Enhanced DNS Message Compression - Optimizing mDNS/DNS-SD for the Use in 6LoWPANs

Ronny Klauck
IHP
Im Technologiepark 25, D-15236 Frankfurt (Oder), Germany
eMail: klauck@ihp-microelectronics.com

Michael Kirsche
Computer Networks and Communication Systems Group
Brandenburg University of Technology Cottbus, Germany
eMail: michael.kirsche@tu-cottbus.de

***Abstract*—With the integration of smart objects into the Internet with the help of tiny IP stacks, a direct connection between these objects and ordinary computational devices can be realized at the IP layer. As IP alone cannot ensure an automatic integration at the higher layers, an homogeneous access via services (e.g., discovery, self-configuration) for the Internet of Things vision should be provided for all connected device types. Multicast DNS and DNS Service Discovery are established and widely used standards in current IP-based networks to enable the discovery of devices and services at the application layer with DNS messages. To comply with the current Internet infrastructure, the lightweight implementation *uBonjour* makes mDNS and DNS-SD available on smart objects. As DNS does not meet the requirements of low data rate smart object networks, we propose to extend DNS with enhanced message compression mechanisms to effectively reduce the number of exchanged IP packets in 6LoWPANs while ensuring backward compatibility.**

***Keywords*-Internet of Things, mDNS/DNS-SD, Contiki**

I. Introduction

The Internet of Things (IoT) vision uses so-called smart objects to interconnect the physical world with the Internet [1]. With the invention of tiny IP stacks, smart objects can use IP to communicate natively with each other, different IP networks as well as any IP device while respecting the Internet protocol's end-to-end principle [2]. The routing between smart object networks and traditional IP networks is done in the border routers, which convert IEEE 802.15.4 (max. 127 Bytes) / 6LoWPAN[1] frames to Ethernet / IPv6 frames (max. 1280 Bytes) and vice versa. To efficiently embed IPv6 packets in 802.15.4 frames (e.g., fragment oversized packets, statelessly compress packet headers, forward packets via multi-hop wireless routes) and thus use IP in smart object networks, the Internet standard RFC 4944 [3] was proposed to enable a seamless integration of resource constrained devices [4, Sec. 6].

Currently developed application layer protocols for smart object networks like the Constrained Application Protocol (CoAP) [5] try to comply with these restrictions in terms of low message overhead. Unfortunately, these protocols are not 100% compatible to established and widely used standards in the Internet, because they rely on specialized protocol translators which lead to a loss of flexibility and end-to-end functionality [4]. A seamless integration of smart objects into the current Internet infrastructure requires: (i) a standardized scheme to discover joining objects and their advertised services while complying with the Internet's IP standard as well as (ii) self-configuration ("arrive and operate", scalability) to handle the possible large number of objects emphasized by the IoT vision [4]. Instead of introducing new protocols that realize discovery mechanisms for smart objects, the adaption of established protocols (e.g., mDNS and DNS-SD) should be favored [6]. As mDNS and DNS-SD were initially designed for desktop systems with nearly no limit of bandwidth, both protocols lack optimizations for low data rate smart object networks. This work introduces enhanced compression methods for DNS which should extend the current DNS standard [7, Sec. 4.1.4] to enable an efficient message transport between smart objects and ordinary computational devices.

The rest of this work is structured as follows. Section II discusses the state of the art and related work. Section III introduces the limiting factor of current DNS message compression in 6LoWPAN while Sections IV and V present technical requirements and architectural solutions for the enhanced DNS message compression in 6LoWPANs. Initial performance evaluations are presented in Section VI while concluding remarks in Section VII complete this work.

II. State of the Art and Related Work

Multicast DNS (mDNS) [8] and DNS Service Discovery (DNS-SD) [9] are parts of the IETFs *Zeroconf* initiative to facilitate standards in the field of service-oriented networking, also known as *Bonjour*. mDNS allows to map domain names to network addresses without the help of any server in the local (ad hoc) network. DNS-SD enables to locate and to publish services in a network while using so-called DNS resource records (e.g., `SRV`, `TXT`, `AAAA`, and `PTR`). This operational scheme is directly connected to the concept of Service-oriented Architectures (SOA) [10]. SOA enables a seamless integration and interaction of different device types while specific devices are abstracted as services according to their offered functions. The abstraction of functionality into services is what uBonjour ultimately aims for with the discovery service for smart objects. Both protocols together enable an application- and network-independent announcement of services and both are nowadays supported on nearly every operating system.

[1]IPv6 over Low power Wireless Personal Area Networks (IPv6 over IEEE 802.15.4) [Online] http://www.ietf.org/html.charters/6lowpan-charter.html

A. *uBonjour: DNS-Based Service Discovery for the IoT Vision*

uBonjour [11] is a lightweight and memory-efficient implementation of mDNS/DNS-SD for resource constrained smart objects running the Contiki OS [12]. It enables discovery and addressing of devices and available services in IP-based network environments. Intermediate systems (i.e., protocol gateways) are not necessary anymore to interconnect smart objects with ordinary computational devices. Thus, smart objects can be treated as a natural part of any IP network while gaining new and direct interaction possibilities for users with their IoT-driven environment. Application protocols can register and announce their availability as services in the network as well as discover other devices that use the same application protocol. This enables a vendor independent discovery and collaboration of different device types (refer to [13]), while the coexistence of a set of application protocols is possible. The general features of uBonjour for application protocols and developers are: the resolving of hostnames and the discovering, registering, removing, and updating of services.

B. *DNS Message Compression*

A DNS message is composed of a DNS header and, at least, one resource record. The DNS header takes 12 Bytes of a DNS message. The number of embedded resource records in a DNS message is stored in several fields of the DNS header, like the *answer record count* or the *additional record count* [7, Sec. 4.1.1]. A DNS record contains fields for the (domain) name, the resource type, the class code, the TTL, the length of the resource data, and the resource data [7, Sec. 3.2]. Embedding a set of resource records in a single DNS message will save 12 Bytes per each additional added record compared to sending each DNS resource record in a separated DNS message.

In DNS, a name has to be split into a sequence of label lengths and then labels. A label length counts 1 Byte while each label is taking 1 Byte per character. DNS name compression is a possibility to reduce the size of a DNS resource record while shortening names by using pointers to a prior occurrence of the same name [7, Sec. 4.1.4]. Therefore, names in a message can be represented as a sequence of labels, as a pointer or as sequence of labels ending with a pointer. The length of a pointer is only 2 Bytes, because it is represented by a two octet sequence containing the pointer flag (the first two bits) and the offset from the start of the prior occurrence of the same name. Pointers can be distinguished from a label by setting the first two bits to ones (pointer flag) while labels must start with two zero bits.

Overall, *only* names of a DNS resource record are currently considered to decrease the DNS message size while requiring just a minimal larger code size and no additional buffers for the generation of name pointers and the calculation of their offsets. A strategy to combine both compression possibilities of DNS for the efficient use in uBonjour in respect to the characteristics of Contiki is introduced in Section IV. Furthermore, Section V presents our ideas for the enhanced compression of further fields and additional information of DNS resource records.

III. Limiting Factor

Examples of all required record types and their corresponding lengths to announce the availability of *XEP-0174* (uBonjour is part of the uXMPP software stack for smart objects, refer to [14, Sec. 4.5.2]) are shown in Table I.

Furthermore, the repetition of DNS names is depicted: the name field of the `SRV` record is included in the name field of `TXT`, in the resource data field and as a substring in the name field of `PTR`; the name field of `AAAA` occurs in the resource data field of `SRV`; the last substring of all names is repeated in every mentioned field and a prior occurrence can be replaced with a pointer within a single DNS record. The minimal length of a DNS record is the sum of fields with fixed lengths, the length of additional information (e.g., port, IP address, user-defined text) and the length of all used names. In Table I the column for the length holds two values: the first value assumes that each name could be replaced with a pointer; the second value in brackets shows the minimal length of a DNS record if no pointer within a record could be used (this represents the case when each record has to be sent by a separated DNS message), including the service name of *XEP-0174* (e.g., `_presence._tcp.local`). It is important to note that both cases do not include the length of the owner name and the hostname, because these are individual values of a device and their length cannot be predetermined. Therefore, the recommended length for each name is set to three[2]. Enabling all pointer possibilities is only possible if all records are included in a single DNS message, because only then the beginning `SRV` record will hold a sequence of labels for the owner name, the service name, and the hostname while the other DNS records can use pointers to a prior occurrence of a substring. In this case, the minimal required DNS payload size is 108 Bytes without owner name and hostname.

IV. Adjustable DNS Message Compression (ADMC)

The main goal of the *Adjustable DNS Message Compression* (ADMC) is the efficient implementation of the already defined DNS message compression (refer to Section II-B) for uBonjour in respect of the lower level fragmentation of different smart object hardware platforms on Contiki. DNS message compression can reduce the response time and the number of packets for retransmissions in case of connection issues, if a small number of IP packets must be sent. DNS compression hence minimizes the DNS message overhead for service announcement via DNS-SD in terms of DNS record length and number of used DNS messages.

The highest compression ratio for a DNS message can be achieved when all four required DNS record types of DNS-SD fit in a single DNS message and when every occurrence of a name can be replaced by a pointer. A DNS record has fields with fixed lengths (e.g., type, class, TTL, resource length) and with variable lengths (e.g., name, resource data). DNS message

[2]A string of the length of three offers more than 2 million possibilities (e.g., 128^3) in the case of using *American Standard Code for Information Interchange* (ASCII) to create enough unique names for a number of smart objects in a local domain.

TABLE I
EXAMPLES OF USED DNS RECORD TYPES FOR *XEP-0174* AND THEIR FULL AND MINIMAL LENGTH.

Type	**Example (name, type, class, TTL, resource length, resource data)**	**Length (Min.)**
SRV	con._presence._tcp.local, SRV, IN, 3600, 17, 12345 ctk.local	20 (41)
PTR	_presence._tcp.local, PTR, IN, 3600, 25, con._presence._tcp.local	14 (35)
TXT	con._presence._tcp.local, TXT, IN, 3600, 13, status=avail	25 (46)
AAAA	ctk.local, AAAA, IN, 3600, 16, aaaa::212:7402:2:202	28 (34)

compression heavily concentrates on the fields with variable lengths. These fields hold the individual service information (e.g., owner name, hostname, service name) of a device.

Unfortunately, some hardware platforms only support small IP packet sizes (refer to [11, Sec. 3.7]) when running Contiki OS. This hinders the combination of all required DNS records into a single DNS message. The supported IP packet size relies on the lower layer fragmentation skills of the uIP stack [15], which again depend on the hardware platform and its built-in radio transceiver. The Zolertia Z1 hardware platform has one of the lowest Contiki IP payload sizes: 80 Bytes vs. 180 Bytes of the Tmote Sky for usable application data. It is therefore a good reference platform to calculate the necessary boundaries for the optimal number of DNS resource records fitting in a single IP packet of the uIP stack. The available DNS payload size for the Z1 hardware platform is only 68 Bytes. A decision matrix (shown in Table II) was hence created to show the boundaries of the IP packet size with the number of DNS records, which can be integrated into a single DNS message depending on the available DNS payload size.

TABLE II
DECISION MATRIX TO INTEGRATE THE CORRECT NUMBER OF DNS RESOURCE RECORDS INTO A SINGLE DNS MESSAGE (IN BYTE).

IPv6 Payload Size (x)	**Min. DNS Payload**	**IPv6 Packet(s)**
$80 \leq x \leq 140$	68	3
$x > 140$	108	1

Possible combinations of integrating two DNS records into a single DNS message in case of 68 Bytes payload size are: the PTR record with the TXT (60 Bytes; leaving 8 Bytes for owner name) or with the SRV record (55 Bytes; leaving 13 Bytes for owner name and hostname) into a single DNS message. So, only three DNS messages need to be sent (further optimizations are necessary, refer to the upcoming Subsection V-C). The implementation of *ADMC* in uBonjour is based on the calculated boundaries shown in Table II. This ensures the efficient use of DNS name compression for each supported hardware platform of Contiki through the integration of the correct number of DNS records into a single DNS message, depending on the available IP payload size. *ADMC* is backward compatible to existing DNS standards and uses the default configuration parameters (e.g., 6LoWPAN fragmentation, uIP buffer sizes, no header compression) of Contiki for each supported hardware platform.

V. ENHANCED DNS MESSAGE COMPRESSION METHODS

Hardware platforms with a small IP payload size (e.g., the popular Zolertia Z1 hardware) could not be fully optimized with *ADMC* for low data rate networks, because the sending of a service announcement by uBonjour still requires 3 separated DNS messages. This subsection discusses not yet considered optimization approaches to reduce the DNS message overhead and the response time of uBonjour in 6LoWPANs. Our ultimate goal is to find standard-compliant compression approaches to integrate all four DNS resource records into a single IP packet, while supporting most hardware platforms. Each optimization is compared in terms of implementation effort, the number of used IP packets, and the backward compatibility to the RFC 1035 [7, 4.1.4]. Finally, the reasonable compromise of all enhanced optimizations was taken and summarized as *ADMC Enhanced*.

A. Thoughts on using 6LoWPAN Compression

The reason for the small IP payload sizes for application data is that the IP and UDP headers take a large part of each sent IP packet (e.g., max. 48 Bytes) when using IEEE 802.15.4 links (cp. [16, Sec. 4]). To overcome this restriction, a compression format for IPv6 datagrams over IEEE 802.15.4-based networks was defined in RFC 6282 [17] as IPv6 Header Compression (IPHC) and Next Header Compression (NHC). For the communication with global addresses, the IP and UDP headers are compressed down to 10 Bytes [18]. This will release 38 Bytes for both headers and could theoretically enlarge the DNS payload size of the Zolertia Z1 to 106 Bytes. As the minimal length for all four DNS resource records is 108 Bytes, a single DNS message cannot be used here without further optimizations. In contrast to *ADMC*, the number of sent IP packets can be reduced from three to two for the Zolertia Z1 hardware platform. Overall, the 6LoWPAN compression is very helpful for hardware platforms with small available IP payload sizes. Since the compression works at the lower layers, no changes to DNS standards are necessary. The drawback is that not all needed DNS resource records can be integrated into a single DNS message (IP packet).

B. Class Code and TTL Field Compression

The standardized DNS message compression does not consider repetitions in the fields with fixed lengths. For this reason a size of 10 Bytes is unusable in each DNS record, although a repetition of the class code or the TTL value can occur when multiple DNS resource records are sent in a single DNS message as shown in Table I. In this case all DNS records have to set the same values for the class code and the TTL field. Both fields take a length of 6 Bytes (e.g., 2 Bytes for class code, 4 Bytes for TTL field), while the use of a pointer to a prior occurrence uses only 2 Bytes. The unusable amount is reduced to 6 Bytes (leaving the type, the resource length fields and a pointer) with our approach of using pointers for both fields. In the best case, 12 Bytes can be saved when all four DNS resource records are sent in a single DNS message or 4 Bytes in the worst case, when only two DNS resource records will fit in a single DNS message. The second case would lead to a length of 12 Bytes for the owner name for the combination of `PTR` and `TXT` record or to a length of 17 Bytes for owner name and hostname for the combination of `PTR` and `SRV` record in the case of 68 Bytes.

As our compression schema of the class code and the TTL field differs from the compression of names, another pointer flag is required to indicate this optimization. Hence, the first two bits cannot start with two zero bits (indicates a label) or to ones (indicates a pointer to a name) as described in Subsection II-B). All other combinations (e.g., `10` and `01`) are reserved by RFC 1035 [7, 4.1.4] and could be used to indicate a pointer for our class code and the TTL compression in the future. This optimization saves only a few Bytes, but its implementation is done in a few lines of code while providing backward compatibility with the current DNS standard.

C. Redundant Information Filtering

Sending four DNS resource records to announce a service in the network contains a lot of redundant information (e.g., names, IP addresses). From our perspective this means that information can be reconstructed while redundant information are kept back to reduce the network traffic. The `SRV` record holds information to build a whole `PTR` record and the beginnings of `TXT` and `AAAA` records. The rest of the information (e.g., IP address, user-defined text) can be easily appended to the resource data of the corresponding `SRV` record. Thus, the minimal length of a single DNS message containing all information for the four DNS resource records will be reduced from 108 Bytes to 74 Bytes with our optimization. Regrettably, this is not small enough for the available DNS payload sizes of the Zolertia Z1 hardware platform, but we can cut the required number of DNS messages into half, while sending only a `SRV` record with appended IP address and an additional `TXT` record (uses 2 instead of 4 separated DNS messages, compare with Table II). Another redundant information are the IP addresses of the `AAAA` records, which we can reconstruct from the lower layers while using the source address of the IP header. This principle is inspired by RFC 2464 [19], which defines the reconstruction of 128 Bit IPv6 addresses from 64 Bit MAC addresses. An implementation can be either realized via a pointer flag[3] in the DNS resource record or by extracting[4] the source address directly from the IP packet. With that in mind, we only need to append the user-defined text to the `SRV` record, which will reduce the minimal length of a single DNS message simulating all four DNS resource records to 56 Bytes. This will allow us to use a single DNS message for the Zolertia Z1 platform (e.g., 68 Bytes), which will leave 12 Bytes for an individual owner name and hostname.

[3]This will work without heavy modifications when no IP header compression method (e.g., HC1, HC2, IPHC/NHC) is enabled in the 6LoWPAN.

[4]The lower layer (IP stack) has to provide such functionality via API calls.

The main drawback of our optimization is that it will not be understood from mDNS- and DNS-SD-compliant implementations, because they will expect all four DNS resource records. To overcome this issue and enable backward compatibility, a filter mechanism (comparable to a filter rule used in firewalls) is needed. Such a filter mechanism can be integrated in the Avahi daemon, which already acts as a DNS message repeater between different network interfaces (e.g., IEEE 802.15.4, IEEE 802.11 radio links) as explained in Section VI-A. Our filter mechanism works in bidirectional order: the `TXT`, `PTR` and `AAAA` records sent to the IEEE 802.15.4 network interface will be blocked and their information will be appended to the corresponding `SRV` records; the `SRV` records sent from the IEEE 802.15.4 network interface are used to generate the necessary `TXT`, `PTR` and `AAAA` records while these information are removed from the corresponding `SRV` records before forwarding them to the other network interface. Overall, our optimization needs a high implementation effort to provide backward compatibility to existing implementations. On the other side, it provides the minimal needed length to integrate all four DNS resource records into a single DNS message (in combination with IP address reconstruction) for hardware platforms supporting very small IP payload sizes.

D. Reasonable Compromise: ADMC Enhanced

Coming back to our presented ideas for enhanced optimizations (e.g., V-B and V-C), the biggest step can be made with our redundant information filtering, but its implementation effort is very high and it introduces new dependencies at the application layer at the same time. The rest of the optimizations are very lightweight and ensure the best form of backward compatibility to existing mDNS and DNS-SD implementations, as their modifications are easily adopted. Table III summarizes these advantages and disadvantages. Furthermore, the required number of DNS messages (e.g., sent IP packets) for the Zolertia Z1 hardware platform are listed to compare each enhanced optimization with *ADMC*.

TABLE III
COMPARISON OF THE ENHANCED DNS MESSAGE COMPRESSION METHODS WITH *ADMC*.

Optimization	**Implementing Effort**	**Backward Compatibility**	**IPv6 Packet(s)**
ADMC	Low	Yes	3
6LoWPAN Compression	Lowest	Yes	2
Class Code and TTL Field Compression	Low	Yes	2
Redundant Information Filtering	Highest	No	1
ADMC Enhanced	Low	Yes	1

As backward compatibility is very important to comply with current DNS standards, we favor seamlessly integrable optimizations. The reserved pointer flags (e.g., `10` and `01`) by RFC 1035 [7, 4.1.4] can indicate our enhanced compression methods for the class code / TTL field and IP address reconstruction for their future use. In contrast, our developed redundant information filtering requires a lot more adoptions to DNS and the application layer to achieve backward compatibility. Combining the class code and TTL field compression with the IP address reconstruction, which are both summarized as *ADMC Enhanced*, will allow for a more optimized message flow of DNS in 6LoWPANs, while preserving the defined format of the four required DNS resource records. Figure 1 depicts this structural difference between *ADMC Enhanced* and the redundant information filtering as well as the backward compatibility of *ADMC Enhanced* to DNS message compression (e.g., *ADMC*). Compared to *ADMC*, the `TXT` and `AAAA` records can yet be integrated into a single DNS message while leaving 12 Bytes for the owner name / hostname and thus reducing the required number of IP packets to only two. The class code and TTL field compression will save 4 Bytes and the IP address reconstruction will save additional 14 Bytes.

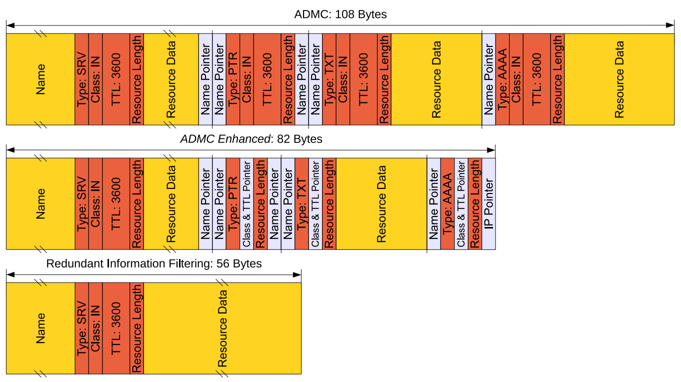

Fig. 1. Comparison of the enhanced DNS compression methods for IPv6.

Activating the 6LoWPAN compression (IPHC/NHC) allows us to send all four DNS resource records in a single DNS message, because the best compression ratio of 12 Bytes for the class code and TTL field compression can be used then. The minimal length of all four DNS resource records will hence be reduced to only 82 Bytes while the maximum available DNS payload size is extended to 106 Bytes when communicating with global addresses. For this configuration, the owner name and the hostname can have a maximum length of 24 Bytes. Overall, *ADMC Enhanced* will combine the class code and TTL field compression with the IP address reconstruction to enable an efficient and standardized DNS message transport in small object networks according to their restrictions (e.g., limited bandwidth). The backward compatibility is ensured by using the reserved pointer flags to indicate both enhanced compression methods of *ADMC Enhanced*. RFC 1035 [7, 4.1.4] has to be updated with these changes to enable the compression for existing DNS implementations.

VI. Evaluation

Our testbed for the evaluation reflects a typical IoT scenario while connecting various computational devices (e.g., smartphones, netbooks), as well as embedded devices, sensors, and actuators over IP links to support users in their daily routines. The discovery of computational devices and their respective services is provided through today's standards like *Bonjour*. For the bridging of different communication technologies, we currently work with USB adapters to link IEEE 802.15.4 and Ethernet over intermediate devices like notebooks.

A. Test Setup

All experiments were performed with Contiki on Zolertia Z1 and Tmote Sky hardware platforms, which feature an IEEE 802.15.4-compliant Chipcon 2420 RF transceiver. The test setup requires to run the 6LoWPAN border router (shipped with Contiki) on a dedicated smart object for the interaction of a computational devices with the smart object network. The border router converts 802.15.4 / 6LoWPAN frames to Ethernet / IPv6 frames. The forwarding of mDNS/DNS-SD messages to the Ethernet interface is done by Avahi [20].

B. Response Time

We measured response times by sending a `PTR` record to the multicast group and stopping the time between the request and all received DNS responses sent by the proper smart object. No optimizations and no active DNS response forwarding were implemented in uBonjour. Multi-hop routing is handled by Contiki's IP stack, depending on the performance of its used lower layers [21]. As the service discovery of uBonjour in IPv6 scenarios relies on a 6LoWPAN border router, packets will always be delayed via one-hop. The average response times for a set of multi-hop scenarios with enabled *ADMC (Enhanced)* and disabled *ADMC (Enhanced)* are listed in Table IV.

As the results proof, *ADMC* and especially *ADMC Enhanced* reduce the response time in all scenarios significantly. The highest reduction of the response time (e.g., around 59% for *ADMC Enhanced* and around 45% for *ADMC*) is reached when all required DNS records can be integrated into a single DNS message. Less IP packets and thus less IEEE 802.15.4 radio frames need to be sent while replacing all repetitions efficiently within a DNS message. Furthermore, using only DNS name compression within a DNS record reduces the response time and lowers the network traffic for low data rate smart object networks as well, because less data is transmitted per IP packet. Comparing the response times of IPv6 with IPv4 over SLIP (e.g., 71 ms without enabled *ADMC*, refer to [11, Sec. 4.4]) shows that the border router introduces a certain amount of latency for IPv6, which cannot be avoided.

With the evaluation of *ADMC (Enhanced)*, we showed that the enhanced DNS message compression methods are efficient for reducing the overhead of DNS messages in 6LoWPANs. Furthermore, the implementation of *ADMC (Enhanced)* can be used for a variety of smart object hardware platforms as *ADMC (Enhanced)* will always combine compression methods in dependency of the available IP payload size.

TABLE IV
RESPONSE TIME OF UBONJOUR IN DEPENDENCY OF THE NUMBER OF SENT IP PACKETS
– COMPARISON BETWEEN ENABLED AND DISABLED ADMC (ENHANCED) WITH ITS REDUCTION SIZE –

IPv6 Packets (Compression)	**1-Hop**	**2-Hops**	**3-Hops**	**Avg. Hop Savings**
1 (*ADMC Enhanced*)	346 ms (72%)	781 ms (60%)	1226 ms (47%)	59%
1 (*ADMC* on Tmote Sky)	600 ms (51%)	1008 ms (48%)	1488 ms (36%)	45%
3 (*ADMC*)	776 ms (37%)	1292 ms (34%)	1706 ms (27%)	32%
4 (DNS Name Compression)	1028 ms (17%)	1420 ms (27%)	2196 ms (6%)	16%
4 (No *ADMC*)	1233 ms	1954 ms	2324 ms	-

VII. FINAL REMARKS

This work promotes enhanced DNS message compression for the efficient use of mDNS/DNS-SD-based service discovery in 6LoWPANs. In the first stage, we have implemented the standardized name compression for uBonjour as *ADMC*, which can reduce the response time significantly while integrating all four DNS resource records into a single IP packet for most smart object hardware platforms (with support of an IP payload size larger than 140 Bytes). In the second stage, further optimizations were investigated to reduce the length of a single DNS message including all required DNS resource records for smart object hardware platforms with a very small IP payload size, but without breaking standards. We hence introduced two enhanced message compression methods for DNS (e.g., the class code and TTL field compression and the IP address reconstruction) and implemented both as *ADMC Enhanced* for uBonjour. Both enhanced approaches are backward compatible to existing DNS implementations and can simply be adopted by RFC 1035, because these methods are indicated by pointer flags reserved for future use. Overall, our enhanced DNS message compression demonstrates that existing standards can be used economically in 6LoWPANs without introducing smart object specific approaches.

REFERENCES

[1] J.-P. Vasseur and A. Dunkels, *Interconnecting Smart Objects with IP: The Next Internet*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010.

[2] J. W. Hui and D. E. Culler, "IP is Dead, Long Live IP for Wireless Sensor Networks," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460415

[3] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," IETF, Request for Comment 4944, Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4944

[4] F. Mattern and C. Floerkemeier, "From the Internet of Computers to the Internet of Things," in *From Active Data Management to Event-Based Systems and More*, ser. Lecture Notes in Computer Science, K. Sachs, I. Petrov, and P. Guerrero, Eds. Springer Berlin / Heidelberg, 2010, vol. 6462, pp. 242–259.

[5] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained Application Protocol (CoAP)," IETF, Internet-Draft, Mar. 2012. [Online]. Available: https://tools.ietf.org/html/draft-ietf-core-coap-09

[6] H. Tschofenig and J. Arkko, "Report from the Smart Object Workshop," IETF, Request for Comment 6574, April 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6574

[7] P. Mockapetris, "Domain Names - Implementation and Specification," IETF, Request for Comment 1035, Nov. 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1035

[8] S. Cheshire and M. Krochmal, "Multicast DNS," IETF, Internet-Draft, Dec. 2011. [Online]. Available: http://tools.ietf.org/html/draft-cheshire-dnsext-multicastdns-15

[9] ——, "DNS-Based Service Discovery," IETF, Internet-Draft, Dec. 2011. [Online]. Available: http://tools.ietf.org/html/draft-cheshire-dnsext-dns-sd-11

[10] R. Zender, U. Lucke, and D. Tavangarian, "SOA Interoperability for Large-Scale Pervasive Environments," in *Proceedings of the International Conference on Advanced Information Networking and Applications Workshops*. IEEE Computer Society, 2010, pp. 545–550.

[11] R. Klauck and M. Kirsche, "Bonjour Contiki: A Case Study of a DNS-based Discovery Service for the Internet of Things," in *Proceedings of the 11th International IEEE Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW 2012)*, ser. Lecture Notes in Computer Science (LNCS), X. Li, S. Papavassiliou, and S. Ruehrup, Eds. Springer Berlin, July 2012, vol. 7363, pp. 317–330.

[12] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN 2004)*. IEEE Computer Society, 2004, pp. 455–462.

[13] D. Barisic and A. Pfefferseder, "Unified Device Networking Protocols for Smart Objects," in *25th Workshop of Interconnecting Smart Objects with Internet*. Prague, Czech Republic: The Internet Architecture Board, March 2011. [Online]. Available: http://www.iab.org/wp-content/IAB-uploads/2011/03/Barisic.pdf

[14] R. Klauck and M. Kirsche, "Combining Mobile XMPP Entities and Cloud Services for Collaborative Post-Disaster Management in Hybrid Network Environments," *Mobile Networks and Applications - The Journal of SPECIAL ISSUES on Mobility of Systems, Users, Data and Computing*, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11036-012-0391-1

[15] M. Durvy, J. Abeille, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels, "Making Sensor Networks IPv6 Ready," in *Proceedings of the 6th ACM Conference on Networked Embedded Sensor Systems (SenSys)*, Nov. 2008.

[16] S. Duquennoy, F. Österlind, and A. Dunkels, "Lossy Links, Low Power, High Throughput," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '11. New York, NY, USA: ACM, 2011, pp. 12–25. [Online]. Available: http://doi.acm.org/10.1145/2070942.2070945

[17] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," IETF, Request for Comment 6282, Sep. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6282

[18] J. Hui and D. Culler, "6LoWPAN: Incorporating IEEE 802.15.4 into the IP Architecture," Internet Protocol for Smart Objects (IPSO) Alliance, White paper #3, Tech. Rep., Jan. 2009. [Online]. Available: http://www.cs.berkeley.edu/~jwhui/6lowpan/IPSO-WP-3.pdf

[19] M. Crawford, "Transmission of IPv6 Packets over Ethernet Networks," IETF, Request for Comment 2464, Dec. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2464

[20] The Avahi Team, "More About Avahi - Details about mDNS, DS-DNS and Zeroconf," [Online] http://avahi.org/wiki/AboutAvahi, 2012.

[21] J. Silva, T. Camilo, P. Pinto, R. Ruivo, A. Rodrigues, F. Gaudêncio, and F. Boavida, "Multicast and IP Multicast Support in Wireless Sensor Networks," *Journal of Networks*, vol. 3, no. 3, pp. 19–26, Mar. 2008.