses of all devices that fulfill these preferences.

A sensor device can be registered using an address record (A), a service locator (SRV) and a text record (TXT). The A record maps the device to its IP address, the SRV defines the generalized service location and an optional TXT record describes the capabilities of each device and its features in a human-readable text. An example simple sensor registry could be the following:

```
geiger182.radiation.fukusima.env IN A 195.14.149.56
geiger182.radiation.fukusima.env IN TXT
"A Geiger Counter measuring radiation"
geiger182_http_tcp.radiation.fukusima.env IN SRV 0 5 80 radiation.fukusima
```

Users are able to send queries for sensor devices through BIND using the dig command, which is a command-line tool for searching name servers. Dig works by reading requests from operating system files. Queries are executed using the following command:

```
dig env.service.location.env axfr
```

The axfr parameter represents the zone transfer, needed to retrieve all sensor devices stored in a particular zone, offering service placed at location. When implementing the .env domain name, it is important to reference all zones in the DNS configuration file (named.conf), indicating the zone’s name and path to its content. The required record types (SOA, NS records) are declared in each zone’s file (db file), along with records about sensors’ information (A, SRV, TXT records). In addition, the reverse mapping zone in-addrarpa must be set, by defining PTR records.

Since working with configuration files may not be optimal, we experimented as well with an implementation of the .env domain using a MySQL database. We took advantage of Dynamically Loadable Zones (DLZ), enabling the insertion of new zones and records dynamically. The database consists mainly of two tables: dns_records, which stores all records for the .env domain name (SOA, NS, A, SRV and TXT records); and xfr_table, which stores all the distinct zones and the data necessary for the queries. In this case, the DNS configuration file named.conf needs to be modified, in order to connect to the database and set the dig command to return the needed records using an appropriate query.

V. Evaluation

Our evaluation efforts focused mostly on the performance of the approach in terms of response times and scalability. Both metrics are crucial for the feasibility of our concept.

We configured a typical laptop (2.10 GHz core duo, 2 GB RAM, Linux Ubuntu 12.04) to become a .env DNS server using BIND, and we used another laptop (1.66 GHz, 1 GB RAM, Linux Ubuntu 12.04) as a client, to send queries about environmental services to the .env DNS server (first laptop). Both laptops were located in the same local network, hence network delay was negligible. We tested the system under a variable number of zones and sensor devices per zone, both for the file and the database case. The number of (virtual) sensor devices ranged from one to 1M records.

To simulate data corresponding to sensor devices, we used the following convention: X.serviceY.locationZ.env where X, Y and Z could take (incremental) values from one to 1M, depending on the scenario. Queries from the client were performed randomly for sensor devices in each iteration, sampling from the uniform distribution.

Two different scenarios were considered: a) each zone had a random number of 1-10 sensors, and b) each zone had 80-100 sensor devices. We believe that these scenarios may represent well the first years of this approach if applied. The results in terms of response time at each scenario are presented in Figures 2 and 3 respectively.

In both scenarios, response times are very satisfactory, ranging below 15 msec in the first case and below 70 msec in the second. Apparently, the system needs less time to locate the proper zone and more time to locate the right sensor device inside the zone. Comparing the file with the database case, it seems that database offers better response times, especially in reduced load. However, as the load increases, the response times tend to reach the file case, increasing linearly. Perhaps
virtual sensor devices. Considering the prediction of 25 billion devices by 2015, obviously our tests cover only 0.004% of the predicted number. Larger experiments with much larger loads, including if possible more realistic data, are definitely needed, and it is something we are currently working on.

VI. Practical Issues

Our approach creates various practical issues. The general operation of the DNS is expected to be affected in terms of increased traffic, management and security, raising issues of reliability, privacy and trust. Traffic could be partly handled by involving numerous .env DNS servers, assigning different locations or service types to each of them.

Concerning management, since the DNS follows a hierarchical structure, the addition of the .env top-level domain is not expected to be a complicated task. However, the .env DNS servers would need to change their operations to support device registration and discovery queries, as explained in Section III. Availability of devices/services and freshness of information may be assured by DDNS update leases.

Our approach requires unique sensorid assignment to sensor devices that may have same services and location. Upon a conflict, the DNS server should automatically select a new name, typically by appending a digit at the end of its name. Since the .env DNS system would be a distributed repository, this assignment should be visible to all .env name servers, causing increased traffic between DNS servers for synchronizations. This issue can be mitigated by area- or location-based assignment of devices to the .env DNS servers.

Naming of services and locations is important too. For example, locations are named differently in each language (e.g. “Lisbon” in English vs "Lisboa" in Portuguese). To avoid ambiguities and assure uniqueness, electronic directory services such as X.50011 could be used, where a distributed database would contain unique names for services and locations.

Reliability and trust would be increased if some user-based feedback system is applied for sensor devices and their environmental services, similar to the way eBay works for rating its users. A feedback system could be realized if devices had social presence on the Web. It would then be easier for users to rate them, according to their quality of service. Such an approach would increase privacy too, allowing the owners of the devices to share them only with family, friends or everyone [21], [22]. Some reliability could be achieved by checking the IP addresses of the sensor devices, if they fall in the locations claimed by them during the registration process.

In regard to performance, the whole system would be optimized by configuring the .env DNS server to return directly the IP addresses of relevant sensor devices and not their hostnames, in case these IP addresses are static. The DNS server could even choose automatically the most appropriate sensor device for some user query (perhaps based on some criteria specified by the user), returning its IP address to the user quickly. However, this would add more complexity to the DNS system and may be undesirable.

11http://www.x500standard.com/
Moreover, performance could be improved by exploiting the built-in caching mechanism of DNS, for caching sensory data. For example, during their (re-)registration, sensor devices could include their latest measurements, which could then be forwarded to users who queried the .env DNS server for a relevant service, in case these measurements are still fresh. Finally, security is a crucial, large topic that needs to be considered. In general, all the aforementioned practical issues are matter of future research and need to be studied well in case the proposed approach is finally adopted.

VII. DISCUSSION

In the previous section, we discussed practical problems in adopting the proposed approach and we suggested some candidate (partial) solutions. This section focuses on the potential of the overall idea, touching upon aspects of automation, personalization and generalization. At first, the practice of discovering environmental services can be automated by selecting some particular sensor device from the list returned by the .env DNS server, parse its WADL file and construct HTTP requests without the user’s intervention. This is easier to achieve when services are exposed as RESful resources.

The whole procedure can be personalized, by selecting only devices that meet particular user preferences. For example, users may wish to interact solely with devices having positive online feedback or those belonging to well-known authorities such as governmental organizations. User preferences could be extracted from their online social networking profiles.

Even though our approach targets environmental services, it could be well generalized to support any kind of physical devices and pervasive services. To achieve this, standardized domain vocabularies need to be created, for facilitating the construction of queries by end users. For example, a user that wishes to park his car in Singapore could just need to type in his Web browser parking.cars.Singapore.env.

Defining extended environmental ontologies would encourage automatic information retrieval, generalized inferences and advanced Web mashup development very easily. For example, when the temperature in Seoul is obtained, then the general temperature of South Korea can be automatically inferred.

Since the whole idea is participatory-based, only people who are willing to share their sensor devices with the online community would do so. A culture could be created in the future around the concept of sharing environmental services with the rest of the world.

VIII. CONCLUSION

Embedded computing is becoming ubiquitous in our lives, being gradually blended with the Internet and Web. Discovering in real-time embedded devices enabled to the Internet, located anywhere around the world, remains an open problem. In this paper, we investigated how the DNS system can be extended to support global discovery of environmental services. Our small-scale implementation and evaluation indicate that it could be feasible to achieve automatic, real-time discovery of sensor devices, by means of DNS servers. Our proposal constitutes only yet an interesting idea and it would require wide acceptance by key stakeholders, researchers and engineers, in order to be adopted.

For future work, we plan to continue our measurements involving massively large numbers of sensor devices (around 1-10 billions), using realistic data if possible, to better study scalability and performance. We will consider the practical issues identified in Section VI, and try to suggest more complete and effective solutions. In parallel, we started developing a Mozilla Firefox extension that automatically discovers sensor devices just by typing relevant URLs, constructing their hostnames and interacting with them. We will use this extension as an effective demonstration of the proposed approach.

REFERENCES