TRENDY: An Adaptive and Context-Aware Service Discovery Protocol for 6LoWPANs

Talal Ashraf Butt, Iain Phillips, Lin Guan, George Oikonomou
Department of Computer Science, Loughborough University, UK
{T.A.Butt,I.W.Phillips,L.Guan,G.Oikonomou}@lboro.ac.uk

ABSTRACT

We propose, TRENDY, a new registry-based Service Discovery protocol with context awareness. It uses CoAP-based RESTful web services to provide a standard interoperable interface which can be easily translated from HTTP. In addition, TRENDY introduces an adaptive timer and grouping mechanism to minimise control overhead and energy consumption. TRENDY's grouping is based on location tags to localise status maintenance traffic and to compose and offer new group based services. Our simulation results show that TRENDY techniques reduce the control traffic considerably and also reduce the energy consumption, while offering the optimal service selection.

Categories and Subject Descriptors

D.2.8 [**Discovery and look-up for things and their services on the Web**]: Service discovery—*Service selection, Web services, CoAP, 6LoWPAN*

1. INTRODUCTION

The IoT (Internet of Things) vision has drastically changed the way we foresee the future Internet. The low cost, data rates and power consumption based devices are becoming smarter with the advancement of computing technology. IETF's 6LoWPAN standard has made it possible for even networks of resource-constrained devices to connect directly to the Internet. Such IP based connectivity also gives the opportunity for the reusability of existing infrastructure and proven standards, without any translation gateways. These developments have changed the profile of WSNs from isolated and application-specific to be more interconnected and directly accessible 6LoWPANs. However, these networks still have constraints, including limited packet size, intermittent communication, high packet loss, and restricted power and throughput. Within these challenges, resource sharing can be achieved by abstracting resources as services. A service can provide any data reading or measurement, e.g. temperature reading of a sensor, etc., which can be shared inside the 6LoWPAN or with an external IP network. Furthermore, services can be composed to create higher-level services, which can be combined to create mashups and provide new functionality [12]. The discovery of the available services is crucial to avail this potential, which can be achieved by employing suitable service discovery and selection protocols.

The use of the existing IP-based service discovery solutions is an obvious choice to be considered. However, large packet sizes and heavy control overheads of these solutions make it infeasible to operate them directly on a 6LoWPAN, due to the necessary fragmentation. Similarly other available distributed solutions are also not viable, due to their excessive demand for broadcast.

The WoT (Web of Things) vision depicts a view where a collection of web services can be discovered, composed and executed [12]. Although web services paradigm can remarkably change the way we access the services, they are a poor match with constrained networks [11]. Both SOAP (Simple Object Access Protocol) based big web services, and REST (Representational state transfer) are based on heavy protocols, which make them impractical for such networks. However, constrained nodes can benefit from IETF's CoAP [11] (Constrained Application Protocol) standard, which offers compact and optimized RESTful web services.

In this paper, we have presented TRENDY, a registry based service discovery protocol based on CoAP based web services. TRENDY uses intelligent adaptive timers to vary the reporting time period, which reduces the control traffic significantly. Furthermore, TRENDY's architecture is established on location-based grouping of the nodes, which diminishes the energy consumption and also enables the localised composition of services. In addition, TRENDY offers optimal service selection based on the context and reliability of the server and popularity (demand) of the service.

2. RELATED WORK

There are several service discovery protocols. These can be classified into three broad categories on an architectural basis: *centralized*, *distributed* and *hierarchical*. Centralized architectures have a central directory, where Service Agents (SA) register their services. Subsequently, User Agents (UA) discover the services by sending unicast queries to directory. On the other hand, distributed architectures demand every node to collaborate using broadcast or multicast to discover a service. Hierarchical architectures employ some nodes with high capabilities, to represent a cluster of nodes in their vicinities.

The industry-standard, IP-based Service Discovery Pro-

WoT 2012, June 2012; Newcastle, UK

tocols (SDP) including SLP, UPnP, JINI and Salutation are not directly applicable to 6LoWPANs. NanoSLP [3], a miniaturised version of SLP, is an attempt to use SLP with reduced features and only supports a directory-less architecture. In [2], the author employs SSLP [4] inside the 6LoWPAN and provided the 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. NanoSD [5], a distributed approach uses multicast and broadcast extensively and requires each node to keep the service information of its neighbors. In [1], a DNS based approach is introduced, which introduces compression. However, it makes assumptions regarding the availability of the required information and database entries. In [10], TCP/IP based web portal's mechanism is used with the help of regional locals (master nodes) for the local service discovery and data servers are used in the network, which provide a complex mechanism that increases cost burden. Another recent approach [8] provides RESTful web services using HTTP based service discovery using existing or injected strategies, but it does not address the management and status maintenance of the registered services. Some Internet drafts of the IETF CoRE work group have offered few solutions for service discovery. The mDNS based solutions are mostly relied on the availability of IP multicast. However, Resource directory [1] uses CoAP as underlying communication protocol, but it does not address the bottleneck problem caused by the service update messages. In addition, other requirements, e.g. optimal service selection are not considered.

The IoT vision has significantly changed the way we approach the solution for a 6LoWPAN. There is a need of a SDP for 6LoWPAN which should consider:

- **Scalability:** The solution which can manage and control discovery overhead in a dense network, while keeping track of the availability of service hosts.
- **Compact size:** The complexity of the solution should be implementable on the resource-constrained devices.
- **Compact communication:** The limited bandwidth (20kbps to 250 kbps) and packet size (60-80 bytes) at the application layer of 6LoWPAN, demands a careful use of bandwidth. Large packets will get fragmented and result in multiple packets, which can overwhelm the entire network.
- **Sleeping nodes:** Constrained nodes usually save their limited power sources by sleeping most of the time, using a Radio Duty Cycling (RDC). The solution should assign high priority to this issue, by giving some alternatives like proxy which can act on behalf of sleeping nodes.
- **Heterogeneity:** Normally, heterogeneous devices are used in a 6LoWPAN (e.g. home automation); the solution should accommodate the devices regardless of their hardware.
- **Interoperability:** Different standards for lower layers are being used in the LowPANs. The solution should be inter-operable to work between networks with diverse standards. Interoperability aids the application development, as single application can be used across different networks.
- **Energy efficiency:** This is the key to increasing the lifetime of the network. The solution should optimise its operation by minimising its control overhead. Furthermore, most of the nodes in multi-hop networks waste their energy by passing the message towards the sink.
- **Fewer requirements:** The assumptions of multiple resource-rich devices should be ignored. However, the protocol should be opportunistic enough to exploit the availability of high-capacity devices in the network.
- **Service selection:** During the process of service discovery, redundant nodes can be found in a network offering the same service. A solution must be able to select the optimal service in specific context (location) depending on factors, including past service demand, reliability and remaining energy.
- **Service composition:** Many services are discovered and invoked together, mostly depending on their vicinity. For example, switching lights on and off in a room will require all the actuators of lights to be called collectively. A service discovery protocol should consider this aspect to allow grouping of different services in a locality, to offer new services.
- **Adaptivity:** A SDP should be adaptive enough to change different parameters: status maintenance interval, etc., with the changing requirements, to decrease the bandwidth in use.

3. TRENDY AIMS AND ARCHITECTURE

TRENDY maintains a registry on a DA (Directory Agent), where SAs (Service Agents) register services. UAs (User Agents) query the DA, to find the location of a service. Other features of TRENDY are:

- An adaptive timer mechanism, which reduces status maintenance overhead considerably.
- An architecture based on lightweight, location-based grouping to reduce the effect of high traffic towards the single DA, without the prerequisite of resource-rich capability nodes. It ensures that most of the traffic generated in one location does not affect the whole multi-hop network. Furthermore, grouping mechanism enables the network to offer new group based services (e.g. switching lights).
- A service selection algorithm, which searches for the optimal service in terms of location and remaining battery life of a SA.
- CoAP (Constrained Application Protocol) is used as an application-layer communication protocol, which enables the interoperable RESTful web service paradigm in 6LoWPAN. Therefore, it enables a more efficient and easy programming paradigm for potential future applications.
- The DA can optionally act as a proxy providing caching and decreasing service invocation time.
- It operates independent of the specific routing protocol. This allows for future development of routing without affecting TRENDY. We do not yet investigate the potential of combing clustering at the routing and service-discovery levels.

3.1 Architecture

The architecture of TRENDY categorises SAs into GLs

[1]http://tools.ietf.org/html/draft-shelby-core-resource-directory-02

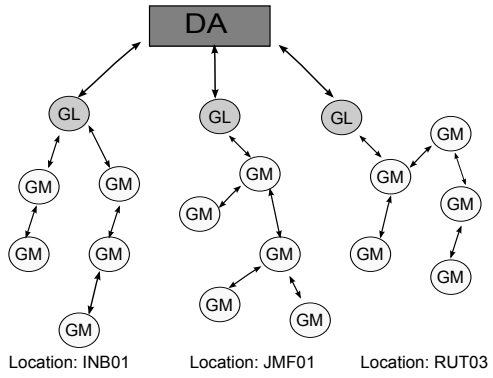

Figure 1: Architecture of TRENDY

(Group Leaders) and GMs (Group Members). We argue that a directory based architecture is inevitable for a 6LoWPAN, where nodes mostly sleep, and broadcasting is expensive to discover a service in case of a distributed solution. Figure 1 presents TRENDY's architecture. In a 6LoWPAN, an edge router works as a bridge between the WSN and IP networks by adaptation layer. In most of the 6LoWPANs, edge router is resource-rich compared to all other devices in the network. TRENDY aims to exploit the underlying architecture of 6LoWPAN by interweaving the service discovery solution with the existing infrastructure. This is achieved by embedding the DA role in an edge router.

TRENDY's architecture is distinct from clustering as a GL does not provide the functionality of a distributed directory, nor it is elected by the surrounding nodes. The DA uses the capability information of the nodes to assign them different roles. TRENDY provides the scalability by creating groups to deal with the increasing size of the PAN. The benefits of using our grouping technique are multifaceted; it localizes the status maintenance and empowers DAs to execute location-based commands. For example, the DA can ask a GL in a specific location to switch off the room lights or reduce the level of heating in the area. Resultingly, a more evolvable programming paradigm is enabled to facilitate the future WoT applications.

3.2 Entities and Roles

Figure 2 shows the relationship between different entities of TRENDY. Details of the functionality of these entities and their operation together is covered in this section.

3.2.1 Group Member (GM):

All SAs are required to implement a CoAP resource for the role of Group Member (GM), to send an UPD (Update) message periodically to DA. These messages will keep DA informed, about the availability of a GM. Later, during the formation of groups a GM receives a YGL (Your Group Leader) message, it then starts sending the UPD messages to its GL.

3.2.2 Group Leader (GL):

TRENDY requires some SAs to offer the functionality of Group Leader (GL) by implementing an additional CoAP resource called Group Leader to perform its role. On top of GM's capability, a GL is required to understand messages for grouping including SGL (Selected Group Leader), YGM

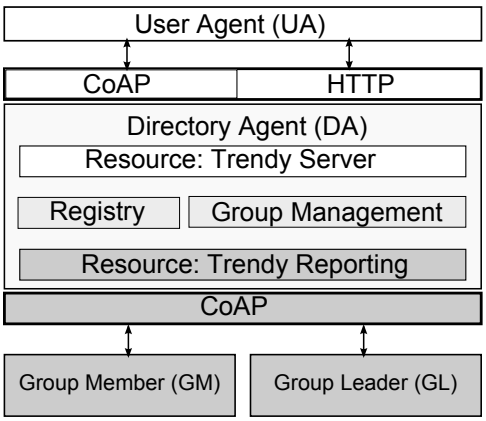

Figure 2: Relationship between TRENDY's Entities

(Your Group Member), YGL, NRP (Not Reported) and GLD (Group Leader Done). Every selected GL keeps the record of assigned GMs by keeping a list of their IP addresses and status. There can be multiple GLs in a single location, if one GL is not able to maintain the nodes in that location.

3.2.3 Directory Agent (DA):

The DA has a backbone role in TRENDY's architecture. We have used the RPL (IPv6 Routing Protocol for Low power and Lossy Networks) routing protocol in experiments, and the DA sits at the root. This ensures that the DA can communicate with all other nodes in the network. However, any other routing protocol can be used for this purpose. The responsibilities of a DA include:

- Maintenance of a Registry table to store all the service information in the network.
- Provision of adaptive timer to decrease the overall control overhead.
- Construction and re-construction of groups, by selecting of Group Leaders and negotiating with them.
- Provision of different discovery options, including selection of optimal service.
- Optionally it can act as a proxy, in that situation other benefits like caching can be exploited.

3.2.4 User Agent (UA):

The UA (User Agent) is a client interested in discovering services available in the network. A UA sends queries to the DA. The protocol used for communication between a UA, and DA can be COAP (default) or HTTP (if DA acting as a proxy), depending on the solution. A UA can be either part of the sensor network or send a request from elsewhere in the Internet.

4. TRENDY PROTOCOL DETAILS

This section covers the working of TRENDY's protocol by explaining it's different stages.

4.1 Registration

In TRENDY, it is mandatory for every SA (GM or GL) to register with the DA. At first, the SA sends a UPD message with its location, battery information and service information. The service information is formatted using comma separated URLs of available resources on a SA. Therefore, it

is recommended for developers to use self explanatory URLs for resources. However, every SA is required to implement a `well-known` resource to enable the detailed resource discovery using core-link format[2]. The DA responds with an acknowledgement carrying a period of time window. After successful registration, every SA is bound to inform the DA about its status at least once in a time window, using UPD message. Upon reception of every subsequent UPD message, the DA marks the respective SA active in the registry. The DA checks its registry after every time window, and deletes the records of inactive SAs. Moreover, an adaptive timer is employed at the DA, which keeps a counter for every node and increments it when a UPD is received from that node. This counter value is piggybacked in the acknowledgement by the DA. GM multiplies its reporting time with the received counter value. Consequently, the number of UPD message decrease considerably over the time.

The IP address of the DA can be either hard coded in a SA, or distributed as DHCP parameter if this protocol is in use. The assumption taken here is that all SAs will inform the DA with location information that can be used to form groups. The location information either be static in the SA or can be assigned using web service interface. This information depends on the locality of a device, e.g. in a building monitoring and automation scenario: the SAs deployed in the same room or specific corridor can be assigned similar location tags.

4.2 Grouping

In order to guarantee the availability of the discovered service, the DA needs to maintain the status of the nodes. However, as the number of nodes increases in the network, the load of status maintenance increases. In a multi-hop network, other nodes get involved in the operation of passing the message forward until it reaches at DA. TRENDY proposes a grouping overlay, to keep the load of status maintenance in the specific area of the 6LoWPAN and to deal with group-based requests.

Grouping mechanism eradicates the bottleneck problem associated with the centralized directory, by reducing the traffic for status maintenance at the DA. Location tags of the nodes are exploited by this mechanism to form groups of nodes. The DA then adaptively recognises nodes in the close vicinity and groups them, with a respective GL. Group Leaders keep track of their remaining battery and after reaching a threshold value can inform the edge router to select a new GL.

The architecture of TRENDY in figure 2 shows the relationship between the different agents. The process of making a group is:

1. The DA periodically analyses the registry for grouping.
2. The DA selects all GLs by using their service information, to send SGL messages to them. The confirmation from GL is received in an acknowledgement, with a piggybacked value specifying the maximum number of GM it can support. If multiple GLs are available for the same location, the DA will select the optimal one based on the rank (explained in section 5.2).
3. For every new GM, the DA sends a YGM message with the GM's IP address to respective GL.

[2]http://tools.ietf.org/html/draft-ietf-core-link-format-11

Discovery queries	Description
`?location=INB01`	Location based
`?location=INB01&type=temp`	Location and type based

Table 1: Trendy's discovery queries

4. The GL confirms the addition of new GM and sends back the confirmation to the DA, with the GM's IP address (for validation).
5. The GL periodically checks the newly added GMs, and synchronises them by sending YGL message with its trendy counter value to them. This needs an acknowledgement from GM to confirm the call.
6. The registered GM then starts sending UPD message to GL instead of DA, and uses the received counter value to reset their reporting interval.
7. The GL periodically checks the status of GMs after an appropriate interval, and reports the inactive ones to DA by sending NRP message with GM's IP address to DA.

A GL acts as a mediator to watch availability status of GMs in its group. Furthermore, it ensures that only the most closer nodes in the vicinity are involved in passing the message, thus it reduces the energy overhead for rest of the network. In addition, the overlay creates an opportunity to make GLs more responsible, by including other functions, e.g. aggregation, group actuation (turning all lights in a room). However, the current specification of TRENDY does not support these features, but future work will include more advanced GLs. A GL can inform the DA anytime about it's depleting battery, when it comes to a certain threshold, by using GLD message. It continues the GL responsibility until informed by DA with a new GL address.

4.3 Discovery

Services are maintained at the DA to enable easy discovery. A UA sends a SREQ message to DA, using CoAP (default) or HTTP protocol. A UA specifies the `trendy/server` URL with CoAP's GET method and URL queries (examples shown in Table 1) for service discovery. Furthermore, the UA can optionally append `best` in the payload to get the optimal service. The DA discovers the most appropriate service using context, by reading the query variables. In case of an optimal service selection request by the UA, the DA selects the best service (if multiple services have been discovered). Subsequently, the DA responds back to the UA by appending the service information (resource's URL and IP address of the host) of optimum service or list of services in the payload. If a DA is offering the HTTP protocol interface, then it acts as a proxy and responds on behalf of a SA.

5. TRENDY TECHNIQUES

This section covers the detail of TRENDY's proposed techniques.

5.1 Adaptive Reporting Timer

We propose an adaptive timer for reporting. At the start, nodes send UPD messages by selecting a random value between 0 and n (selected based on number of nodes), to reduce the probability of collisions, as every node sends UPD message at different intervals. The algorithm significantly decreases the number of packets required for status mainte-

nance by the nodes.

The DA maintains a trendy counter value for every registered node, to enable the TRENDY timer. The value of the trendy counter starts with 1 and it is incremented till `counter_maximum`. Depending on the environment and application needs, the maximum trendy counter value can be chosen to guarantee optimal system status update. However, this value can be changed dynamically. In future work, we will enable DA to keep separate maximum value for each group, which will be increased or decreased depending on the demand of a certain area. The first UPD from a node is considered as registration by the DA, and responded with a basic time window value. Subsequent UPDs are responded with trendy counter in the payload. Whenever DA receives an UPD message from a new node it initialises the `trendy_counter` at node's registration time. After registration, the DA follows algorithm 1 for every subsequent UPD message. This algorithm updates the `trendy_counter`, which subsequently enables a SA to increase the time period for its next UPD message. The `trendy_current` is used to keep

Algorithm 1 On receiving an UPD message

```
if trendy_counter_not_changed&&trendy_current <= 1
then
  if trendy_counter < counter_maximum then
    trendy_counter++
  end if
end if
trendy_current ← trendy_counter
```

the current counter's value, which gets decremented after every time window. On the other hand, the `trendy_counter` holds the standard counter value for next increment. This algorithm only increases the counter value, when it reaches its minimum and is not already incremented in the time window.

Every SA gets the basic DA report time window in the payload for the first UPD, and a `trendy_counter` value in later responses. Whenever an acknowledgement is received from DA, SA follows the algorithm 2. This algorithm randomises a point between 50-90% of time window and then multiplies it with the `trendy_counter`. Each SA starts with the default `trendy_counter` value of 1.

Algorithm 2 SA report time interval calculation

```
trendy_counter ← received_trendy_counter
report_time ← 0.5 × time_period
interval ← 0.4 × time_period
report_time += random between 0 and interval
if trendy_counter = 1 then
  default_report_time ← report_time
end if
report_time ← report_time * trendy_counter
```

The selection of a random value ensures fewer numbers of collisions. Every SA follows the same algorithm while reporting to GL or DA.

5.2 Optimal GL Selection

TRENDY devises an optimal GL selection mechanism to select among multiple potential GLs in a location. Whenever DA finds any GM with no GL allocated, it selects a GL based on the location. If more than one GLs are available, it goes through the list of GLs and compares their ranks. These ranks of GLs are maintained by DA and updated every time the registry is analyzed for grouping. The equation to calculate rank is:

$$rank = sT + nGM - f - \frac{b}{1000}$$

Where sT is serving time of a GL in hours, nGM is the number of GM supported by GL, f is the number of failures in response to YGM messages, b is battery consumed, which is divided by 1000 to reduce its effect on the equation.

5.3 Optimal Service Selection

In TRENDY, the DA maintains service information, location, battery consumed, and registration time for all registered nodes. Furthermore, it keeps hit count for every service, which is incremented whenever the service is discovered and selected to be sent to the UA. The DA offers location-based discovery by exploiting the available information. In addition, if a UA has also specified the `best` in the request message, the DA will select the optimum one from the searched list. In this case, the DA, first of all, discovers services by using query parameters and uses the following algorithm:

1. Select the best service based on hit count.
2. If the decision is not made, then select the one whose server has less battery consumed.
3. If still deciding, then select one with earliest registration time

6. EXPERIMENTS AND RESULTS

The main objective of TRENDY is to provide optimal service selection and architecture to offer group based services, while minimising the control overhead and reducing energy consumption. This section covers the details of the experiments undertaken for performance analysis and discusses the gathered results.

6.1 Simulation Setup

We use the CONTIKI operating system with RPL as a routing protocol and COOJA [13] to simulate all SAs. The DA is implemented in JAVA and runs on a Linux machine communicating with COOJA. We use two different implementations of CoAP. Erbium [6]: a CONTIKI based CoAP implementation, which is used inside the 6LoWPAN for SAs; and Californium[3]: used to implement the DA. All our simulations consist of 26 Tmote Sky nodes where one node acts as a border router, which are placed randomly in a $30m \times 20m$ wide field. We use ContikiMAC as Radio Duty Cycling (RDC) and Carrier Sense Multiple Access (CSMA) as MAC protocol. CONTIKI's ENERGEST module is used to measure the energy consumed at each node. All SAs register with the DA. Each node starts with a reporting interval randomly selected between 100 and 300 seconds to prevent collisions. In experiments, the DA's basic time window is fixed to 600 seconds and maximum `trendy_counter` value to 9. Each simulation is executed for 10 time windows and repeated 10 times to get the average results. The performance of TRENDY is analysed by comparing following three scenarios: 1. BASIC: All nodes report their status to DA using a constant interval. 2. TRENDYTIMER: TRENDY adaptive timer is used by all nodes to report status. 3. TRENDYFULL: Both TRENDY

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

Method	Control Packets (MEAN)	(SDEV)	Energy (J) (MEAN)	(SDEV)
BASIC	810	16.13	62.08	5.33
TRENDYTIMER	163	28.18	61.27	7.53
TRENDYFULL	297	10.62	60.13	4.85

Figure 3: Control Overhead and Energy Consumed

grouping mechanism and adaptive timer are employed. This scenario maintains three GLs and 22 GMs. In all scenarios, nodes implement battery, location and light CoAP-based services. GLs in the third scenario only implement resources needed to act as a group leader.

6.2 Control Overhead

The control overhead of *TRENDYTIMER* scenario surpasses the performance of other cases. The *TRENDYFULL* scenario constitutes more control overhead, which is understandable as traffic increases with the extra grouping communication. However, in both cases number of packets for control overhead remained far lower than the *BASIC* scenario, as shown in figure 3.

6.3 Energy Consumption

The results of our experiments depict that *BASIC* scenario with the constant reporting interval has consumed more energy than TRENDY based scenarios. The benefits of proposed grouping mechanism are supported by the results. The implementation with *TRENDYFULL* has performed slightly better compared with other solutions, as shown in figure 3. The figure shows the total energy consumption of the network (all 25 nodes excluding border router) in joules (J). These results are over 10 runs of the experiment.

7. CONCLUSIONS AND FUTURE WORK

The advent of new standards has enabled constrained networks to become an active part of the Internet. The next step is to integrate these networks with the web to fully conceive WoT paradigm. Service discovery and selection play a significant role in this regard, as it is a costly process due to communication overheads. Moreover, node sleep cycles pose concerns regarding the accuracy of a directory-less service discovery solution. This paper proposes TRENDY, which is an adaptive and context-aware service discovery protocol. It uses an adaptive timer to decrease the bandwidth utilisation by reducing the control overhead. Moreover, TRENDY's grouping mechanism decentralises the status maintenance traffic burden and provides an application level grouping, which can be exploited to compose new group based services (e.g. switching all lights in a room). In addition, it has a compact implementation size (experiments done with Tmote Sky) and employs CoAP as a communication protocol which has an optimised header size. Furthermore, CoAP enables the RESTful web service paradigm and thus has inherent interoperability with other solutions. TRENDY also provides the optimal service selection mechanism based on service popularity, remaining battery and host uptime.

The future plan is to enable a DA to act as a proxy and to offer new group based services by service composition. Additionally, new GL roles will be designed and implemented, where a GL can forward a request to whole group and can aggregate the results of its group. Furthermore, TRENDY is currently simulated with 26 nodes, so more focus will be given to enhance it to work for large-scale network. The performance will be analyzed by simulations and deployment to physical hardware testbeds [9]. Lastly, we plan to adopt trickle-style [7] algorithms for TRENDY's adaptive timers and investigate the potential of performance and efficiency gains.

8. REFERENCES

[1] F. Anwar, M. Raza, S. Yoo, and K. Kim. Enum based service discovery architecture for 6lowpan. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6. IEEE, 2010.

[2] S. Chaudhry, W. Jung, C. Hussain, A. Akbar, and K. Kim. A proxy-enabled service discovery architecture to find proximity-based services in 6lowpan. *Embedded and Ubiquitous Computing*, pages 956–965, 2006.

[3] C. Jardak, E. Meshkova, J. Riihijarvi, K. Rerkrai, and P. Mahonen. Implementation and performance evaluation of nanoip protocols: Simplified versions of tcp, udp, http and slp for wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC 2008) IEEE*, pages 2474–2479. IEEE, 2008.

[4] K. Kim, S. Yoo, H. Lee, S. Park, and J. Lee. Simple service location protocol (sslp) for 6lowpan. *drraft-daniel-6lowpan-sslp-00*, 7, 2005.

[5] A. Kovacevic, J. Ansari, and P. Mahonen. Nanosd: A flexible service discovery protocol for dynamic and heterogeneous wireless sensor networks. *Mobile Ad-hoc and Sensor Networks, International Conference on*, 0:14–19, 2010.

[6] M. Kovatsch, S. Duquennoy, and A. Dunkels. A low-power coap for contiki. *Mobile Ad-Hoc and Sensor Systems, IEEE International Conference on*, 0:855–860, 2011.

[7] 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*, pages 2–2. USENIX Association, 2004.

[8] S. Mayer and D. Guinard. An extensible discovery service for smart things. In *Proceedings of the Second International Workshop on Web of Things*, page 7. ACM, 2011.

[9] 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)*, Barcelona, Spain, June 2011. IEEE.

[10] M. Raza, S. Yoo, K. Kim, S. Joo, and W. Jeong. Design and implementation of an architectural framework for web portals in a ubiquitous pervasive environment. *Sensors*, 9(7):5201–5223, 2009.

[11] Z. Shelby. Embedded web services. *Wireless Communications, IEEE*, 17(6):52–57, 2010.

[12] D. Zeng, S. Guo, and Z. Cheng. The web of things: A survey. *Journal of Communications*, 6(6):424–438, 2011.

[13] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level simulation in cooja. In *European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, Delft, The Netherlands, JAN 2007.