Adaptive and Context-aware Service Discovery for the Internet of Things

Talal Ashraf Butt[1], Iain Phillips[1], Lin Guan[1], and George Oikonomou[2]

[1] Department of Computer Science, Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK
{T.A.Butt,I.W.Phillips,L.Guan}@lboro.ac.uk

[2] University of Bristol, Faculty of Engineering, Merchant Venturers Building, Woodland Road, Clifton BS8 1UB, UK
g.oikonomou@bristol.ac.uk

Abstract. The Internet of Things (IoT) vision foresees a future Internet encompassing the realm of smart physical objects, which offer hosted functionality as services. The role of service discovery is crucial when providing application-level, end-to-end integration. In this paper, we propose TRENDY: a RESTful web services based Service Discovery protocol to tackle the challenges posed by constrained domains while offering the required interoperability. It provides a service selection technique to offer the appropriate service to the user application depending on the available context information of user and services. Furthermore, it employs a demand-based adaptive timer and caching mechanism to reduce the communication overhead and to decrease the service invocation delay. TRENDY's grouping technique creates location-based teams of nodes to offer service composition. Our simulation results show that the employed techniques reduce the control packet overhead, service invocation delay and energy consumption. In addition, the grouping technique provides the foundation for group-based service mash-ups and localises control traffic to improve scalability.

Keywords: Adaptive, Context-aware, Service Discovery, 6LoWPAN, Internet of Things, CoAP, RESTful, Web of Things

1 Introduction

The Internet of Things (IoT) concept has revolutionised the vision of the future Internet with the advent of standards such as 6LoWPAN making it feasible to extend the Internet into previously unreachable environments, e.g. Wireless Sensor Networks (WSN). The abstraction of resources as services, has opened WSNs to a new plethora of potential applications. Moreover, the web service paradigm can be used to provide interoperability by offering a standard interface to interact with these services. However, these networks pose many challenges in terms of limited resources. Consequently, the adaptability of existing IP-based solutions is not feasible. As traditional service discovery and selection solutions demand heavy communication and use bulky formats, which are unsuitable for these resource-constrained devices incorporating sleep cycles to save energy. Even a registry-based approach exhibits burdensome traffic in maintaining the availability status of the devices. The feasible solution for service discovery and selection is instrumental in enabling wide application coverage of these networks in the future [1].

The contribution of this paper is a compact and optimise registry based service discovery solution with context awareness for the IoT, which is more focused on constrained domains such as 6LoWPAN. It uses CoAP-based [14] RESTful web services to provide a standard interoperable interface which can be easily inter-worked with HTTP. The modular design of protocol features allows its implementation on the constrained devices. High capability devices can benefit by implementing profiles to share the load of other devices. Thus, it allows the productive usage of resources in the network. TRENDY intelligently uses the context information to provide optimal service selection, which make sure that more superior hosts will be suggested to an enquirer. This paper extends our previous work [3] by introducing new adaptive timer and caching techniques. Adaptive timer minimises the protocol's control overhead and energy consumption. Its grouping mechanism is based on location tags to localise status maintenance traffic and to compose and offer new group based services. The APPUB (Adaptive Piggybacked Publish) technique balances the trade-off between service invocation delay and packet overhead by adaptively making cache available for highly requested resources. We have performed simulations to demonstrate the benefit of using TRENDY techniques in terms of energy consumption, packet overhead, scalability (packets towards the sink and cache hits) and service invocation time.

This paper covers the related work in Section 2, before describing protocol and its architecture, entity interactions and techniques in Section 3. In the end, Section 4 discusses the performed experiments and generated results.

2 Related Work

The service discovery protocols are generally classified into three broad categories on an architectural basis: *centralised*, *distributed* and *hierarchical* [2]. Centralised architectures have a directory, where Service Agents (SA) register their services. Subsequently, User Agents (UA) discover the services by sending unicast queries to the directory. On the other hand, distributed architectures demand that nodes collaborate using broadcast or multicast to discover a service. An example of a distributed service discovery mechanism is ADDER [12]. Hierarchical architectures employ some nodes with high capabilities, to represent a cluster of nodes in their vicinity.

The industry-standard, IP-based Service Discovery Protocols (SDP) including SLP, UPnP, JINI and Salutation are not directly applicable to 6LoWPAN because of the employed complex formats and high communication demand. uBonjour [8] is bonjour's compact variant, based on mDNS and DNS-SD. Even though mDNS/DNS-SD message sizes were recently optimised for 6LoWPANs [9], uBonjour still relies on the availability of IP multicast and entails more communication

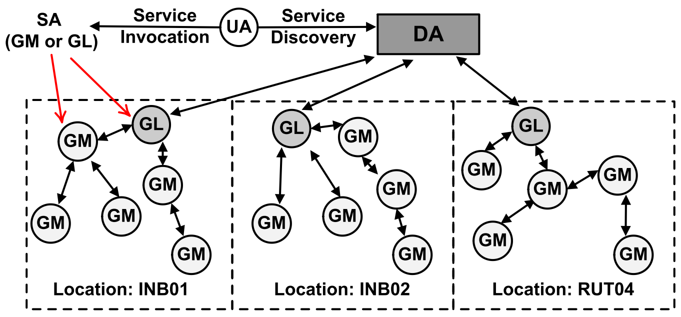


Fig. 1: Hybrid architecture of TRENDY

overhead. A SLP adaptation approach [4] employs SSLP inside the 6LoWPAN and provides interoperability with SLP by using a Translation agent (TA). However, this solution involves complexity and delay of translation; each time message is translated to or from SLP. An industry focused SOA (Service Oriented Architecture) based middle-ware solution [7] uses WS-* and RESTful web services. A recent approach [10] provides RESTful web services using HTTP based service discovery with existing or injected strategies. The IETF Resource directory [15] uses CoAP as an underlying communication protocol for service discovery. However, most of these existing directory-based solutions do not address the service discovery requirements of IoT environments. This paper proposes TRENDY service discovery solution that provides context-aware discovery, efficient service management, service selection, caching and service composition.

3 TRENDY: Trend-based Service Discovery for the IoT

This section describes the various design aspects of the TRENDY service discovery protocol, including architecture, interaction between entities in context of protocol features, adaptive timer and caching techniques.

3.1 Architecture

TRENDY maintains a registry: the DA (Directory Agent), where SAs (Service Agents) register services. UAs (User Agents) query the DA, to find the location of a service. The grouping mechanism further categorises SAs into GLs (Group Leaders) and GMs (Group Members). Fig. 1 presents TRENDY's architecture.

Directory Agent (DA): The DA has a backbone role in TRENDY's architecture maintaining the registry and using a demand-based adaptive timer (Section 3.3) to increase or decrease the interval between status maintenance updates. The DA responds to service discovery requests and uses collected context information to provide optimal service selection. In case of a constrained network, e.g. 6LoWPAN, it can be at the root of RPL (IPv6 Routing

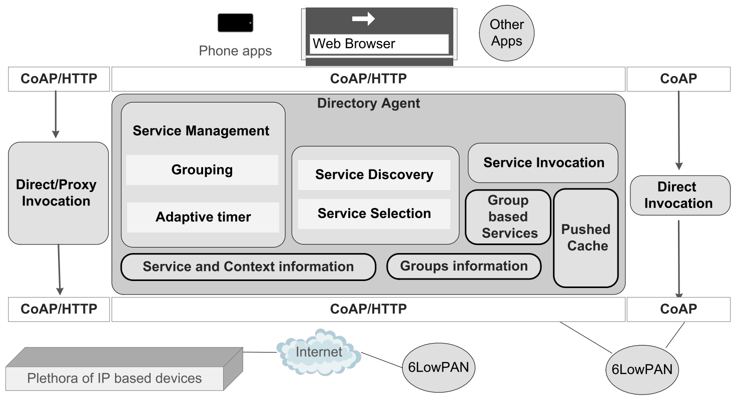


Fig. 2: Interoperable framework of TRENDY: The DA collects service and context information to provide service discovery and selection, and uses grouping and adaptive timer mechanisms to reduce status maintenance traffic.

Protocol for Low power and Lossy Networks) routing protocol. However, any other routing protocol can be used for this purpose. In our work, the DA role is embedded in an edge router that acts as a bridge between the WSN and IP networks using adaptation layer. However, the only requirement is that the DA is on a resource-rich node with IP reachability to the SAs.

Group Member (GM): This is the most basic entity of TRENDY, and represents a service host who registers its services with the DA. Furthermore, it periodically sends status updates to the DA using TRENDY's UPD (Update) message by selecting a random interval of 50% to 100% of the DA's time window between messages.

Group Leader (GL): The GL plays a key role in the grouping mechanism. TRENDY's modular design allows a GL to choose a different feature-set depending on its available resources and application needs. The responsibilities of a GL depend on its implemented resources; it can just collect the status updates, forwards the query to group, aggregate the results for a query or can act like a local registry or proxy.

User Agent (UA): The UA is a client that dicovers services available in a network by sending queries to the DA. It can be either part of the sensor network or can send a request from elsewhere in the Internet.

3.2 Entity Interaction

TRENDY introduces an open and interoperable framework to deal with the diversity of networks that can be the IoT. The challenge posed by the IoT's requirements is managed by employing a layered architecture, intelligent DA and enabling a RESTful web service paradigm to deal with various formats and protocols as shown in Figure 2.

This section covers the detail of TRENDY's features from the perspective of different entities.

Web Service Paradigm: TRENDY uses a RESTful web service paradigm. Entities use either CoAP (default) or HTTP (in case the targeted host understands it or DA is acting as a proxy) to define their services and to communicate with each other. The use of CoAP/HTTP simple proxy can seamlessly translate requests from both protocols. This blends the real-world devices into the existing web and enables the Web of Things (WoT) paradigm.

Context-awareness: In TRENDY, the DA stores all service and contextual information, including service descriptions, location, battery consumed, and registration time for all registered nodes. Furthermore, it maintains a hit counter for each service, which is incremented whenever a service is discovered and selected. The grouping, optimal GL and service selection is based on the available context information. TRENDY allows the use of any context attributes in an attribute-value pair format separated by "=". These context attributes can be defined in the service description by adding "," to separate them e.g. "`l=INB01,b=10`" describes a host's location and battery attribute.

Grouping: Context-based grouping serves several purposes, including simple localisation of status maintenance, execution of group-based queries to offer an optional local service repository. It costs in terms of some packet overhead. However, networks can get the benefit in the form of localised communication, which conserves energy. In addition, this enables a DA to compose and offer group-based services, e.g. to actuate a command in a certain area.
The DA periodically analyses its registry for grouping and for every ungrouped GM, it sends a YGM (Your Group Member) message (with GM's IP) to a GL in the same location. In case of multiple GLs, it selects one based upon available context information. The GL then responds with an acknowledgement and completes grouping process by sending a YGL (Your Group Leader) message to GMs. This shifts the status maintenance burden to the GL, which reports the DA about each unresponsive GM. A GL can inform the DA about a missing GM using NRP (Not reported) message and its depleting battery using a GLD (Group Leader Done) message.

Hybrid architecture: Basically, TRENDY has a centralised architecture that converges to a distributed one when the DA uses context information to group GMs as shown in Fig. 1.

Service Descriptions: There are diverse requirements for service descriptions posed by the IoT. TRENDY defines a default compact format for service description consisting of only semi-colon separated URLs of resources offered by a device. This simplistic format of resource description is a compact and efficient choice for constrained environments. However, TRENDY considers the requirement of extra semantic information, and recommends the IETF Core Link Format[3] while allowing any other format depending on the application.

Service Management: The DA maintains all service records with soft states, which need to be updated regularly by SAs. TRENDY introduces an adap-

[3] https://datatracker.ietf.org/doc/draft-ietf-core-link-format/

Table 1: A UA specifies *trendy/server* URL with *GET* method and some URL queries for a service discovery request to get the appropriate service.

URL queries for appropriate discovery	**Matching criteria**
`?location=INB01`	Location based
`?location=INB01&type=temperature`	Location and type based
`?location=INB01&type=temperature&info=sensor`	Location, type and relevant information based

tive timer that considerably reduces the number of update messages. Timer adapts to the demand of a service to increase or decrease the status update interval.

Service Discovery: The DA determines the matching service from the registry using the attributes of the UA request described by the URL queries (examples shown in Table 1). Subsequently it responds back to the UA by appending the service information (resource's URL and IP address of the host) of one or more matching services in the payload.

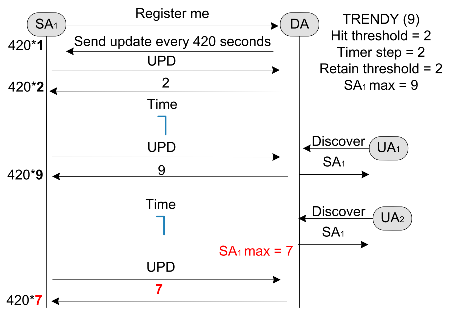
Service Selection: A UA can specify `best` in a service discovery request's payload, to seek the DA's assistance in selecting the best matching host if multiple prospective hosts are found. In this case, the DA determines the most appropriate service (if multiple services have been discovered) using available user and network context information, e.g. battery, hops count, UA location, etc.

Service Invocation: Service discovery is completed when an application gets the response with a service identifier and address of its host. TRENDY, however, enables service invocation using a RESTful web service interface and takes a step ahead by defining an adaptive publishing protocol (Section 3.4).

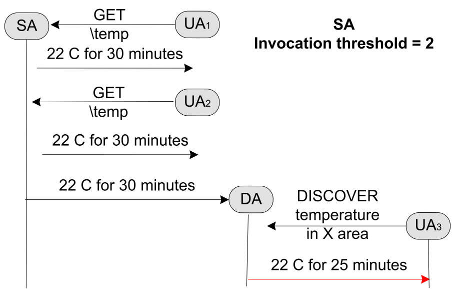
3.3 TRENDY Timer

The DA maintains soft state for each service description, so host devices send status updates within a time period given by the DA. TRENDY uses an adaptive timer to vary this time window length by maintaining a TRENDY counter for individual nodes. The algorithm senses the demand of services to adaptively increase the status maintenance interval for individual nodes. This significantly decreases the number of packets required for status maintenance by the nodes.

The DA is configured with global attributes including, hit count threshold value, timer step, retain threshold and the maximum TRENDY counter value, which can be changed dynamically. It also maintains individual maximum counter values for all registered SAs, which are changed in response to high demand of services hosted by a node. Figure 3a describes the adaptive timer in a scenario. Whenever the DA receives a status update message from a new node, it acknowledges the registration with a time window for the next status maintenance update. The subsequent update messages from the SA are responded with a



(a) Adaptive timer: The DA keeps on increasing the status update interval for a SA until its hosted services become popular.



(b) APPUB: The SA pushed the cache to the DA after two service invocations, which is used by the DA to serve a UA.

Fig. 3: Scenarios demonstrating adaptability of TRENDY

TRENDY counter value which is incremented every time a new update message is received. The SA multiplies the acknowledged TRENDY counter with the basic time window period to determine when the next update message is expected by the DA. The DA keeps on increasing the TRENDY counter up to the maximum value for the SA.

This maximum counter value for the SA is decremented by timer step, when hit count (number of times discovered and selected) of its hosted services surpasses the hit count threshold value during the passage of a time window. Figure 3a shows how the maximum counter value of a SA decreased to 7 from 9 after two discoveries. If the hit count of a SA remained below the retain threshold value, then the counter is increased by the timer step. In case of grouping, all GMs follow the GL's TRENDY counter value.

3.4 Adaptive Piggybacked Publishing (APPUB)

TRENDY devises a demand-based caching technique the APPUB (Adaptive Piggybacked Publishing) as an alternative to balance the trade-off between service invocation delay and network efficiency. It adapts to the demand of a resource for caching rather than blindly maintaining cache of all resources in the network.

The DA maintains cached values with cached time and cache lifetime for each service. The SA implements APPUB algorithm by sending cached values and corresponding lifetime values to the DA, when the number of invocations exceeds the hit count threshold. This enables a SA to get the help from the DA to share the burden by acting as a proxy in busy times. The DA does not pass the node's IP address to a UA, if the fresh (not expired) cached value of the resource is available. Figure 3b shows how the cache is pushed by a SA and then used by the DA to serve a UA.

4 Experiments and Results

Our simulations use the CONTIKI with RPL as a routing protocol and employed COOJA [13] to simulate all SAs (GMs and GLs). ContikiMAC [5] is used as the Radio Duty Cycling (RDC) scheme and Carrier Sense Multiple Access (CSMA) as MAC protocol. Two implementations of CoAP are used in experiments. Erbium [8] is a Contiki based CoAP implementation, which is used inside the 6LoWPAN for SAs; and JAVA based Californium[4] is used to implement the DA and UA. All simulations consist of 36 Tmote Sky nodes where one node acts as a border router to connect the COOJA-based 6LoWPAN to the DA running as Linux process via the Serial Line Internet Protocol (SLIP). All nodes are placed randomly in a $190m \times 180m$ wide field. Each node hosts three resources: temperature, humidity and light. All SAs also share their location and current state of battery with the DA. The nodes are given 5 different location tags with 7 nodes (for grouping: 1 GL and 6 GMs) for each location. The implemented GL nodes in grouping scenarios are only capable of maintaining the status of their GMs. Contiki's ENERGEST [6] module is used to measure the energy consumed at each node. All simulations are executed for 20 DA time windows each of 7 minutes long. Each experiment was repeated 10 times using a different random seed for each iteration. All UA queries are stateless (each is sent as if from a new UA) and randomly selected to send a *GET* request for a resource value in one of the five locations after a random interval between 0 and 10 seconds. The number of queries are varied (100 and 1000) for following Scenarios:

Case 1 - *Basic TRENDY Service discovery (SD)*: This scenario only enables the basic functionality of TRENDY. The UA gets an appropriately selected resource's URL and IP address of a SA hosting the matching service.

Case 2 - *Basic TRENDY SD with adaptive timer*: This scenario has a TRENDY timer with global maximum counter fixed at 9 and hit count threshold at 2 on top of case 1's functionality.

Case 3 - *Basic TRENDY SD with adaptive timer and grouping*: In this scenario, grouping technique is employed with the functionality of case 2.

Case 4 - *TRENDY APPUB and timer*: This scenario employs TRENDY's APPUB technique with the threshold for service invocations fixed at 2 on top of case 2. Therefore, SAs send the cached value of a resource to the DA after two service invocations.

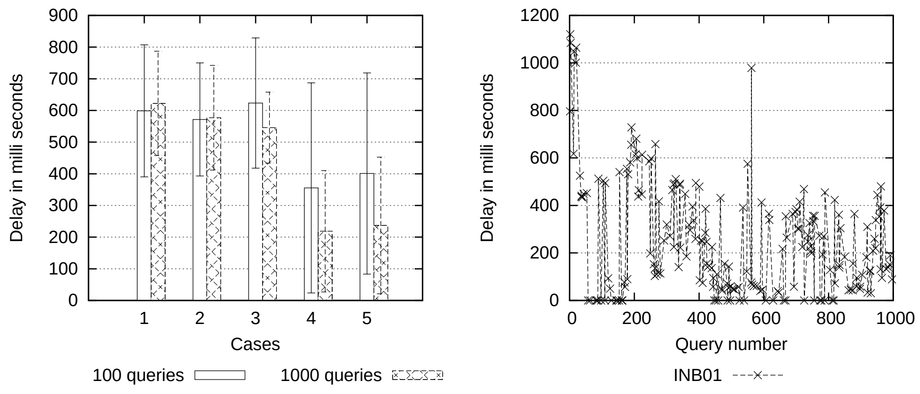
Case 5 - *TRENDY APPUB with timer and grouping*: In this scenario, grouping is also enabled on top of the case 4.

The service invocation delay, number of control packets, number of packets at the DA and energy consumption are measured in all experiments.

4.1 Measurements

Service Invocation delay: Service Invocation (SI) delay is defined as the time interval between issuing an invocation request and the reception of a response. The network traffic load, mean path length and message processing

[4] http://people.inf.ethz.ch/mkovatsc/californium.php



(a) Average Service invocation delay is less in cases 4 and 5 which have employed TRENDY's APPUB.

(b) TRENDY APPUB's effect in case 5: The service invocation delay falls after few service invocations.

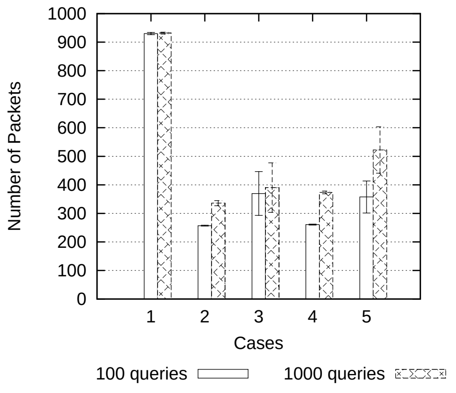
Fig. 4: Service Invocation delay for queries (small is better)

time are the factors which affect the SI delay. Figure 4a shows the advantage of using TRENDY's APPUB technique in cases 4 and 5, which have the lowest average service invocation delay. SAs push the cache value to the DA after two service invocations, which is used to serve the next UA queries resulting in cache hits. Figure 4b depicts this trend for case 5 in one of the five locations.

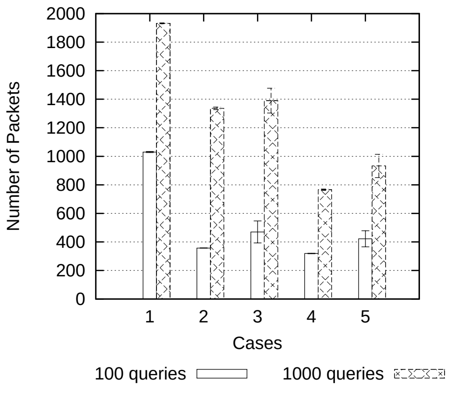
Control Overhead: A protocol's control overhead is measured as the number of control packets used by the protocol to complete its operational and management tasks. Figure 5a shows the control packet overhead as the sum of all registrations, grouping and reporting messages. The TRENDY adaptive timer used in cases 2 to 5 has reduced control overhead. Subsequently, in Figure 5b we additionally display the number of service invocation messages directly served by nodes in the control overhead equation. This figure illustrates the benefit of using TRENDY's APPUB and adaptive timer in cases 4 and 5.

Scalability factor: Packets received at the DA: With TRENDY, all services are stored and maintained by the DA. This requires messages for registration and status maintenance to be sent to the DA from all the SAs. Consequently, the number of messages generated by nodes can overwhelm the network, as most of the messages need to pass through multiple hops to reach the destination. Thus, we consider the number of packets received by the DA from 6LoWPAN network as an important scalability factor of a service discovery solution. Figures 6a and 6b show that case 3 and 5 using grouping mechanism reduced the flood of messages towards the DA by localising status maintenance.

Energy consumption and Network lifetime: To estimate network lifetime, we have considered top nodes in terms of energy consumption from each of

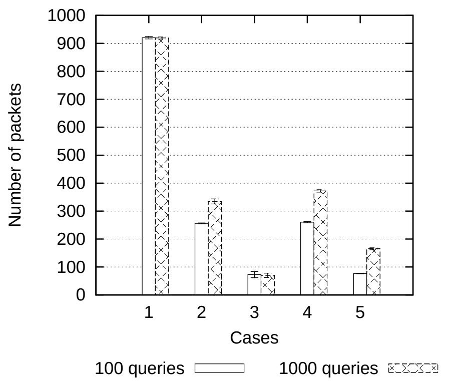


(a) Control packet overhead is far less in cases 2-5 using adaptive timer.

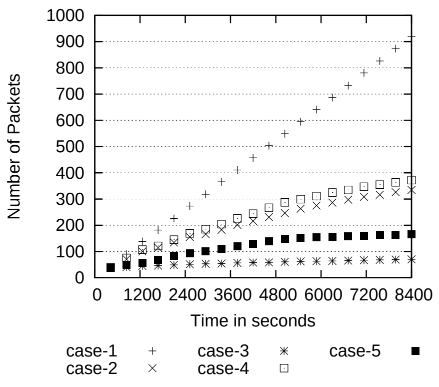


(b) Cases 4-5 with APPUB performed best when service invocations are included.

Fig. 5: Control packet overhead of 35 nodes after 8400 seconds (small is better)

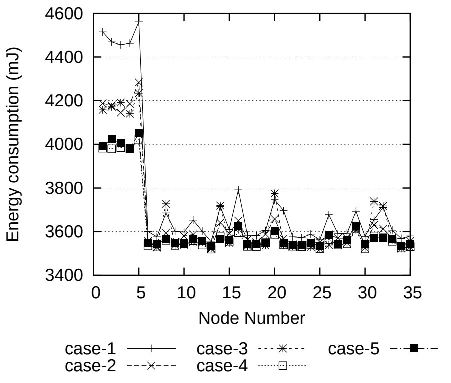


(a) After 8400 seconds: Cases 3 and 5 with grouping have performed best because of localised status maintenance.

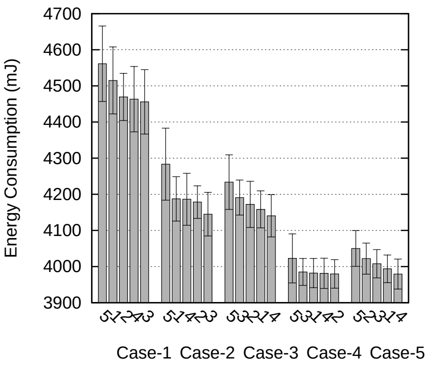


(b) Over the time for 1000 queries: Case 5 has sent more packets to the DA compare to case 3 because of APPUB messages.

Fig. 6: Total Packets received at the DA (small is better)



(a) All 35 nodes: case 4-5 with TRENDY techniques are the best.



(b) Top nodes in each location: Network will last longer in Cases 4-5.

Fig. 7: Energy Consumption after 8400 seconds of simulation and 1000 queries

the five locations. Figure 7a shows the individual energy consumption of all 35 nodes, whereas only top nodes are considered in Fig 7b. Both figures show that cases 4 and 5 using TRENDY's APPUB and adaptive timer will increase the energy efficiency and network lifetime.

5 Conclusion and Future work

This paper presents TRENDY: an adaptive and context-aware Service Discovery Protocol for the IoT. This protocol employs CoAP based RESTful web services, which enable application-layer integration of constrained domains and the Internet. TRENDY's resource directory provides service discovery with a context-aware service selection using user- and network-based context. The trade-off between status maintenance load and reliability is managed by TRENDY's adaptive timer based on demand. TRENDY's APPUB technique has the following benefits: it allows the service hosts to share their load with the resource directory and also decreases the service invocation delay. Furthermore, TRENDY introduces a context-based grouping technique where the resource directory divides the network at the application layer, by creating location-based groups. This grouping of nodes localizes the control overhead and provides the base for service composition, localized aggregation and processing of data. Our simulation results show that TRENDY's techniques decrease the control overhead, energy consumption and service invocation delay. Additionally, the grouping technique considerably decreases the number of packets towards the sink and thus improves scalability in a multi-hop network. In future work, we intend to experiment with service composition by employing more appropriate group leaders for the groups. In addition, the experiments with multiple heterogeneous networks and physical hardware testbeds [11] are also in the pipeline.

References

1. O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer, et al. Internet of Things Strategic Research Roadmap. *Internet of Things-Global Technological and Societal Trends*, pages 9–52, 2011.
2. Fen Zhu, Matt W Mutka, and Lionel M Ni. Service Discovery in Pervasive Computing Environments. *Pervasive Computing, IEEE*, 4(4):81–90, 2005.
3. T. A. Butt, I. Phillips, L. Guan, and G. Oikonomou. TRENDY: An Adaptive and Context-Aware Service Discovery Protocol for 6LoWPANs. In *Proc. Third International Workshop on the Web of Things*, page 2. ACM, 2012.
4. S. A. Chaudhry, W. Do Jung, C. S. Hussain, A. H. Akbar, and K.-H. Kim. A Proxy-Enabled Service Discovery Architecture to Find Proximity-Based Services in 6LoWPAN. In *Embedded and Ubiquitous Computing*, pages 956–965. Springer, 2006.
5. A. Dunkels. The ContikiMAC Radio Duty Cycling Protocol. *Swedish Institute of Computer Science*, 2011.
6. A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based Sensor Node Energy Estimation. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 409–410. ACM, 2007.
7. D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio. Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. *Services Computing, IEEE Trans. on*, 3(3):223–235, 2010.
8. R. Klauck and M. Kirsche. Bonjour Contiki: A Case Study of a DNS-Based Discovery Service for the Internet of Things. In *Ad-hoc, Mobile, and Wireless Networks*, pages 316–329. Springer, 2012.
9. R. Klauck and M. Kirsche. Enhanced DNS Message Compression - Optimizing mDNS/DNS-SD for the Use in 6LoWPANs. In *Proc. 9th International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS 2013)*, Mar. 2013.
10. S. Mayer and D. Guinard. An Extensible Discovery Service for Smart Things. In *Proc. Second International Workshop on Web of Things*, page 7. ACM, 2011.
11. G. Oikonomou and I. Phillips. Experiences from Porting the Contiki Operating System to a Popular Hardware Platform. In *Proc. 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, June 2011.
12. G. Oikonomou, I. Phillips, L. Guan, and A. Grigg. ADDER: Probabilistic, Application Layer Service Discovery for MANETs and Hybrid Wired-Wireless Networks. In *Proc. 9th Annual Communication Networks and Services Research Conference (CNSR 2011)*, pages 33–40, Ottawa, Canada, May 2011.
13. F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648. IEEE, 2006.
14. Z. Shelby. Embedded Web Services. *Wireless Communications, IEEE*, 17(6):52–57, 2010.
15. Z. Shelby, S. Krco, and C. Bormann. CoRE Resource Directory. *draft-shelby-core-resource-directory-05*, IETF, 2013.