Lightweight Multicast Forwarding for Service Discovery in Low-power IoT Networks

Mattia Antonini*, Simone Cirani*, Gianluigi Ferrari*, Paolo Medagliani†, Marco Picone* and Luca Veltri*
*Department of Information Engineering
University of Parma,
Viale G.P. Usberti, 181/A, 43126 - Parma (Italy)
Email: mattia.antonini1@studenti.unipr.it, [simone.cirani, gianluigi.ferrari, marco.picone, luca.veltri]@unipr.it
†Thales Communications and Security
4 Avenue des Louvresses, 92622, Gennevilliers, France
Email: paolo.medagliani@thalesgroup.com

***Abstract*—The Internet of Things (IoT) will interconnect billions of devices (denoted as "Smart Objects," SOs) in an IP-based Internet-like structure. SOs are typically sensor/actuator-equipped devices with severe constraints on processing capabilities, available RAM/ROM, and energy consumption. In a context where billions of deployed SOs, it is important that the SOs are able to self-configure and adapt to the surrounding environment with minimal, if any, external human intervention. Among the service discovery mechanisms proposed in literature for deploying SOs without any prior knowledge, Zeroconf represents a good candidate to automate service and resource discovery in local constrained environments. In this paper, we propose a lightweight forwarding algorithm for efficient multicast support in Low-power and Lossy Networks (LLNs) targeting service discovery for duty-cycled SOs. Among the advantages achieved by the proposed solution, SOs might benefit from smaller memory footprint with respect to those required by other multicast implementations. The performance of the proposed forwarding algorithm is evaluated through Contiki-based nodes in the Cooja simulator.**

I. Introduction

The Internet of Things (IoT) will bring together more than 50 billions of heterogeneous devices by 2020, interconnecting them in a global Internet-like network of networks. The foreseen use of an IP-based protocol stack for IoT devices will allow them to connect seamlessly to standard Internet hosts enabling new applications and forms of interactions between machines and humans. The IoT will therefore include nodes with different features, in terms of processing and communication capabilities, power supply, and energy consumption, spanning from constrained devices, also referred to as "smart objects" (SOs), to smartphones and other personal devices, Internet hosts, up to the Cloud. The majority of deployed IoT devices will likely operate in Low-power and Lossy Networks (LLNs). In this context, pushing IP to SOs requires to adapt running protocols to constrained environments [1]. Some working groups of the IETF have contributed to the standardization of protocols suitable for LLNs, such as the IETF 6LoWPAN Working Group or the IETF ROLL Working Group. The former group introduced 6LoWPAN, a header compression protocol that allows to send IPv6 frames over IEEE 802.15.4 networks. The latter, instead, defined RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) [2], a lightweight routing protocol for LLNs.

A very important aspect for the successful deployment of IoT applications is the capability of nodes to self-configure and adapt to the surrounding environment without the need of prior configuration or human intervention, enabling long-lasting robust and fault-tolerant applications suitable for highly dynamic scenarios. A major requirement for self-configuration is service discovery, i.e., the ability of a node to become aware of the presence of some services or resources in a given network, and the endpoints to use to get access. Several techniques for service discovery have already been defined in the literature. Zeroconf [3] is a protocol suite which reuses the semantics of Domain Name System (DNS) messages over IP multicast to provide name resolution and service discovery/advertisement over local networks. In order to support Zeroconf service discovery mechanisms, it is very important that the network supports IP multicasting and implements proper forwarding techniques to guarantee that packets are delivered to all group nodes and avoid the establishment of loops. Using efficient packet forwarding mechanisms can bring benefits in multihop communications among SOs, in terms of delay and energy consumption. Moreover, it is also important to remark that the limited amount of memory available on SOs requires the adoption of small footprint mechanisms, in order to allow developers to integrate a complete software stack, without the need to sacrifice some modules in order to abide by the practical memory constraints. Although the IETF ROLL working group is defining a Multicast Protocol for Low power and Lossy Networks (MPL) [4], based on the Trickle algorithm [5], some applications might have different requirements and could benefit by the adoption of other multicast techniques.

In this work, we propose a lightweight and low-power multicast forwarding protocol targeted for service discovery in SOs operating in IEEE 802.15.4 multihop networks. The proposed solution features a smaller memory footprint with respect to those of other state-of-the-art solutions. The proposed mechanism has been implemented on Contiki OS-enabled SOs. An extensive experimentation is carried out in the Cooja simulator in order to evaluate the feasibility and efficiency, in terms of delay and energy consumption, of the proposed mechanism.

The rest of this paper is organized as follows. In Section II, we present related works. In Section III, the details of the proposed multicast forwarding strategy for service discovery are

shown. Implementation and experimental results are discussed in Section IV. Finally, in Section V we draw our conclusions.

II. Related Work

Local service discovery mechanisms in Local Area Networks (LANs) have been proposed in the literature. Protocols like Universal Plug and Play (UPnP) [6] and Service Location Protocol (SLP) [7], [8] focus on automatic announcement and discovery of in-network existing services. However, their porting to IoT devices is not straightforward because of the severe computation and energy constraints of the nodes. An alternative to these protocols relies on multicast forwarding. For instance, in [9] the authors propose an efficient group communication strategy for the CoAP and the Efficient XML Interchange (EXI) protocols. To achieve group communication, they rely on the Open Building Information eXchange (oBIX) standard. However, this implementation runs on Raspberry PI nodes, so it is not suitable for constrained devices.

Concerning 6LoWPAN and IPv6, the only active IETF draft on efficient multicast forwarding is the Multicast Protocol for Low power and lossy networks (MPL) [4], that relies on the Trickle algorithm to manage transmissions for both control and data plane. The different multicast interfaces, identified by an unicast address and associated with one or more multicast domains, are handled separately to maintain an independent seed set to decide whether to accept a packet or not. The MPL forwarder, which is in charge of sending data messages, has two different possible strategies: i) proactive or ii) reactive. In the former case, the MPL Forwarder schedules the transmission of MPL Data Messages using the Trickle algorithm, without any prior indication that neighbor nodes are yet to receive the message. After transmitting a limited number of MPL Data Messages, the MPL Forwarder may terminate proactive forwarding for the MPL Data Message. In the latter, the MPL Forwarder sends link-local multicast MPL Control Messages using the Trickle algorithm. MPL Forwarders use MPL Control Messages to discover new MPL Data Messages that have not yet been received. When an MPL Forwarder discovers that a neighbor MPL Forwarder has not yet received an MPL Data Message, it schedules the transmission of those MPL Data Messages using the Trickle algorithm. The two approaches can coexist at the same time.

In [10], the authors propose Stateless Multicast RPL Forwarding (SMRF), which relies on the presence of the RPL routing protocol. This approach is based on a specific routing protocol and requires group management information to be carried inside DAO messages. However, since for our goal a less complicated multicast strategy (no group management is required) is needed, we prefer to rely on a more lightweight flooding technique, which adapts well to duty-cycled devices operating in RPL networks implementing the Zeroconf protocol suite.

In [11], the authors present two tree-based data broadcast strategies for Zigbee networks. Even if these solutions are interesting for information dissemination, their approach loses generality since they rely on the specific hierarchical Zigbee addressing space.

An approach different from group communications in constrained networks is offered by featurecast [12]. The goal of this implementation is data dissemination to devices offering specific features. The result of this approach is a reduced memory occupation compared to standard IP group communication. This solution addresses more the one-to-many paradigm, whereas we are more interested in a many-to-many communication in order to implement a mDNS-compatible local service discovery. In [13], the authors present an opportunistic flooding mechanism tailored for low-duty-cycle networks with unreliable wireless links and predetermined working schedules. Unlike this approach, our strategy aims at allowing efficient data propagation without taking into account any predefined scheduling.

III. Efficient Forwarding Protocol for Service Discovery

Zeroconf is a protocol that allows to automatically create computer networks based on the TCP/IP Internet stack and does not require any external configuration [3]. Zeroconf provides three main functionalities: i) automatic network address assignment; ii) automatic distribution and resolution of host names; iii) automatic location of network services. Automatic network assignment comes into the picture when a node first connects to the network. The host name distribution and resolution is implemented using multicast DNS (mDNS) [14], a service that has the same interfaces, packet formats, and semantic of the standard DNS to resolve host names in networks that do not include a local name server. Zeroconf also allows to publish services (DNS-SD) in a local network. Both mDNS and DNS-SD do not require the presence of any server (and, therefore, its knowledge) to perform publish, lookup, and name resolution operations, but rely on the use of IP multicast communications in order to address all the nodes in the local network. Zeroconf specifies that mDNS and DNS-SD messages (for both requests and responses) must be sent to the mDNS IPv4/IPv6 link-local multicast address (i.e., *224.0.0.251* and *ff02::fb*, respectively). However, Zeroconf does not require per-group multicast routing: according to the protocol specifications, messages should simply reach all nodes in the local network. 6LoWPAN defines methods i) to transmit IPv6 packets and ii) to form IPv6 link-local addresses and statelessly autoconfigured addresses on IEEE 802.15.4 networks. The RPL protocol defines a routing protocol for IP communications in LLNs. The IETF ROLL Working Group is working on the definition of MPL, a multicast protocol providing IPv6 multicast forwarding in constrained networks, which could represent a general multicast technique able to manage multicast groups of any size. However, in some scenarios, such as Zeroconf service discovery, there is no need to actually adopt such a full-feature multicast protocol. For the sake of Zeroconf service discovery, it is sufficient to provide a multicast forwarding mechanism which guarantees that messages can be delivered to all nodes in the local network. In this section, we detail a simple and efficient forwarding algorithm that can be adopted by constrained devices operating in RPL networks with ContikiMAC radio duty-cycling protocol, in order to enable IP multicast communications with a small footprint, targeting Zeroconf service discovery.

A. Multicast through Local Filtered Flooding

Flooding is the simplest routing protocol that can be adopted to broadcast a packet to all nodes in the network.

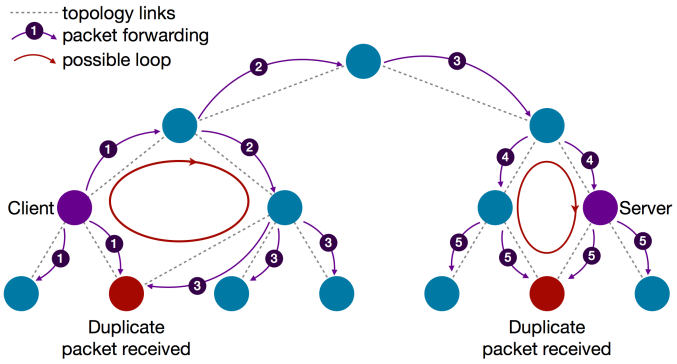


Fig. 1: Flooding of a DNS-SD query in generic topology with cycles.

From a practical implementation point of view, each node forwards a received packet to all its neighbors. This technique is effective only in case of cycle-free topologies (i.e., trees). In the presence of graphs with cycles, it is necessary to implement duplicate detection techniques to avoid forward loops. An illustrative representation is shown in Fig. 1.

In order to implement an efficient mechanism to detect already-processed packets (and, thus, avoid redundant forwarding), we propose the adoption of Bloom filters [15]. Bloom filters are probabilistic data structures that can be used to add elements to a set and to check efficiently whether an element belongs to the set or not. Bloom filters provide two primitives: i) *add(x)*: add element x to the set; and ii) *query(x)*: test to check whether element x is in the set. The filter does not provide a *remove(x)* primitive so it is not possible to undo an insertion. Compared to other equivalent probabilistic data structures (in terms of provided functionalities), such as Quotient filters [16], Bloom filters are slower when performing *check* operations but have a smaller memory occupation. As available memory on SOs is extremely limited, one of the design goals of the proposed forwarding algorithm is to keep memory footprint (both in terms of RAM and ROM) as low as possible. Therefore, Bloom filters have been selected as the most appropriate data structure to keep track of already-processed packets.

A Bloom filter is initially an array of m bits all set to zero. The *add(x)* operation consists in passing the input argument x through k different hashing functions and obtain k indexes in the bits array of the Bloom filter that are going to be set to one. The *query(x)* operation verifies whether the indexes corresponding to x are all set to one. The Bloom filter is probabilistic in the sense that a *query(x)* operation can return false positives: there can exist two values x_1 and x_2, such that *query*(x_1) = *query*(x_2) = *true*. False negatives, on the other hand, are not possible: this means that if a *query(x)* returns *false*, then x is not in filter. The *query(x)* operation can thus return either *probably in the set* or *not in the set*.

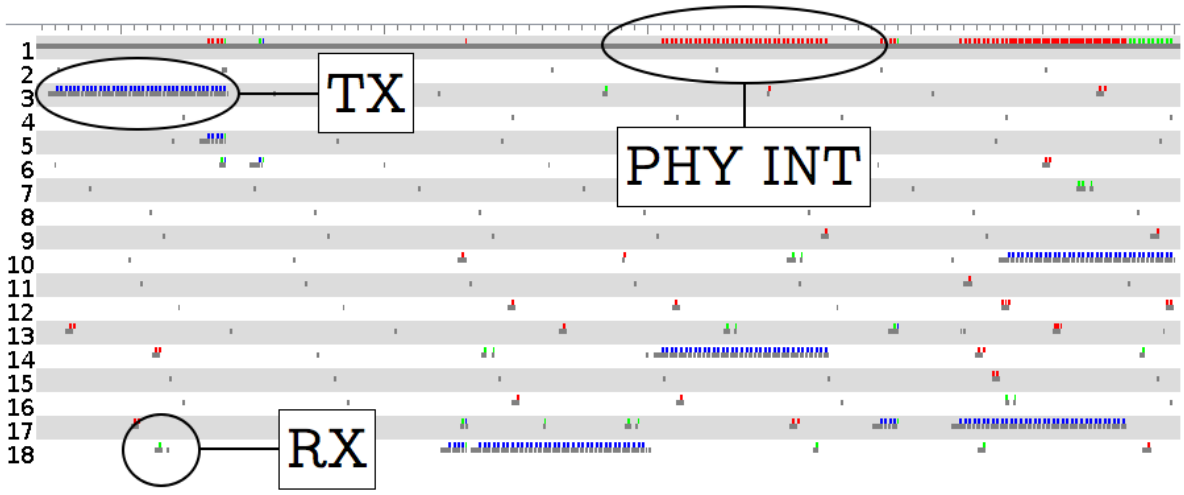
Bloom filters can be instantiated to meet specific application requirements by selecting the parameters m (number of bits in the array) and k (number of hashing functions). For instance, the choice of m and k has an impact on the probability of getting false positives for *query(x)* operations and on memory occupation. In any case, the impossibility to remove an element from the filter leads to an increase on the probability of false positives as more and more elements are added to the filter. Since the purpose of using a Bloom filter in the forwarding algorithm is to detect duplicate elements, in order to cope with the problem of false positives, the Bloom filter is periodically reset. Resetting the filter might introduce some unnecessary retransmissions, if the filter is emptied before receiving a duplicate packet. However, retransmissions are preferable to packet drops in order to guarantee that a multicast packet reaches all hosts. Moreover, such unnecessary retransmissions might occur no more than once, as the packet would than be added to the filter and not processed upon future receptions. To summarize, upon receiving a packet, a node will perform the following steps:

1) check if the incoming IP packet has already been processed, by performing a *query* operation in the Bloom filter;
2) if the Bloom filter contains the packet, discard it; otherwise, the packet is added to the Bloom filter through an *add* operation;
3) if needed, forward the received IP packet to all neighbors by means of local IEEE 802.15.4 broadcast.

B. Efficient Multiple Unicast Forwarding

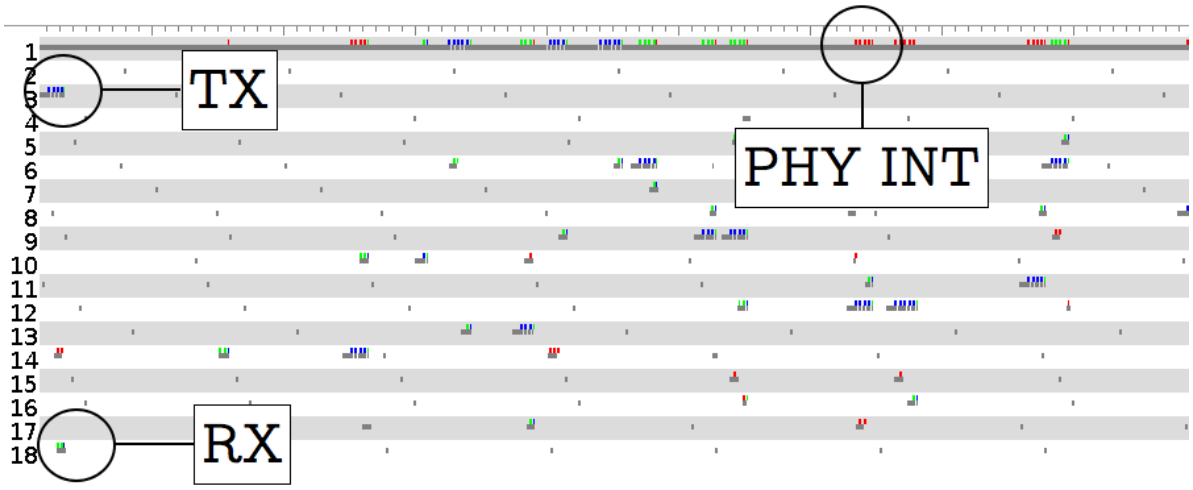
While the described algorithm implements an optimized flooding mechanism by avoiding loops owing to the introduction of Bloom filters, the use of broadcasting with the ContikiMAC radio duty-cycling protocol results in inefficient transmissions, which lead to higher energy consumption and end-to-end delays. In fact, in ContikiMAC, a broadcasting node must repeatedly transmit a packet for the full wake-up interval [17], in order to ensure that the transmitted packet can be received by all neighbor nodes, regardless of their wake-up time. This conservative approach has the following drawbacks: i) the number of transmitted packets is larger than necessary, and, therefore, energy consumption is higher; ii) when a node is broadcasting a packet, other nodes are not allowed to transmit, and this delays the transmission until the channel is clear; and iii) since ContikiMAC broadcasting does not make provision to acknowledge received packets, it might happen that not all neighbors have successfully received the packet, thus leading to unreliable transmission. These inefficiencies are magnified when the Channel Check Rate (CCR) decreases, since the full wake-up interval is longer and therefore the channel is busy for a longer time, thus leading to even more repeated transmissions and delays. This contrasts with the assumption that lower CCR leads to lower energy consumption. In order to tackle these issues, we replace local broadcast with multiple unicast transmission. The forwarding algorithm can therefore be optimized by selecting the receiving nodes from the list of next hops, retrieved from the RPL routing table. In fact, ContikiMAC provides per-node-pair synchronization, which ensures that packets are sent only when the receiver is supposed to be active and the receiver is required to send an acknowledgement for the received packet, thus transmitting packets only as long as necessary and leading to more reliable transmissions.

The enhanced version of the proposed multicast protocol can therefore be detailed as follows:



(a)

Fig. 2: DNS-SD query propagation DNS-SD query propagation with ContikiMAC broadcast. Time is on the x-axis while node identifiers are on the y-axis.



(a)

Fig. 3: DNS-SD query propagation with multi-unicast. Time is on the x-axis while node identifiers are on the y-axis.

1) check if the incoming IP packet has been processed already by performing a *query* operation in the Bloom filter;
2) if the Bloom filter contains the packet, discard it; otherwise, add the packet to the Bloom filter through an *add* operation;
3) retrieve the list of next hops from the routing table;
4) if needed, forward the received IP packet to each next hop using IEEE 802.15.4 unicast communication.

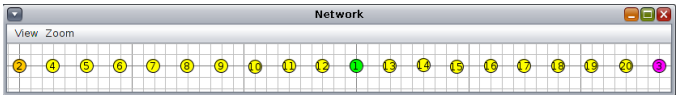
An excerpt of a sequence of transmitted frames, using broadcast for a DNS-SD query, is shown in Fig. 2.

The equivalent packet flooding with multiple unicast transmissions is shown in Fig. 3. Transmitted packets (TX), received packets (RX), and PHY interference (PHY INT) are highlighted. The root of the RPL tree (node 1) is always active, while all other nodes have CCR = 8 Hz. The timelines clearly show that multiple unicast transmissions optimize i) the number of transmitted packets and ii) the packet propagation delay in the network, while guaranteeing more reliable transmissions, at the cost of a slightly increased ROM/RAM footprint.

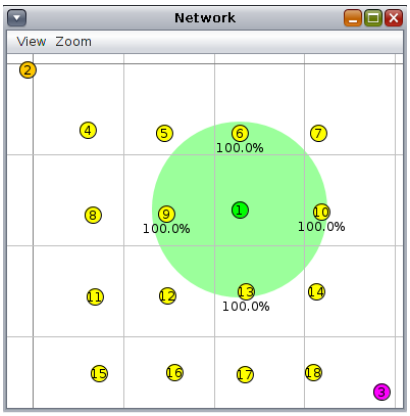
IV. Implementation Results

In order to evaluate the performance of the proposed multicast packet forwarding mechanism, a Contiki-based implementation has been developed. Besides the proposed multicast forwarding algorithm, the mDNS and DNS-SD protocols have been re-implemented, in order to have a smaller memory footprint than those of other already available implementations. The performance evaluation of the Zeroconf-based local service discovery strategy has been conducted using WiSMote[1] Contiki nodes, simulated in the Cooja simulator. The Contiki

[1] http://wismote.org/



(a)



(b)

Fig. 4: Linear topology (a) and grid topology (b) considered for Zeroconf service discovery experimentation with the proposed multicast protocol.

Library	ROM Occupation [B]	Occupation of overall available ROM
This work (broadcast)	842	0.64%
This work (multiple unicast)	1454	1.11%
MPL with Trickle	3804	2.90%

TABLE I: Comparison of memory occupation (dimension: [B]), in terms of ROM, for the proposed multicast protocol and MPL implementation. Comparison of memory occupation, in terms of percentage of occupied ROM with respect to overall available memory on the considered 128-kB WiSMote, for the proposed multicast protocol and MPL implementation.

software stack running on each node has been configured in order to fit in the WiSMote's available memory, in terms of both RAM and ROM — WiSMote nodes feature nominal 128 kB, 192 kB or 256 kB ROM and a 16 kB RAM. The simulated smart objects run Contiki OS, uIPv6, RPL, ContikiMAC.

The local service discovery mechanism has been tested on Contiki nodes arranged in linear and grid topologies, shown in Fig. 4, in IEEE 802.15.4 networks with RPL as routing protocol and ContikiMAC as radio duty-cycling protocol. In particular, node 1 is the 6LoWPAN Border Router (6LBR), which is the root of the RPL tree; node 2 is the node acting as DNS-SD server; and node 3 is the node acting as DNS-SD client. The first performance indicator is memory occupation (dimension: [B]), in terms of ROM. The proposed multicast protocol — both with broadcast and multiple unicast transmissions — is compared with the MPL (with Trickle algorithm) implementation available in the Contiki 3.x fork. The results are shown in Table I.

As expected, the footprint of the proposed solution is significantly smaller than that of the MPL implementation available on the Contiki 3.x fork, approximately of 78% (broadcast) and 62% (multiple unicast), respectively.

The next phase of experimentation aims at evaluating the time and the overall network energy consumption needed to perform the advertisement and the resolution of services using Zeroconf in the topologies shown in Fig. 4, using the proposed broadcast-based and multiple unicast-based approaches. All the results have been obtained by performing 100 service discovery runs on each configuration. The specific performance metrics are: i) Query Client time (QC), which is the time needed by a node acting as client to send a DNS-SD query and receive a response (dimension: [ms]); and ii) Energy consumption (E), which is the overall network energy consumption for a DNS-SD query operation (dimension: [mJ]). The impact of the CCR (varying in a range from 8 Hz to 128 Hz) of the nodes participating in the constrained network is analyzed. The results for QC and E are shown in Fig. 5 and Fig. 6, respectively. It is possible to observe that, as expected, QC is inversely proportional to the CCR. The benefit of using multiple unicast transmissions, instead of broadcast, is higher when the CCR is low, while the two approaches tend to overlap for higher values of the CCR. At typical CCR values, multiple unicast performs better than broadcast because, with lower CCR, the wake-up interval is longer and ContikiMAC broadcast transmissions occupy the channel for the whole interval. Higher CCR values mean shorter wake-up intervals and, therefore, other nodes in the network are likely to be blocked. In the case of the grid topology, with a CCR of 128Hz, broadcast actually is slightly faster than unicast. In fact, at this high rate, the smart objects are almost behaving as with null duty-cycling, which is the best-case scenario for broadcast transmissions. As for energy consumption, the results clearly show that the use of broadcast is much more inefficient than that of multiple unicast transmissions. It is important to point out that a significant contribution (more than 60% with multiple unicast and 30% with broadcast, at CCR = 8 Hz) to the energy consumption of the overall network is due to the border router, which does not perform duty-cycling. Again, this is motivated by the ContikiMAC broadcast strategy, which requires nodes to transmit for the whole wake-up interval. As for QC, the two approaches tend to overlap at high CCR. However, these results should not be interpreted as a suggestion to use higher CCR, as this would invalidate all the advantages of duty-cycling, which is particularly beneficial in other scenarios.

V. Conclusions

In this work, we presented a novel multicast forwarding mechanism targeting service discovery in IoT scenarios. The proposed solution is suited to bring efficient IP-multicast support to low-power IoT networks with duty-cycled devices. The rationale behind the presented forwarding mechanism is the will to rely on a lightweight and with low memory footprint implementation for specific Zeroconf service discovery operations. In such scenarios, the adoption of a full-featured multicast implementation, such as MPL, might be overkill, since there is no need to provide multicast group support but, rather, an efficient flooding mechanisms in the local network. The proposed multicast protocol relies on filtered local flooding, which adapts well to duty-cycled devices operating in LLNs with RPL. In order to avoid forward loops, we introduce Bloom filters, an efficient probabilistic data structure,

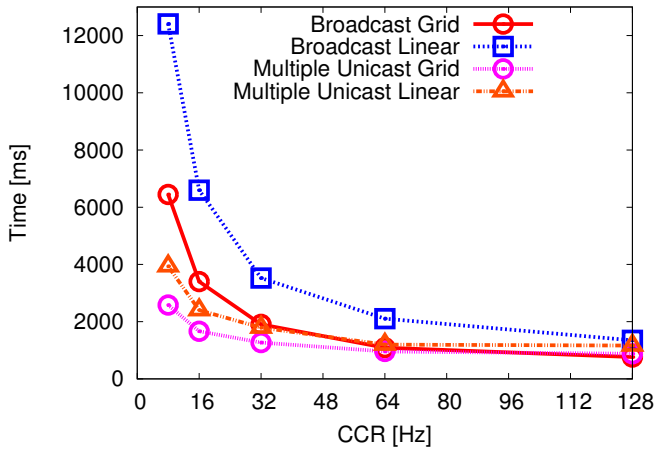


Fig. 5: Time (dimension: [ms]) needed to perform a DNS-SD query by a client in linear and grid topologies with broadcast and multiple unicast.

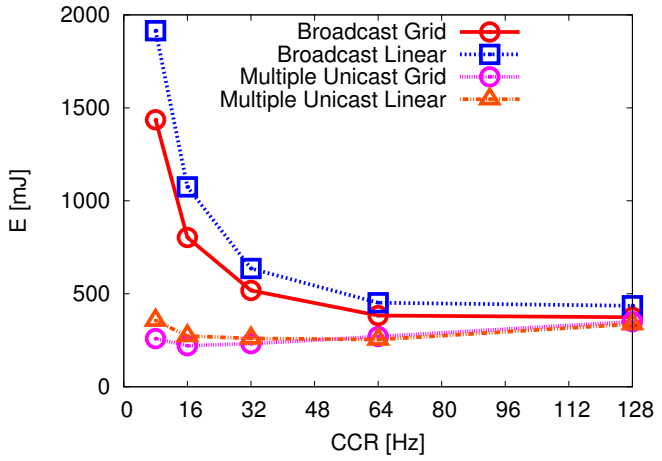


Fig. 6: Overall network energy consumption (dimension: [mJ]) needed to perform a DNS-SD query by a client in linear and grid topologies with broadcast and multiple unicast.

to detect duplicate packets and prevent forward loops. The experimental results demonstrate that the proposed multicast protocol features a much smaller footprint, in terms of ROM occupation, compared to the MPL implementation available in the official Contiki fork. Finally, delay and network energy consumption have been evaluated.

Acknowledgment

The work of Simone Cirani, Gianluigi Ferrari, and Luca Veltri is funded by the European Community's Seventh Framework Programme, area "Internetconnected Objects", under Grant no. 288879, CALIPSO project - Connect All *IP*-based Smart Objects! The work reflects only the authors views; the European Community is not liable for any use that may be made of the information contained herein. The work of Paolo Medagliani is supported by the French National Research Agency (ANR) under grant reference IRIS ANR-11-INFR-0016. The work of Marco Picone is supported by Guglielmo srl, Reggio Emilia, Italy.

References

[1] "Connect All IP-based Smart Objects (CALIPSO) - FP7 EU Project." [Online]. Available: http://www.ict-calipso.eu/

[2] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[3] Zeroconf Website. (1999) http://www.zeroconf.org/.

[4] J. Hui and R. Kelsey, "Multicast Protocol for Low power and Lossy Networks (MPL)," Internet Engineering Task Force, Internet-Draft draft-ietf-roll-trickle-mcast-09, Apr. 2014. [Online]. Available: http://tools.ietf.org/html/draft-ietf-roll-trickle-mcast-09

[5] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, ser. NSDI'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 2–2. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251175.1251177

[6] UPnP Forums. (1999) http://www.upnp.org/.

[7] E. Guttman, C. Perkins, and J. Veizades, "Service Location Protocol, Version 2," Internet Engineering Task Force, RFC 2608, Jun. 1999. [Online]. Available: http://tools.ietf.org/html/rfc2608

[8] E. Guttman, "Vendor Extensions for Service Location Protocol, Version 2," Internet Engineering Task Force, RFC 3224, Jan. 2002. [Online]. Available: http://tools.ietf.org/html/rfc3224

[9] M. Jung and W. Kastner, "Efficient group communication based on web services for reliable control in wireless automation," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, Nov 2013, pp. 5716–5722.

[10] G. Oikonomou and I. Phillips, "Stateless multicast forwarding with rpl in 6lowpan sensor networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, March 2012, pp. 272–277.

[11] G. Ding, Z. Sahinoglu, P. Orlik, J. Zhang, and B. Bhargava, "Tree-Based Data Broadcast in IEEE 802.15.4 and ZigBee Networks," *IEEE Trans. on Mobile Computing*, vol. 5, no. 11, pp. 1561–1574, Nov 2006.

[12] M. Krol, F. Rousseau, and A. Duda, "Featurecast: Lightweight data-centric communications for wireless sensor networks," Submitted for publication, 2014.

[13] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '09. New York, NY, USA: ACM, 2009, pp. 133–144. [Online]. Available: http://doi.acm.org/10.1145/1614320.1614336

[14] S. Cheshire and M. Krochmal, "Multicast DNS," Internet Engineering Task Force, RFC 6762, Feb. 2013. [Online]. Available: http://tools.ietf.org/html/rfc6762

[15] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970. [Online]. Available: http://doi.acm.org/10.1145/362686.362692

[16] M. A. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, D. Medjedovic, P. Montes, P. Shetty, R. P. Spillane, and E. Zadok, "Don't thrash: How to cache your hash on flash," in *Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems*, ser. HotStorage'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 1–1. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002218.2002219

[17] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," Swedish Institute of Computer Science, Tech. Rep. T2011:13, Dec. 2011. [Online]. Available: http://dunkels.com/adam/dunkels11contikimac.pdf