Paper

ID Layer for Internet of Things Based on Name-Oriented Networking

Jordi Mongay Batalla, Piotr Krawiec, Mariusz Gajewski, and Konrad Sienkiewicz

National Institute of Telecommunications, Warsaw, Poland

Abstract—Object and service identification is considered as one of the main challenges in the field of Internet of Things (IoT), which can be solved by the introduction of the so called ID (IDentifier) layer. The objective of this layer is to expose IoT objects and services offered by them, to users. Common approach for ID layer is to create it in overlay manner, on the top of existing network. This paper presents a novel architecture of the ID layer, which is characterized by embedding ID layer functionality into the network plane. Moreover, this approach takes advantage of the Name-Oriented Networking (NON) paradigm. To gain easy access to the IoT objects and services, as well as native support for multicast service, human readable ID-based unified addressing with hierarchical structure was exploited. Additionally, in-network caching of forwarded IoT data, inherited from the NON, helps to reduce total network load and supports applications during collaboration with energy-constrained sensors. Such sensors may enter sleep mode to save energy and then the network nodes can serve requests for sensing data, arrived from applications, by using data stored in nodes' cache. The paper shows the concept of NON-based ID layer and describes functional architecture of network node paying attention on modules and mechanisms related with ID layer functionality. Primary ID layer processes, i.e., object/service registration, resolution and data forwarding are explained in detail. Moreover, the ID-aware network node was implemented on Linux-based platform and tested to check its forwarding characteristics. The tests showed the performance of the of ID network node in data plane operations, which are the more sensitive for scalability issues.

Keywords—Future Internet, ID-based routing, ID layer, Internet of Things, Name-Oriented Networking.

1. Introduction

Internet of Things (IoT) is considered as one of the main trends for further evolution in the area of information and communication technologies. IoT refers to a global network infrastructure linking a huge amount of everyday things, i.e., physical and virtual objects from the surrounding environment, which can communicate between them without human interaction. In IoT, objects are active participants of network ecosystem – they can recognize changes in their surroundings, share information about those changes or detected events with other network members, and perform appropriate actions in an autonomous way.

The open question in IoT research is identification and accessing of the objects and services offered by them, as well as how to bind objects/services to machine addressable and identifiable names [1]–[3]. These issues can be solved by creation of so-called ID (IDentifier) layer, common for all IoT devices. ID layer aims to expose objects and their services in unified way, and should be separated from the IP layer in order to avoid the limitations of IP addressing structure (i.e., no context or location awareness).

The problem of naming and addressing is recently widely investigated in the context of efficient content delivery through the Internet. Solutions which refer to Name-Oriented Networking (NON) model, as Content Centric Networking [4], change the "host-centric" paradigm of the current Internet, in example host-to-host communication, into a "content-centric" paradigm, which treats delivery of content as a primary communication primitive. NON-based architecture seamlessly supports identification, resolution and delivery of content.

In this paper a novel architecture of ID layer, which exploits features of NON for IoT purposes is proposed. Such a solution deals with the registration of object and services as well as the search and delivery of the information related to them. The idea is to include the ID layer into the network level and offering NON network facilities as, among others, in-network caching of IoT data, ID/location separation and support for multicast.

Basically, the proposal is a hierarchical addressing of objects following their physical location. This addressing also contains services offered by objects or groups of objects. The requests of data by the applications as well as the information from objects and/or services are sent through appropriate hierarchical tree. The nodes in the tree are able to cache the data for further requests during the validity period of the data. Over this time, new requests will be served from the nearest node that cached the data. This way, the request and data frames do not need to transfer the whole network (what avoids overload) and data are provided in a short time. Moreover, sensors may go on sleep mode for saving energy consumption, whereas sensing data are still available for applications.

In an intelligent house each room has many objects and the objects offer services to be used by applications in the house, the energy control application decides when to run the radiators located in each room. The objects are attached to the network node and the object and the services are addressed in the network. Then, the network can cache information given by, i.e., humidity meters in ambient with flowers or light sensors in order to avoid constant operation of sensors/actuators.

The paper is organized as follows. Section 2 presents a brief review of approaches for IoT ID layer. Section 3 explains applying the NON concept for creation of ID layer. Section 4 describes proposed ID layer architecture and details of ID-based routing. Performance tests results of prototype implementation are shown in Section 5. The paper is summarized and concluded in Section 6.

2. Related Works

The need for introduction of ID layer to communication stack is outlined by several current research projects which focus on the definition of the IoT architecture (for example FP7 IoT-A [5]). Such layer should perform identification tasks of IoT objects and services, regardless of their network localization.

Approaches for ID layer proposed so far can be divided into two groups, according to the way the separation between network locators and identifiers is assured. The first group assumed implementation of IoT objects identification inside of the application layer and realization of registration and resolution processes for IDs by using distributed databases. An example is the Ubiquitous Code (ucode) proposed by Koshizuka and Sakamura [6].

The ucodes are IDs of the objects that are created as unique numbers with a fixed length of 128 bits. Ucode ties virtual object with the thing, however without any correlation between characteristics and meaning of the object and value of the number assigned to it. Such relationship between ucode value and information about object which is linked with it, is recorded in distributed database created by resolution servers with hierarchical structure. The ucode approach implies, that terminals with installed the ucode client library have access to resolution servers' infrastructure. When the terminal reads the ucode, it has to query the resolution servers to obtain context represented by this ucode.

The second approach assumes introduction to network a new layer, what improves efficiency of messages transfer cause decisions during forwarding process can be made taking into account the ID. Such approach is represented in Veil-VIRO [7] and MobilityFirst [8], [9].

The Veil-VIRO solution [7] introduces a uniform convergence layer on the top of the link layer to ensure connectivity for large number of heterogeneous physical devices. This convergence layer provides support for underlying networks with dynamic structure and various layer 2 technologies, including both Ethernet-based as well as non-Ethernet solutions. The Veil-VIRO concept relies on structured *virtual ids* (vid) used by the convergence layer [10]. The vid address space contains representation either physical (i.e., layer 2) identifiers like Ethernet MAC addresses etc. and identifiers related with higher layers (for example IPv4/IPv6 addresses or flat-id names used at application level). The end hosts' vids have special structure and are created as follows. The first part of the vid, so called host-node part, is L-bit long and indicates VIRO switch (host-node), which given end host is directly attached to. The remaining l-bit long part is used to identify the end host in a set of end hosts connected to the same host-node, in example with the same L-bit prefix.

In VIRO a structured virtual id space is used not only for object address resolution, but also for routing and forwarding purposes. VIRO routing is based on Kademlia-like DHTs (Distributed Hash Tables) [11]. However, in contrary to traditional DHT approach, which assumes end-to-end connectivity and uses IP layer routing and look-up mechanisms, the VIRO must build end-to-end connectivity by itself, using to this aim layer 2 connections established between VIRO nodes.

MobilityFirst [8], [9] assumes that each object is distinguished by a Global Unique Identifier (GUID). The GUID is a string which comprises two parts: object owner's Public Key and value of hash key calculated for the object. The GUID is assigned to the object without any relationship with object's network address and/or location. MobilityFirst proposes to utilize a logically centralized Global Name Resolution Service (GNRS) in order to store information about mapping between object's GUID and its network address. GUID layer is on the top of the network layer and features flat structure. MobilityFirst assumes that network nodes forward data according to hybrid GUID/network address routing: routers can deliver packets considering packets' GUID only or using GNRS to discover network address associated with given GUID. Moreover, in order to allow applications to search objects, MobilityFirst requires implementation of external name assignment service, which is responsible for assigning and publishing object's GUID jointly with its semantic description.

In conclusion, presented solutions do not affect network technology and relay on overlay systems, particularly for discovering and accessing objects and services in IoT.

3. Using Name-Oriented Networking Concept for ID Layer

Name-Oriented Networking, exploited in Content-Centric Networking (CCN) [4], and its successor Named Data Networking (NDN) [12], constitutes a new paradigm for solving naming and addressing aspects in the Internet. It assumes, that network is aware, what data (content) are requested by users and transferred through nodes. When the user wants to download given content, it sends to the network an Interest packet that identifies the required data by its name. Next provider's application, which registered for a given name prefix, sends a Data packet as a response to this Interest. When the Data packet traverses through the network, network nodes cache it for serving forthcoming Interest messages.

In this paper an exploit the NON concept for creation of IoT ID layer is proposed. The main idea based on human readable IDs for objects and services. Also network nodes are labeled with such ID, and the structure of IDs is or-

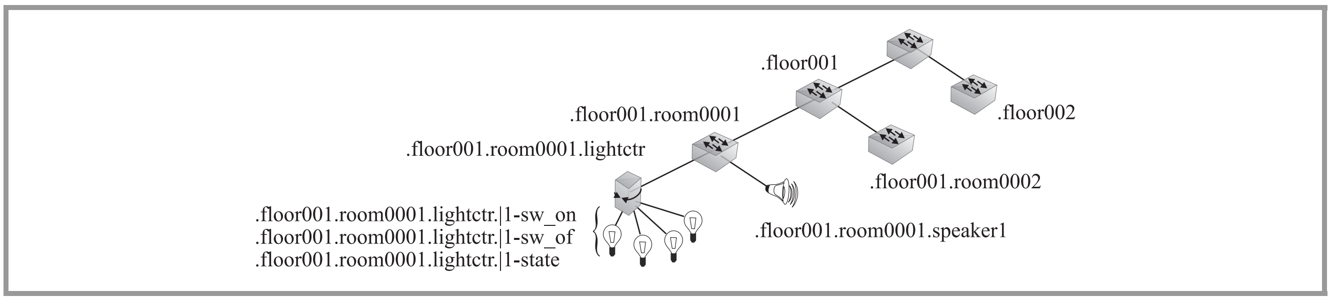
.floor001
.floor002
.floor001.room0001
.floor001.room0002
.floor001.room0001.lightctr
.floor001.room0001.speaker1
.floor001.room0001.lightctr.|1-sw_on
.floor001.room0001.lightctr.|1-sw_of
.floor001.room0001.lightctr.|1-state

Fig. 1. ID naming example.

ganized in hierarchical manner, which allows a tree-based ID routing, i.e., the data frames are transferred, according to ID carried in their headers, by established trees in both upload and download directions.

The IDs are created and managed taking into account the location of each IoT object. Name (ID) of every object, service or network node is formed by 8-byte ASCII-based word compatible with ISO 8859-1 standard [13]. The address (locator) of object/service is created as a chain of all the names, concatenated by a dot, from the root (which is marked as "∗") to given object/service. Figure 1 presents an example of address of the light controller (ID: lightctr) in the first room on the first floor (.floor001.room0001.lightctr) with exemplary services related with light 1: "switch on" (.floor001.room0001.lightctr.l1-sw_on), "switch off" (.floor001.room0001.lightctr.l1-sw_of) and "check status" (.floor001.room0001.lightctr.l1-state). Using human readable addresses for objects and services provides flexible and convenient way for creation multicast services. It is assumed that two symbols are reserved in naming: a dot "·" in order to build hierarchy essential in ID-based routing, and a star "∗" for multicast calls. Proposed hierarchical naming scheme and ID routing simplifies multicast connections. To address message to group of objects on given region, i.e., "switch on" all lights on the first floor, ID of the form: .floor001.∗.lightctr.∗-sw_on could be used. It is assumed that the object indicated by the full address can recognize requested service and takes the adequate action, otherwise it will return error message.

The forwarding nodes compare the ID from frame header with the address of the node and based on comparison result decide on which port forward the frame. Similar to CCN/NDN, network nodes can store forwarded data in cache, and then future requests from applications are served using node caches. This feature can bring benefits in IoT scenario if common behavior of small, battery-powered sensors is considered. These devices may work at active mode and serve requests received from applications, or may enter into sleep mode turning off both the receiver and transmitter to save power. In-network caching allows network nodes, which forward IoT data, to keep the results of sensing. In this way, the applications have access to them all the time, also within the period when the sensor is in a sleep mode. This characteristic is particularly profitable in scenarios, which assume existing of many different sensors in the network, where each sensor may enter into energy saving phase in different periods [14].

Embedding the ID layer into the network level offers awareness about IoT data to the network nodes. In this way, the network nodes are responsible for crucial operations for IoT: registration of objects and services available in the network, and retrieval information about the registered elements to the applications that ask for it. On the other hand this approach does not exclude attaching of IoT devices which are not aware of ID layer as, in example devices based on 802.15 standard family. In that case, the connected objects or applications would require performing appropriate encapsulation operations in the so-called object controller (see Fig. 2). This approach is in conformance with ETSI M2M functional architecture [20], where M2M devices are divided into two groups: first are those which have enough resources to implement encapsulation mechanisms and they interact directly with network nodes, and second are devices with constrained resources which connect with network nodes via object controllers (i.e., ZigBee sensors). This architectural approach is shown in Fig. 2.

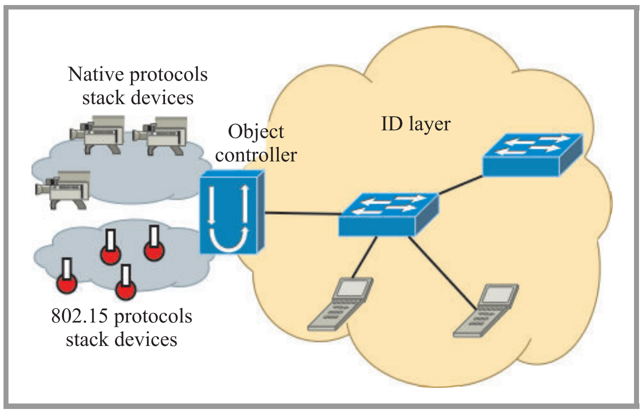
Native protocols stack devices
Object controller
ID layer
802.15 protocols stack devices

Fig. 2. Interworking between IoT devices connected to ID layer capable network nodes.

Proposed architecture for the ID layer implies the following three phases of data distribution:

Registration and publication – objects register itself, together with all the services that they offer, in the closest network node. It is the ID layer aware network node, which the registering object is directly connected to. After registration, object and services are available for applications connected to the system.

Resolution – the applications can ask for objects and services in the whole network (by sending resolution request to the root node) or in particular segment of the network (by sending resolution request to selected network node).

Data delivery – IoT-specific caching in network nodes improves the data gathering as well as delivery process. Since the data is placed closer to application servers, response time for application requests can be reduced, and also limit amount of traffic in the core network. When a sensor sends any information to an application, the information is cached in the network nodes, which responds fast to future requests of these data (during the validity time of the data) allowing that the sensors go on sleep mode. It is worth noting, that in contrary to the content data, significance of IoT data is restricted, therefore for IoT purposes an additional parameter has to be introduced: the validity time of the data. Its value is configurable and strongly depends on the place and the context of object usage. In the context of temperature control, some scenarios require validity time to be set to value of several minutes [19] whilst others may require higher dynamicity and, therefore, validity time should have lower value.

Summarizing, the most outstanding features of the proposed approach for ID layer are as follows. First, thanks to human-readable hierarchical naming scheme, our solution integrates addressing of IoT objects and services. By creating network address as a concatenation of IDs, it ensures separation of ID and locator while using one unique addressing structure. Involvement of mechanisms inherited from Name-Oriented Networking approach improves forwarding of ID layer frames as well as enables implementation by IoT objects different energy efficient operating modes. Proposed hierarchical ID-based routing considerably simplifies multicast communication at various scopes. Furthermore, characteristics of IoT objects/services are location-specific, which facilitates fundamental IoT processes as object/service registration and publication, and allows distribute them among many network nodes.
Other proposition for exploitation of NON concept for IoT purposes is presented in [15]. The authors propose a CCN-oriented service platform to implement IoT services in the network. Similar to this solution, they exploit hierarchical names for services and name tree to determine how *Interest* and *Data* messages should be treated. However, the platform proposed in [15] is designed as over-the-top solution above the network layer, and based on UDP/IP service.

4. ID Layer Architecture

In Fig. 3 the functional model of an IoT-aware node which complies with proposed ID layer architecture is shown. It contains three main planes:

- resolution and reachability – exposes registered objects and services to users and manages the reachability of the objects and services,
- naming and registration – aims at assigning unique, hierarchical IDs to the objects connected to the network,
- data forwarding – provides forwarding and routing rules for ID-labeled frames.

Taking into account that each network node contains interfaces for each plane and the network has hierarchical structure, all IoT-related operations, i.e., registration, publication, resolution, data request and data delivery, are managed only locally in the nearest network node. This results in a high flexibility and manageability and improves the response of the system when the number of handled things scales.
Detailed operations performed by the network node related with ID layer functionality and describe the exact functionalities of the blocks are presented in Fig. 3. Even when Fig. 3 shows security/privacy blocks, their functionalities are not described in the text since they are out of the scope of this paper.
When a new object is connected to the system or a new service is created, this object/service registers by sending a *Register* message to the edge node it is attached to. The *Register* message contains the name of new object/service jointly with its description, which will be used during resolution operations. When the node receives this message, it checks validity of the name and, if the name is correct, it performs a publication action by storing the name and information about newly connected object/service in the *Object/Service* register.
In the case when the ID is already registered or contains any reserved character, the *Register_failed* message is returned to the object. Otherwise, the *Register_accepted* message with the complete chain of concatenated IDs, which constitutes an address of the new object/service, is sent back to the node's port by which the *Register* message arrived. Since the registration and publication process is performed only in the edge network node, the *Register* message is not transferred to further network nodes. A newly attached object registers not only its own ID but also the offered services. These services are associated with actions taken by the object. Note that given action could have several different IDs for distinguishing the context in which the action is taken. It is important in case of controlling the same set of objects by different applications (i.e., door/window locking controlled by both user commands and fire monitoring). Also, given service ID may trigger aggregated actions in the object, for example switching on all the bulbs in one lamp.
The full address of new object or service is created as concatenation of the full address of the network node the new object/service is attached to, and the ID of this object/service. Address assignment is responsibility of the *ID naming* module. That structure of hierarchical routing

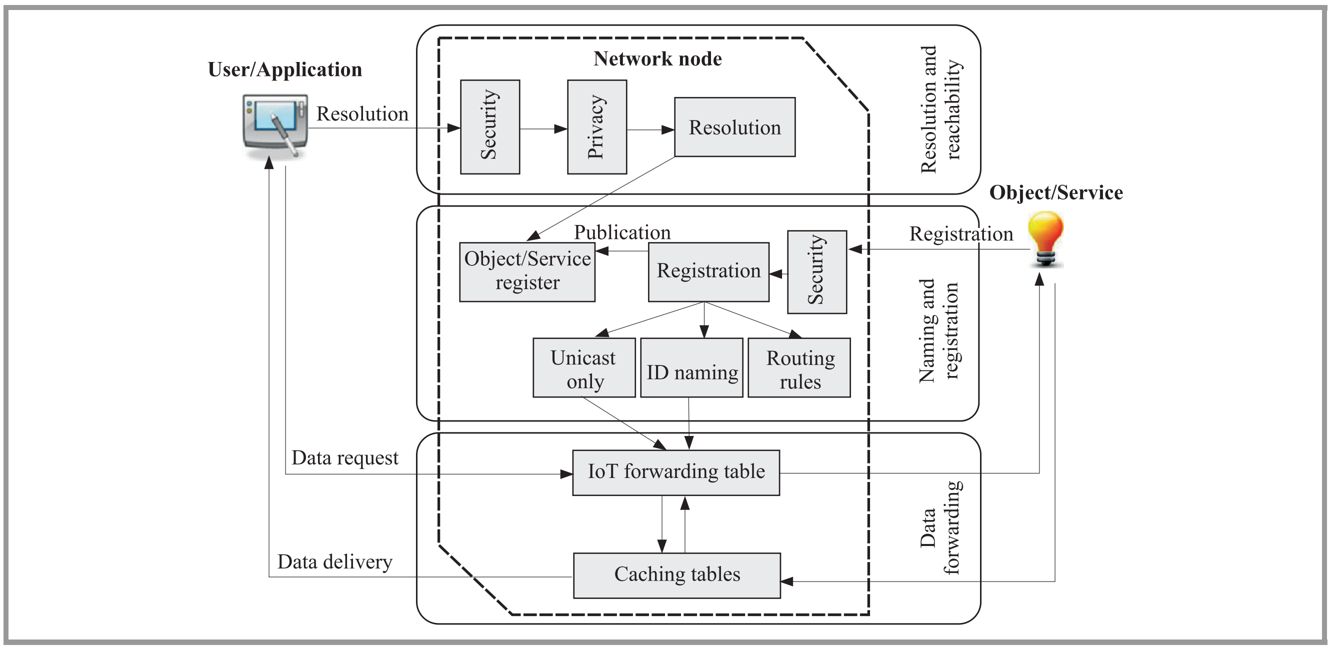

Fig. 3. Functional blocks of ID layer network node.

tree is delimited up to 64 levels, therefore the network permits addresses which contain at most 64 concatenated identifiers, where each ID being 8 bytes long. Taking into account that the maximum size of the payload field of an Ethernet frame is 1500 bytes, in case of the longest allowed object/service address, there are still almost 1000 bytes available for data. Such size of the available payload is sufficient for most IoT applications, which mostly send and receive short messages with control instructions, remotely sensed information or actuator commands. By limiting number of hierarchy levels, the problem of unbounded, variable-length naming, which is considered as one of the scalability issues related with NDN approach, is avoided [12].

When applications want to discover objects and services available in given node (i.e., in the second floor), they send a *Resolution* message to it. The node, in response, returns sequence of *Resolution_response* messages, where each message corresponds to one of the objects and services stored in node's *Object/Service* register. The *Resolution_response* message contains fields that hold the full address of the object/service as well as a description of this object/service. Note that such a description should be sufficiently extensive in order to make feasible easy manage of the services. At the same time, the description of the objects can be uploaded by the objects when any characteristic of the object/service changes. Additionally, the applications receive *Resolution_response* messages with addresses of the network node's child nodes. These addresses can be used by application for discovering all the objects and services existed in the sub-tree of the network node, i.e. in all rooms in the second floor.

The second way to obtain information about objects and services available in given domain is to send to the network a multicast *Resolution* message for discovering all registered objects/services i.e., in one room or at one floor. Using *Resolution* message addressed to the root node, the application can get information about objects and services existed in the whole network.

Note, that from application point of view, the resolution process is performed locally because the applications can know where desired object or service are located (i.e., in floor1.room1). This characteristic, which is inherited from IoT features, was exploited in this system. Moreover, it is assumed that it should be investigated to cope with mobility issues.

There is a multicast flag in each ID layer message header, which indicates whether the message is multicast-enable or not. If the multicast flag in the *Register* message header is not set, it points out that it is not allowed to access given object or service in a multicast manner (i.e., with using "$*$"). Information about exclusion of the object/service from multicast service is stored in the *Unicast only* module, which precludes that future multicast requests will be transferred to this object/service.

The *Routing* rules in the network are directed to maintain stable hierarchical addressing for routing packets. When one node is added or deleted to the network, then *Routing rules* modules should change for maintaining appropriate forwarding tables in data forwarding plane. The same occurs whenever one network node changes the position in the hierarchical tree.

The Data *Forwarding* plane, responsible for transferring ID-layer frames according to ID-layer header and rules defined by upper planes, is located on the bottom (Fig. 3). The network node begins frame forwarding process by checking, in the frame header, the address length and next reading appropriate number of concatenated identifiers (as the address itself). If the frame's destination address equals the node's address, then the frame is directed, according to the message type, to the Registration or Resolution module inside this node. Otherwise, the forwarding decisions are taken using the *IoT forwarding table*. This table stores IDs of objects and services registered in the node together with the multicast flag and the node port they are attached to.

The forwarding rules applied in *IoT forwarding table* are as follows. The prefix (i.e., upper part of hierarchical address) of the frame destination address do not match with the network node address. It means that the frame is addressed to object or service located in the other sub-tree in the hierarchy and therefore it is transferred to the parent node. The prefix of the frame destination address fits in with the network node address, taking into account the multicast symbol also. The frame is addressed to object or service located in the sub-tree rooted at this network node. Then, the next ID in the message destination address (according to the node hierarchy level) is analyzed, and:

– the frame is forwarded to the child node, whose ID fits in with the destination address next ID,

– if the next ID of destination address contains the multicast symbol, then the frame is forwarded to all child nodes except those for which a multicast exclusion entry is stored in the Unicast only module,

– if name of any child nodes does not match with the next ID in the frame destination address, the frame is discarded.

In parallel to forwarding tasks, the network node executes additional caching actions, which are invoked for two types of messages: *Data* and *Request*. Each network node contains two caching tables. The *Data Caching Table* handles the Data messages and stores: the source address of the object or service, the data related with this object/service and its validity time. Whenever the *Data* message with given source address and longer validity period arrives to the network node, an appropriate record in the *Data Caching Table* is updated. Additionally, sanity operation could be performed in order to erase expired entries and avoid problems with oversized table.

Once the *Request* message is received by network node, the node at first finds out if the requested information is available in the *Data Caching Table*. It checks the following conditions: whether the message destination address converges with an entry in the *Data Caching Table*, and whether the validity time of data found in the *Data Caching Table* is longer than the requested validity time carried in the *Request* message header. If the above conditions are met, the request is handled using relevant *Data* message from the table. Note that the synchronization between network nodes, objects and applications, which is required for data validity time verification, does not need to be higher than hundreds of milliseconds. Therefore, simply using of Network Time Protocol should be enough.

Another caching table in the network node is the *Request Caching Table*. It is used to store Request messages which are not served from the *Data Caching Table*. Such messages are transferred through the output port according to message destination address, and corresponding entry is created in the *Request Caching Table*. This entry contains the address of requested object/service and the port from which the *Request* message arrived. When the *Data* message comes to the node, as the answer to the *Request*, a record with the source address of this message is searched in the *Request Caching Table*. Next the message is forwarded to the port indicated in the record and the entry in the table is deleted. In the case when the message source address is not found in the table, network node discards received *Data* frame.

Records in the *Request Caching Table* are erased also when the validity time of given record expires. Moreover, in order to maintain acceptable size of the *Request Caching Table*, the network node may execute sanity operations to avoid persistence of entries in the table during a long time, even if corresponding requests have not been served yet. Thereby, even if the network nodes have limited resources, they are still able to handle large-scale number of flows [12]. On the other hand, some *Request* messages are not cached in the *Request Caching Table*, because their validity time is set to 0 as applications do not expect a response for them. Such situation may occur, for example, when *Request* message carries an action command addressed to actuator.

5. Prototype of ID Layer Node

To validate and test the proposed approach, the ID-layer functionalities of the network node were implemented on the top of Ethernet technology from scratch. Specific Ethertype value for IoT frames were to defined to distinguish ID-layer frames from legacy Ethernet frames. All frames with such an Ethertype are handled by the network node according to the ID-based forwarding rules.

All the necessary modules and procedures required to registration and resolution of IoT objects and services, as well as IoT data forwarding using an ID-based routing, have been developed in a Linux-based server (version 2.6.17 of kernel) with processor Intel Core 2 Duo Desktop Processor E8500 3.16 GHz. The modules responsible for registration and resolution processes were implemented in the user space, due to its flexibility and ease of use, whereas data forwarding modules were implemented in the kernel space to achieve high forwarding efficiency.

The testbed assumes a ring topology, as recommended by the benchmarking methodology for measuring network interconnect devices [16], and consists of one server, with installed ID layer modules, connected to the tester by two 1 Gbps Ethernet links. To generate and receive the IoT traffic with ID layer header the Spirent TestCenter tool, equipped with CM-1G-D4 card was used.

Next the performance tests for understanding the forwarding characteristics of the implemented network node were run. The tests conducted were two-fold: first the influence of the size of the *Request Caching Table* in Data frames forwarding performance was analyzed and secondly, forwarding throughput of *Request* frames was checked. In the first case, a *Request Caching Table* with growing number of entries, i.e., from 1,000 to 100,000 requested services was prepared, so that in the network the number of streams is in this range (up to 100,000) could be considered. Note that current versions of OpenFlow tables can handle up to 100,000 parallel flows [17]. Afterwards, a stream of *Data* messages and calculated the throughput for different *Request Caching Table* sizes was generated. Throughput was defined as the maximum rate of frames forwarded by the network node without any losses.

When the network node received a *Data* message, it searched the source address of the message in the *Request Caching Table*, and next forwarded the message to the output port, which was pointed out in the table entry. Through this output port the message was re-sent to Spirent TestCenter. The validity time of *Data* messages was set to 0, to exclude influence of caching them in the *Data Caching Table*. Size of Data frames was equal to 64 bytes.

The following test measured the forwarding throughput of *Request* messages. For this, a stream of *Request* messages in the Spirent TestCenter was generated. The requested service was the same one in all the messages and was not cached in the *Data Caching Table*. In this case, when the network node received the *Request* message, it did not find requested service in the *Data Caching Table*, therefore destination address of the message is compared with node address and the message is transferred to the output port, to which Spirent tool was attached in a ring topology. Since the validity time of the *Request* message was equal to 0, the message was not cached in the *Request Caching Table*. Also in this case, the *Request* frames were 64 bytes long.

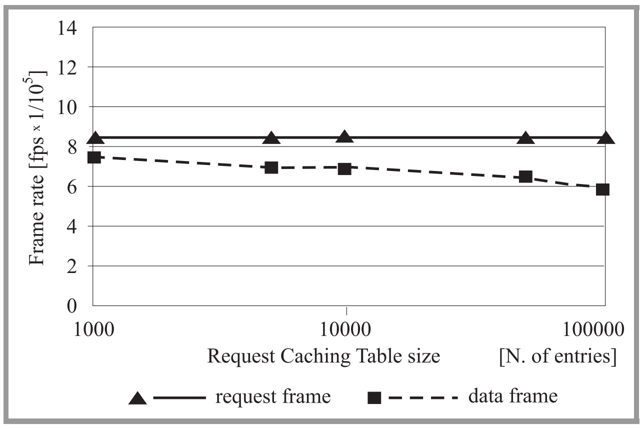

Fig. 4. Throughput of Data and Request frames for increasing size of Request Caching Table.

Figure 4 presents the throughput of *Data* messages for different size of *Request Caching Table*. When the number of entries in the table increase, the *Data* frame throughput slightly decreases. This does not occur for *Request* messages since *Request* forwarding operations did not involve the table during this test (validity time=0), which may be observed in the *Request* forwarding results presented in the same figure. The throughput of *Data* frames is visibly lower than the throughput of *Request* frames, even for small size of *Request Caching Table*. This is because, in the presented test conditions, the *Data* messages required more processing operations in the network node than *Request* messages. Note the ID layer node was implemented without any optimization in *Request Caching Table* searching operations (the table is created using simple ASCII hash function).

The main conclusion of Fig. 4 is that also for bigger *Request Caching Tables*, the forwarding throughput of developed ID-aware network node is in the range of software IP routers – in [18] the authors show, that, in the case of minimum-size Ethernet frames, software router based on high-end PC can forward up to $6 \cdot 10^5$ packets/s. Note, that there are no errors in the network node performance for the forwarded *Data* and *Request* frames for flow rates fewer than the throughput value. Therefore, it could be affirmed that the network node for up to 100,000 entries stored in *Request Caching Tables* works properly.

6. Conclusions

Internet of Things is a network of "smart" objects, which autonomously can find and cooperate with other members of the network. For this purpose, a device-independent abstraction layer, called ID layer, is introduced into the protocol stack.

In this paper, the solution for ID layer, which bases on Name-Oriented Networking paradigm, was proposed. Contrary to many proposals for the ID layer presented in literature, which do not introduce any changes into network plane and build the ID layer on top of it, IoT awareness into the network was introduced. Such approach enables to avoid overlay solutions and helps to achieve high efficiency and simplification for IoT related operations as object/service registration, object/service searching and delivering of IoT data.

This proposition of ID layer involves a Name-Oriented Networking addressing scheme for objects and services offered by them, together with hierarchical ID-based routing. The ID-based routing introduces new capabilities in IoT networks, because it is characterized by separation between identification and location: particular names (words) of an ID indicate entity, virtual or physical, whereas the ID as a whole (i.e., the chain of words) indicates its current localization in hierarchical tree. This provides flexible and convenient access to IoT resources.

Moreover, awareness of IoT data at the network level allows for introducing data caching functionality in network nodes. It improves effectiveness of network utilization and makes feasible cooperation between IoT applications and sensors with applied energy consumption saving mechanisms.

A prototype of the ID-aware network node using server with Linux operating system was developed. The imple-

mentation is based on standard IEEE Ethernet technology, resulting that it can coexist in network with nodes which have implemented only legacy protocol stack. Such nodes are unaware of ID layer messages and the coexistence is possible through tunneling the IoT traffic between two contiguous ID-aware network nodes by applying standard tunneling technique, for example IP-tunneling.
The results of experiments on performance of implemented ID-aware network node for forwarding ID layer frames, shows that proposed solution achieves acceptable throughput, similar to software IP router, even when the *Request Caching Table* has significant size. Nonetheless, it was clear that forwarding performance of ID-aware network node comes down when the number of IoT data flows increase. However, this performance reduction does not seem to be a problem for the correct running of the prototype, so it can be concluded that the solution is acceptable for intra-domain routing, where scalability issues are not crucial.

References

[1] SENSEI Project, "Reference Architecture", Deliverable 3.2, 2008.

[2] SWIFT Project, "D207 – Final SWIFT architecture", 2010.

[3] IoTWork Project, "WP 2 – communication networks D2.1 – IoT Addressing schemes applied to manufacturing", 2010.

[4] V. Jacobson *et al.*, "Networking Named Content", in *Proc. ACM CoNEXT 2009*, Rome, Italy, 2009, pp. 1–12.

[5] Internet of Things Architecture "Project Deliverable D1.2 – Initial Architectural Reference Model for IoT", 2011 [Online]. Available: http://www.iot-a.eu

[6] N. Koshizuka and K. Sakamura, "Ubiquitous ID: Standards for Ubiquitous Computing and the Internet of Things", *IEEE Pervasive Computing*, vol. 9, no. 4, pp. 98–101, 2010.

[7] J. Sourabh, C. Yingying, and Z. Zhi-Li, "VEIL: A "Plug-&-Play" Virtual (Ethernet) Id Layer for Below IP Networking", in *Proc. 1st IEEE Worksh. Below IP Networking 2009*, Hawaii, USA, 2009.

[8] Jun Li *et al.*, "Supporting Efficient Machine-to-Machine Communications in the Future Mobile Internet" in *Proc. IEEE Wirel. Commun. Netw. Conf. Worksh. WCNCW 2012*, Paris, France, 2012, pp. 181–185.

[9] Jun Li *et al.*, "Enabling Internet-of-Things Services in the Mobility-First Future Internet Architecture", in *Proc. IEEE Int. Symp. World Wirel., Mob. Multim. Netw. WoWMoM 2012*, San Francisco, USA, 2012, pp. 1–6.

[10] L. Guor-Huor, J. Sourabh, C. Shanzhen, and Z. Zhi-Li, "Virtual Id Routing: A scalable routing framework with support for mobility and routing efficiency", in *3rd Int. Worksh. Mobil. Evolv. Internet Arch. MobiArch'08* co-located with SIGCOMM 2008, Seattle, USA, 2008, pp. 79–84.

[11] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric", in *Proc. 1st Int. Worksh. Peer-to-Peer Sys. IPTPS'02*, Cambridge, USA, 2002, pp. 53–65.

[12] H. Yuan, T. Song, and P. Crowley, "Scalable NDN Forwarding: Concepts, Issues and Principles", in *Proc. 21th Int. Conf. Comp. Commun. and Netw. ICCCN*, Munich, Germany, 2012, p. 1–7.

[13] "Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1", ISO/IEC 8859-1:1998, 1998.

[14] A. K. M. Azad and J. Kamruzzaman, "A Framework for Collaborative Multi Class Heterogeneous Wireless Sensor Networks", in *Proc. IEEE Int. Conf. Commun. ICC'08*, 2008, Beijing, China, pp. 4396–4401.

[15] I. Cianci *et al.*, "Content Centric Services in Smart Cities", in *Proc. Next Generation Mobile Applications, Services and Technologies Conf. NGMAST 2012*, Bari, Italy, 2012, pp. 187–192.

[16] S. Bradner and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, 1999.

[17] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Comp. Commun. Rev.*, vol. 38, pp. 69–74, 2008.

[18] A. Bianco *et al.* "OpenSource PC-Based Software Routers: a Viable Approach to HighPerformance Packet", *Qual. of Serv. in Multiserv. IP Netw.*, vol. 3375, pp. 353–366, 2005.

[19] "ISTA Temperature Report" [Online]. Available: http://www.ista.org/pages/resources/TempRH.php

[20] ETSI TS 102 690 V1.1.1 (2011-10): "Machine-to-Machine communications (M2M); Functional architecture". ETSI Technical Specification.

Jordi Mongay Batalla was born in Barcelona, Spain, in 1975. He received the M.Sc. degree in Telecommunications from Valencia University of Technology in 2000 and Ph.D. from Warsaw University of Technology in 2010. His work experience includes jobs in Centro Nazionale di Astrofisica in Bologna, Italy, as well as Telcordia Poland. Currently, he is with National Institute of Telecommunications as Associate Professor. His research interest focus mainly on quality of service in diffserv networks and next generation network architecture. Moreover, he is an active researcher in the challenges related with Future Internet.
E-mail: jordim@itl.waw.pl
National Institute of Telecommunications
Szachowa st 1
04-894 Warsaw, Poland

Piotr Krawiec received M.Sc. and Ph.D. (with honours) degrees in Telecommunications from Warsaw University of Technology (WUT), in 2005 and 2011, respectively. Since 2012 he is an Assistant Professor at the Department of Internet Architectures and Applications, National Institute of Telecommunications, Poland. Since 2005 he is a member of the Telecommunication Network Technologies research group at Warsaw University of Technology. His research areas include IP networks (fixed and wireless), content-aware networks, Future Internet architectures, prototyping and testbeds.
E-mail: P.Krawiec@itl.waw.pl
National Institute of Telecommunications
Szachowa st 1
04-894 Warsaw, Poland

Mariusz Gajewski has been employed at the National Institute of Telecommunications since 1998. He received his M.Sc. degree in Telecommunications from the Warsaw University of Technology. He specializes in technical aspects of network architecture, IPv6 protocol testing as well as Future Internet architectures.
E-mail: M.Gajewski@itl.waw.pl
National Institute of Telecommunications
Szachowa st 1
04-894 Warsaw, Poland

Konrad Sienkiewicz has been employed at the National Institute of Telecommunications since 1997. He holds a graduate degree in Telecommunications from Warsaw University of Technology (1997). He specializes in technical aspects of network architecture, NGN and IP networks, as well as Future Internet.
E-mail: K.Sienkiewicz@itl.waw.pl
National Institute of Telecommunications
Szachowa st 1
04-894 Warsaw, Poland