Service Knowledge Discovery in Smart Machine Networks

Durmus, Yunus; Onur, Ertan

DOI
10.1007/s11277-015-2483-2

Publication date
2015

Document Version
Final published version

Published in
Wireless Personal Communications

Wireless Pers Commun (2015) 81:1455–1480
DOI 10.1007/s11277-015-2483-2

Service Knowledge Discovery in Smart Machine Networks

Yunus Durmus · Ertan Onur

Published online: 8 March 2015

Abstract The devices surrounding us become smarter and can autonomously form a network without requiring our intervention. However, our needs can be even better accommodated when the networked devices cooperate and complement each other's capabilities. One of the initial steps towards achieving a cooperative platform of smart devices is the discovery of resources and capabilities within the network. Today's operational service discovery protocols carry simple text-based uniform resource identifiers that are not expressive enough. Machines cannot comprehend the meaning of a new service that is not in their knowledge base. In addition to being more expressive, service discovery protocols must compensate the diversity to improve cooperation between the devices that use different application protocols and operate on different communication interfaces. In this paper, we propose the Smart Discovery Protocol (SDP) which outperforms the operational service discovery protocols with three main features: (1) more expressive semantic representation of the services, (2) operating in the network layer to deal with diversity, and (3) unifying existing service discovery protocols. SDP represents services with ontologies as some recently proposed semantic service discovery protocols. It further enhances the success of semantic representations by creating a unified platform that can carry legacy discovery services. In this respect, the novelties of SDP are as follows: firstly, it operates in the network layer and consequently abstracts both the application layer and communication interfaces. Secondly, SDP unifies the legacy service discovery protocols by integrating their simple text-based service representations in one message. The underlying transport mechanism of SDP is designed as an add-on to the Neighbor Discovery Protocol (NDP) of the IPv6 standard. The metadata is carried in the payload of ICMPv6 packets. Simple text-based representations of other service discovery protocols are embedded in type-length-value options of NDP. Authenticity of the devices is ensured by the IPv6 Secure Neighbor

Y. Durmus (✉)
Embedded Software Group, TUDELFT, Delft, The Netherlands
e-mail: ydurmus@ieee.org

E. Onur
Department of Computer Engineering, Middle East Technical University, Ankara, Turkey
e-mail: eronur@metu.edu.tr

Discovery protocol. Unlike previous semantic approaches on service discovery, we have implemented our protocol on real hardware. The results demonstrate the feasibility of carrying semantic representations of the services and integration of other service discovery protocols.

Keywords Service discovery · Machine to machine · Ontology · IPv6 · Neighbor Discovery · Semantics · ICMPv6

1 Introduction

The computational power of networked devices enhances every day and these devices can accomplish very complex tasks. For instance, a smartphone can observe our daily travel patterns and suggest improvements. However, these smart devices are not *swiss army knives*: they cannot encompass all kinds of capabilities and they have limitations that cannot be surmounted by simply boosting their processing power. Take a portable smartphone as an example, it cannot have a 50 in. display or wash our clothes. Often smart devices have to utilize the resources and capabilities of other devices and complement each other for accommodating the needs of their owners. The cooperation must be accomplished without human intervention because the configuration requirements may be unmanageable as the number of devices increases rapidly. According to Cisco, there will be over 10 billion mobile-connected devices including machine-to-machine (M2M) modules in 2016, exceeding the world's population [4]. Such estimates signal the need for an autonomous network of smart devices (and machines) where cooperation exists without humans in the loop.

One of the significant challenges in establishing a network of smart devices that complement each other is the lack of an expressive and autonomous service knowledge discovery system. In the scenario illustrated in Fig. 1, a device searches for a piece of

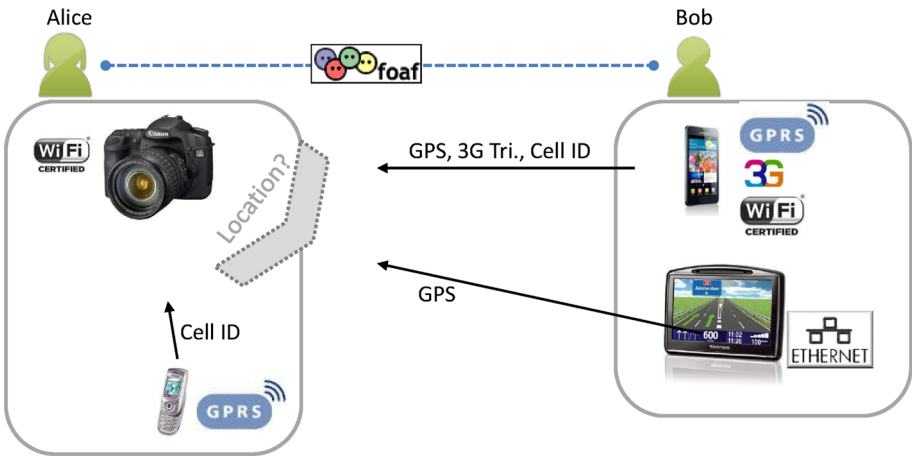

Fig. 1 Missing location information is queried from neighbor nodes. The device owners and network interfaces are also shown

missing context information. Alice's camera does not have any capabilities for discovering its geographic position. To determine its location, it queries the devices in the vicinity as to whether or not they are capable of providing the location information. Instead of asking for specific services that give GPS, 3G triangulation and cell id information, simply any service that can extract location information is queried. Nearby devices that know they have hardware for location information advertise their specifications with the properties of the equipment such as the precision of the location information. After identifying the devices that have the required capability, the camera requests the location information from one of the capable devices according to its required level of precision. This scenario clearly depicts that devices have to be smart and infer on collected pieces of information. We refer to this process of making machines understand the semantics of service descriptions as service knowledge discovery.

Semantic Web researchers achieve a similar goal of making machines capable of understanding the content in a web page without human intervention. Machines collect the information scattered in several web pages through the logical relationships among them. The power of machine-understandable web pages originates from the representation of knowledge in ontologies. In an autonomous network of smart devices, we need a service knowledge discovery protocol that can evolve and adapt to future requirements just as in the Semantic Web. The ontologies comprising the information about the services should be distributed among the devices. With ontologies, the discovery system builds a knowledge base that evolves and comprises new systems and services. Fortunately, the tools for handling the ontologies are available, and the scientific challenge for us is to propose a distribution system for the information. Some of the recent proposals for service discovery (e.g., mRDP [15]) also promote the use of ontologies as knowledge representation. However, they are not able to deal with heterogeneity of both the networks and applications. Compatibility with legacy systems in these architectures is also missing.

There are many operational and even recently proposed service discovery protocols that are being used today in many systems. However, there is still no widely deployed autonomous service discovery protocol that exists in every device. The reasons are as follows: firstly, although some of them improve expressiveness by using ontologies, still many of them prefers simple representations that do not evolve for future requirements of the services. Secondly, there are various application protocols for services and existing service discovery protocols add new application protocols to the list instead of decreasing the heterogeneity. While deploying a service, a separate application layer protocol for discovery has to be installed. Thirdly, companies push vendor locking. Each service discovery protocol has its own island of connected devices where there is no interaction with an outsider.

In this paper, we propose the Smart Discovery Protocol (SDP) as an add-on to the Neighbor Discovery Protocol (NDP, IETF RFC 4861) of IPv6, which is positioned in the kernel and comes as pre-installed in the operating systems. SDP is independent of the application layer and can exist in heterogeneous networks by the convergence to IP. It operates on ICMPv6 packets that carry semi-structured (ontological) service representations and queries. Not only the knowledge about the resources and capabilities that form a service, but also the details about the service owner and the context are included in the representations. In the type-length-value (TLV) options of NDP, the ICMPv6 packets carry the Uniform Resource Identifiers (URI) that define legacy services. Semantic data is carried in the payload of these packets. Multicast advertisement and solicitation messages are employed to maintain scalability. Secure Neighbor Discovery (SEND, RFC-3971) is

used for authenticating the collaborating devices. For confidentiality, multicast group security proposals can be employed. The contributions of SDP are the following:

- SDP is a semantic service knowledge discovery protocol which decreases human intervention in service discovery for devices complementing each other and enables cooperation among the machines.
- SDP can operate in all IPv6-based networks independent of the lower and upper layers.
    - Devices that operate on the network layer like routers can collect more information about the structure of the network by looking at ICMPv6 packets and improve the QoS.
- Legacy systems are unified in one message type.
- Low-power devices are also involved in discovery by using URI identifiers of existing service discovery architectures.

In the next section, we propose and describe SDP. First the challenges and requirements of a service discovery protocol are given, then the design of SDP is matched with the requirements. The performance of SDP is presented in Sect. 4. Existing service discovery architectures are explained briefly and discussed in Sect. 5. Then, we conclude and discuss future work.

2 Smart Discovery Requirements

In this section, we present the Smart Discovery Protocol (SDP) that paves the way for an autonomous network of smart devices. We present the challenges of service knowledge discovery in smart machine networks using the scenario illustrated in Fig. 1.

In the light of these challenges, we will explain service representation format and ontologies for readers who are not familiar to semantic technologies in Sect. 2.2. Then the SDP protocol will be described starting with Sect. 3. We will elaborate the packet format, message types, legacy support features, protocol operation, and security.

2.1 Challenges and Requirements of Knowledge Discovery

The challenges of service discovery have been identified in [8] and they are still valid with an increasing importance. We summarize and elaborate some of those challenges that are still open research questions and introduce new challenges (last two) to indicate the need for a new knowledge discovery protocol.

Nodes in a network need to employ a service knowledge discovery system to obtain extensive information about the other collaborating devices and establish their own knowledge base. The combination of resources and capabilities can be referred to as services as defined in the Information Technology Infrastructure Library.[1] Resources are the hardware components and the capability is the software running on the hardware that makes it usable. The knowledge discovery involves not only the resources and the capabilities of devices, but also the context and the user-specific information including the social network profiles of the users. When a new service appears in a network, devices should be able to infer its functionalities and start using it autonomously. The information and inferred knowledge of the services should also involve the context of the physical and

[1] http://www.itil-officialsite.com.

social environments that can be a distinctive factor in service selection. The discovery protocol should abstract the heterogeneity both in terms of networking and semantics. While satisfying all these requirements, the protocol should accommodate the legacy (operational) service discovery systems.

2.1.1 Knowledge Representation

Service identification is a serious problem that cannot be addressed by just standardization of the list of services in the market. Firstly, there is a diverse set of services. It is rather difficult to keep an up-to-date list of the services and distribute it. Even if such a standardization is accomplished by the vendors, each vendor reflects its own categorization. Secondly, users cannot remember all the keywords for an exact naming of the services or user may want to query a service with different categorizations such as purpose. In the scenario presented in Fig. 1, camera asks for "location" and the peers reply if they have any capabilities for providing location information, such as GPS position, cell id or 3G triangulation. Instead of the exact name match, the purpose of the resources are matched. Lastly, services can alter their inputs, outputs or procedures, knowledge representation should be capable of evolving to adapt such future demands of the services.

2.1.2 Personalization for Authorization

Services do not live in a closed world environment where they only interact with trusted parties. There are untrusted peers that access management should eliminate by requiring a trust relationship between the services. Since resources like smartphones have one-to-one relation with their owner, trust between the resources is indeed the trust relation between their owners. In the scenario presented in Fig. 1, camera and cell phone belongs to Alice while smart phone and navigation device belongs to Bob. The trust between the devices indeed the trust between the owners of the devices. When camera asks for a service which can provide location information, others check the owner of the camera. If a trust relation does not exist between the owners in the social network then service query is not replied. Authorization can be easily addressed by using trust relationships of the owners' themselves, incorporating the personal information in the service representation.

2.1.3 Context Dependency

Context information determines the value of the service. Depending on the context, service may become useless or crucial. In the scenario presented in Fig. 1 for instance, if all the devices were in an indoor environment, the GPS information would be useless since GPS satellite signals do not penetrate indoors. Most reliable information in that case would be the 3G triangulation. Therefore, navigation device and smart phone should not offer their GPS services or the node which uses the GPS service should infer that the information is not reliable. If the service representation involves a piece of information about the constraints of the GPS, machines take better decisions by combining the information with the context.

Ontologies are more expressive data representations than URI like description of the services that are being used in operational service discovery protocols. The above issues can be addressed by the ontologies. In Sect. 2.2, the ontologies will be explained in detail.

2.1.4 Network Intrinsic Approach

The smart machine networks are full of heterogeneity of the devices in terms of hardware and software. Differently from [8], today we can claim that network diversity is being unified in IP based networks. Several researchers call the IP layer as the narrow waist of the hourglass model which is a bridge between the lower and the upper layer. Even if devices operate in different networks or use different application layer protocols, IP layer is the common interface. Therefore, a discovery protocol in the network layer can be supported by many devices. Machines do not have to implement separate application layer protocols which are not used by the service itself. In the scenario, GPS, 3G triangulation services can be provided by different application layer protocols. Devices like camera may use wifi while phones may use the cellular network. To overcome the heterogeneity in network architectures and to avoid gateways, the discovery protocol should operate at the IP layer.

2.1.5 Unified Service Discovery

Currently many systems deploy various service discovery protocols which depend on simple text based matching of the service identifiers. These protocols are not compatible with each other. As a consequence, there are disjoint islands in which only compatible devices exist. And the islands are closed to foreign standards. To enhance the adoption of new service discovery protocols, they should be designed to be compatible with the legacy discovery protocols. In Sect. 5, existing service discovery protocols are reviewed. Most of the protocols transport URI like identifiers and have distributed architectures that do not require a central broker. Unification can be achieved by carrying the URI like identifiers of each standard along with the semantic descriptions.

2.2 Knowledge Representation

In this section, we represent knowledge representation for readers who are not familiar to Semantic Web. Knowledgeable reader may skip this section and proceed to Sect. 3 for the details of SDP.

Semantic Web was introduced in 2001 [2] aiming at making the web machine-understandable. At present, humans are the only contributors to the Web, we create the web pages and understand the content. Machines are just dealing with the distribution of the content. However, when they become aware of the content, they may adapt to human behavior and needs.

It is crucial to create a vocabulary which machines can comprehend. As a first attempt, the Resource Description Format (RDF) was proposed to represent the resources in triples: *subject-predicate-object*. To be able to define domain specific vocabularies, RDF Schema (RDFS) was designed and it became possible to describe classes, sub-classes and properties of RDF resources. However, still there was a requirement for defining complex relationships between the objects modeled with RDFS. Therefore, the Web Ontology Language (OWL) was created. As it is seen the abbreviation OWL is not consistent with Web Ontology Language. The reason is that OWL sounds better and it honors the *One World Language* artificial intelligence project at MIT around mid-70s.

With OWL it is possible to construct new classes by simple set operations such as union and intersection. Existential quantifiers *for all* ($\forall$), *there exists* ($\exists$) and even cardinality constraints (such as max or min) become available to describe inter dependence of the classes. After the success of OWL1, OWL2 was accepted in 2009 as a W3C standard

which promoted researchers to implement tools for manipulating it. To query data sets, SPARQL query language is used. SPARQL is similar to SQL that is used to fetch data from an RDF data content. Therefore, it is a perfect fit for resource and capability solicitation. Since OWL and SPARQL are widely accepted by the community, we also favor the use of them as the metadata format. However, SDP does not restrict itself in one

```
@prefix foaf:           <http://xmlns.com/foaf/0.1/> .
@prefix ns1:            <http://en.wikipedia.org/wiki/> .
@prefix ns2:            <http://live.dbpedia.org/resource/> .
@prefix ns3:            <http://live.dbpedia.org/property/> .
@prefix owl:            <http://www.w3.org/2002/07/owl#> .
@prefix ns6:            <http://live.dbpedia.org/ontology/> .
@prefix rdfs:           <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbpedia:        <http://dbpedia.org/resource/> .
@prefix ns13:           <http://dbpedia.org/datatype/> .
@prefix rdf:        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
ns1:Samsung_Galaxy_S_II        foaf:primaryTopic
        ns2:Samsung_Galaxy_S_II .
ns2:Samsung_Galaxy_S_II        rdf:type        owl:Thing .
ns2:Samsung_Galaxy_S_II        rdf:type        ns6:Device .
ns2:Samsung_Galaxy_S_II        rdfs:label "Samsung Galaxy S II"@en.
ns2:Samsung_Galaxy_S_II        ns3:imagesize        "200"^^xsd:int;
        ns3:predecessor        ns2:Samsung_Galaxy_S ;
        ns3:successor        ns2:Samsung_Galaxy_S_III ;
        ns6:weight        130 ,130.41 ;
        ns3:display        "AMOLED with 480\u00D7800 pixels"@en ,
                "800"^^xsd:int ;
        ns3:cpu        "2"^^xsd:int ;
        ns3:memory        "1"^^xsd:int ;
        ns3:input        "Multi-touch screen, headset controls,
                proximity and ambient light sensors,
                3-axis gyroscope, magnetometer, accelerometer,
                aGPS, and stereo FM-radio"@en ;
        ns3:storage        "16"^^xsd:int .
ns2:Samsung_Galaxy_S_II ns3:connectivity "210.0"^^ns13:second ;
        ns3:gpu        "Adreno 220"@en ,
                "PowerVR SGX540"@en ,
                "ARM Mali-400 MP"@en ;
        ns3:battery        "120000.0"^^ns13:second ;
        ns3:memoryCard        "microSD"@en ;
        ns3:networks        "802"^^xsd:int ,
                "Dual band CDMA2000/EV-DO Rev."@en ,
                "HSPA+: 21/42 Mbit/s;
                  HSUPA: 5.76 Mbit/s LTE 700/1700 Rogers Only"@en ,
                "WiMAX 2.5 to 2.7 GHz;"@en ,
                "UMTS: 850, 900, 1700 , 1900, and 2100 MHz"@en ;
        ns3:soc        "Samsung Exynos 4 Dual 45nm"@en ;
        ns3:rearCamera        "8"^^xsd:int ;
        ns3:frontCamera        "2"^^xsd:int .

<http://example.com/foaf#me> a foaf:Person ;
    foaf:mbox_sha1sum "50a842005e63853ab00d2d46dab152d2e16e92e3" .
```

Fig. 2 Sample OWL instance document gathered partially from DBpedia. It advertises the device as a smart phone, gives details about the properties of the device with the owner information

format, it can carry any semi-structured data format. It is an evolutionary system which adapts to unforeseen future requirements and services.

2.2.1 OWL and SPARQL

Semantic Web makes the web machine-understandable. Similarly, SDP in smart machine networks makes the resources and capabilities machine-understandable. Advertisement and solicitation messages employ the Semantic Web standards. For instance, the device definition is presented with OWL ontology language in the advertisement messages and solicitation (query) messages are composed of queries expressed in SPARQL.

An example smart advertisement message from a smart phone is depicted in Fig. 2. The specification is an OWL instance document serialized with Notation 3 [1]. The device details are taken from DBpedia but shortened to fit in one column. The owner identifier (digest of the email address) is also given in Friend of a Friend (FOAF) social network language.

In Fig. 1, the location capability is requested from the neighboring devices. A solicitation message that contains SPARQL query like the one in Fig. 3 can be sent to discover a device with GPS service. In the query, GPS service that belongs to a specific person is requested by appending the digest of the email address. Since the GPS can be in different formats, a regular expression is used to increase the possibility of a match.

First three challenges mentioned in Sect. 2.1, knowledge representation, personalization and context dependency can be addressed with the use of the ontologies which are described above. With ontologies instead of exact name match, services can be queried with different categorizations. Ontologies can adapt to changes in the definition of the services since they are more expressive than the URIs. Semantic definitions of the services do not require an acceptance from a standards association which can slow down the adoption of a new service or an update in a service. FOAF ontology clearly shows the ability of personalizing the services. The owner information can be embedded into the service definition. Context and hardware information can also be used in ontologies.

```
PREFIX prop: <http://live.dbpedia.org/property/>
PREFIX ont:  <http://live.dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

select ?phone ?who ?os ?weight
  where
  {
        ?phone    prop:input ?gps;
                     FILTER regex(?gps, ".*a?GPS.*")
                  prop:battery ?battery ;
                  ont:operatingSystem ?os ;
                  ont:weight ?weight .
        ?who      a   foaf:Person ;
                  foaf:mbox_sha1sum
                     "50a842005e63853ab00d2d46dab152d2e16e92e3".
  }
```

Fig. 3 Sample service query message written in SPARQL vocabulary. GPS property is filtered with a regular expression

3 The Service Knowledge Distribution Protocol

In Sect. 2.1.4, we mentioned that SDP is implemented in the network (IP) layer to abstract the heterogeneity. Furthermore, SDP has to work in ad hoc fashion in opportunistic networks.

There are different candidates for a messaging protocol namely HTTPU, SOAP and DNS. In Sect. 5, protocols that use these messaging protocols will be summarized. Key features of the protocols related to service discovery are compared in Table 1. All the protocols have multicast support without reliability. Security covers only authentication and integrity. Though SSL is mentioned for some protocols, in fact it is not available with UDP and multicasting. ICMPv6 is a better choice in terms of complexity and network intrinsic feature that is required for heterogeneity. Moreover, serialization is flexible with both structured and payload fields. Structured parts (TLV options) are used for security and representing the legacy services, whereas payload is used to carry semi-structured data.

SDP has two types of packets. The Smart Advertisement (SA) and the Smart Solicitation (SS) messages are similar to the Neighbor Advertisement (NA) and the Neighbor Solicitation (NS) messages of the ICMPv6 Neighbor Discovery Protocol (RFC-4861). The NA and NS messages are used to discover the MAC-IP address association of the neighboring devices in a network. SA and SS messages are also ICMPv6 messages which differ from NA and NS in terms of the functionality. Since SA and SS messages operate at the network layer, they are independent from the application layer protocols. They work on any data link layer protocol.

3.1 Packet Format

The packet format is presented in Fig. 4, *type*, *code* and *checksum* are the common ICMPv6 fields. *Total* field stores the number of packets that semi-structured data is divided into and *sequence* is used to re-assemble the payload. *Security options* are defined in Secure Neighbor Discovery (RFC 3971), used for authentication and integrity. The *Backward compatibility* part includes the URI representations of services expressed as in the existing architectures. Lastly, *payload* field involves the ontologies, semantic definition

Table 1 Comparison of different carrier protocols for semi-structured data

	HTTPU & HTTP	SOAP	DNS	ICMPv6
Transport layer	UDP& TCP	UDP	UDP	–
Reliability	With TCP	–	–	–
Serialization	Any text	XML	Structured without payload	Structured with payload
Header overhead and parsing complexity	Medium	High	Medium	Low
Operating layer	Application layer	Transport layer	Application layer	Network layer
Multicast	With UDP	✓	✓	✓
Security (authentication and integrity)	With SSL	SSL or WS-Sec	DNSSEC	SEND

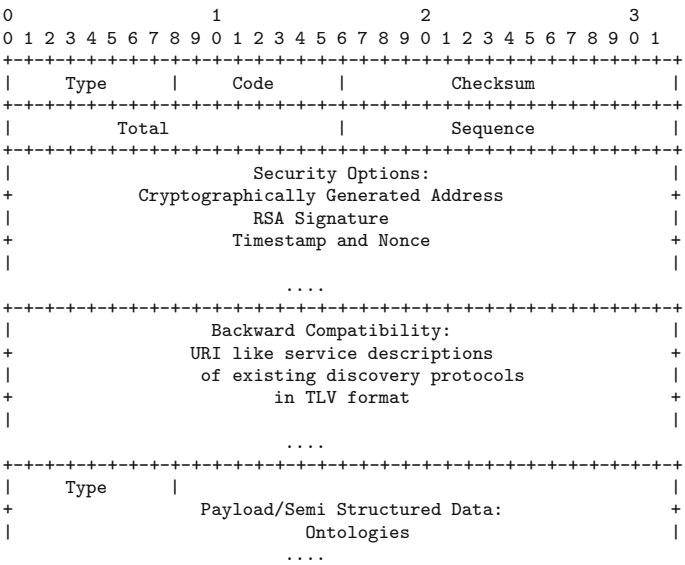

Fig. 4 Packet format for the Smart Advertisement and the Smart Solicitation messages

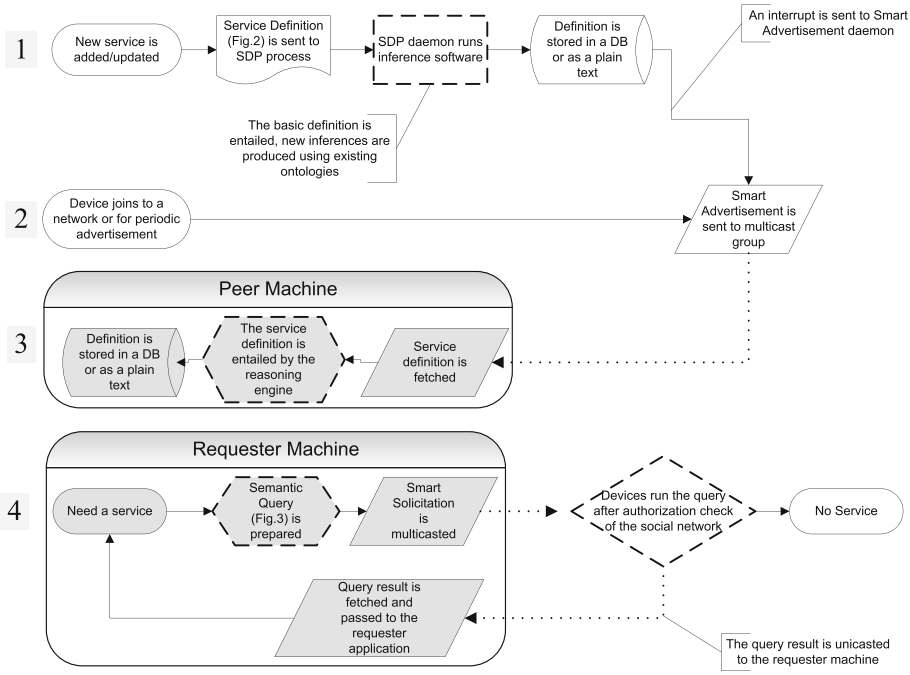

Fig. 5 The process of the Smart Discovery Protocol

of the services and the device. While parsing the packet to distinguish the payload field from the TLV options, payload starts with a predefined type value but does not have length and value fields.

3.2 Protocol Operation

The Smart Discovery Protocol operates in ad hoc mode and no central entity is required to distribute the service definitions. Devices send the SA message to declare their existence in the network. The SS message on the other hand aim to query a required service inside the network. Both the SA and SS are multicast messages however, the SA message which is sent as a reply to the SS, is a unicast message.

All the device details, service descriptions and owner profile are placed in a data store, it can be a database or the file system. When a service is added or updated, the data store is also refreshed. A reasoning engine like FaCT++ [14] runs and infers new properties about the services. For instance, a device has GPS hardware and GPS is classified as a location supplier, then the device is considered as it can supply location information. Later on when the device gets a query, it does not have to run the reasoning engine again, therefore it can be concluded that reasoning is an offline procedure. Some devices may lack support or do not have enough computing power to run reasoning software. In such cases the reasoning step can be eliminated. A detailed service description may eliminate the need of inference.

In Fig. 5, we present the protocol operation. The dashed boxes represent the semantic operations like inferences which do not exist in operational service discovery protocols. An SA message declares the existence of a device and its services in the network. The semantic data describing the resources, capabilities and even the owner details as presented in Fig. 2 encapsulated in SA are sent when the node joins the network, in case of a change in the information or periodically (presented as the cases 1 and 2 in Fig. 5). The message is published to a pre-determined multicast group address. Peer devices can overhear the SA messages inside the network and create a local database of services for future use (case 3 in Fig. 5). In need of a service, the local database may be queried first. However, local cache does not guarantee the existence of the service.

To query a functionality, a multicast SS message is published to the group, shown as the fourth case in Fig. 5. The payload involves a semantic query in SPARQL vocabulary (such as Fig. 3). When the devices get the SS message, they run the query in their local semantic data store. Then they reply with unicast SA messages that give the details of the service. The size of the reply message depends on the query, if the details of just one service is required one packet may be enough. It is best practice to prepare a query whose result fits into just one packet.

3.3 Unified Service Discovery for Legacy and Low Power Devices

In Sect. 2.1.5, the diversity of the service discovery protocols is mentioned. Devices that use the same service discovery protocol implicitly establish clusters that are disjoint from the devices which adopt another standard. Therefore new service discovery protocols should embrace the legacy standards.

Converting messages of different service discovery protocols to each other is not an easy task. Protocols have diverse set of functionalities which may not have a counterpart in the corresponding protocol. INDISS [3] is an interoperability system for service discovery protocols. In the prototype of the INDISS system, the messages in Simple Service Discovery Protocol (SSDP) and Service Location Protocol (SLP) are converted to each other.

The messages are first converted to a intermediary scheme where the messages semantically matched to each other. It is stated that there are still some functionalities that do not match. For instance, SSDP does not have a central entity, whereas SLP employs a directory agent to store all the service definitions inside the network.

INDISS divides the service discovery events into three: "Registration Events", "Discovery Events" and "Advertisement Events". As seen in Table 2, SDP covers only discovery (solicitation) and advertisement events. There are also registration related events such as *SrvReg, SrvDeReg* in SLP which are not covered in SDP. Therefore, SDP cannot be matched exactly with other protocols. However, matched messages can satisfy a distributed service discovery protocol. For instance in SLP protocol the matched messages are the ones that are used in distributed discovery. Uncovered messages are mainly required for the communication with a central server that is responsible for collecting all the service details and dispatching them from one source.

INDISS like interoperability systems aim to convert the messages to each other without any change in the services. Since semantically matching the standards is not easy and requires excessive effort, in SDP a different approach is taken. As shown in Fig. 6 simple text representations (sometimes in URI format, eg. SLP, DNS-SD) of the services in different discovery protocols are carried along with ontologies in single SDP message. Devices which prefer using the parsing and service matching APIs of legacy protocols instead of ontology precessing, keep consuming these simple representations. Both sides again supports SDP messages in the their operating system. The initiator embeds the URI representation inside the packet with semantic representation. The receiver omits the semantic representation and only fetches the URI representation that it supports. The URI services are matched by using existing APIs. By combining all the protocols in one message, less packets are transmitted in total. Legacy devices are supported with a small software update which only parses the SDP message and dispatches the messages to corresponding protocols. Moreover, INDISS like systems can incorporate SDP in their design to decrease the message traffic in the network.

One question still remains open is how we will embed the different messages into one SDP message which can be parsed easily. Most of the service discovery protocols summarized in Sect. 5 such as SSDP, SLP and DNS-SD, distinguish the services by URI like texts and mainly text based matching is performed. The URIs are mostly short texts, for instance DNS-SD TXT records are intended to be around 200 bytes or less. In order to embed the legacy service messages, type-length-value (TLV) options of the ICMPv6 are used. New TLV options are created for each text based service discovery protocol. The size of these options are determined by the length field which is an 8-bit unsigned integer in units of 8 octets and results in $2^8 \times 64 = 16$ Kb = 2 KB. After parsing the TLV options they are passed to their protocol's daemon.

Another motivation for these TLV options is that they are easier to parse for low power devices. Other semantic oriented service discovery protocols exclude such devices from

Table 2 SDP messages and corresponding counterparts of other protocols

SDP	SLP	SSDP	DNS-SD
Multicast SA	SAAdvert	ssdp:alive	–
Multicast SS	SrvRqst	ssdp:discover	Query
Unicast SA	SrvRply	HTTP/1.1 200 OK	Response

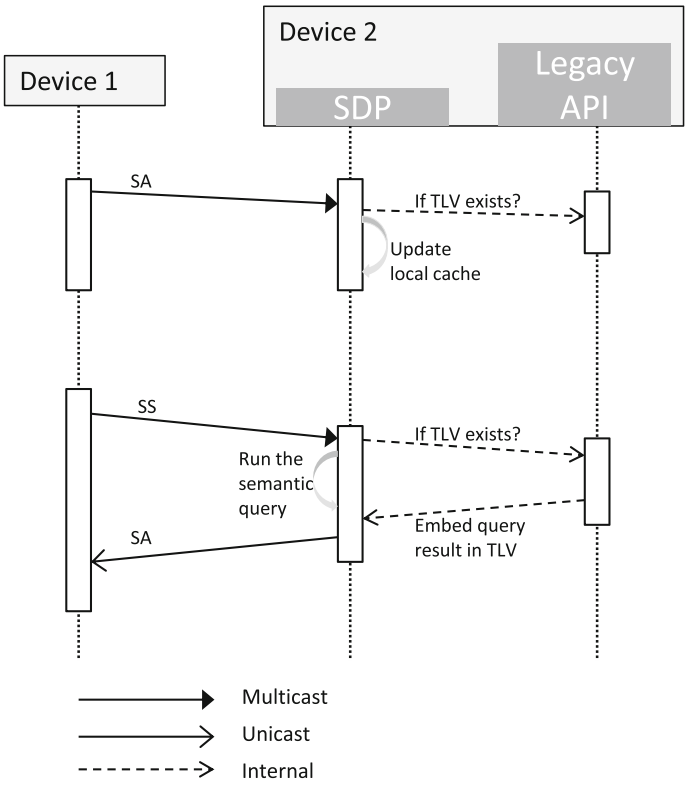

Fig. 6 SDP messages are given in the sequence diagram. If there is a TLV that carries a legacy service identifier, the legacy service API is called

their design. Low power devices like sensor nodes can easily parse TLV options and fetch the URI identifiers of the services expressed in different discovery protocols. In a single message both computationally high and low powerful devices are addressed which enhances the adoption of the protocol, and makes the transition to semantic technologies smoother and easier.

3.4 Reliability

SDP is a distribution protocol for the ontologies and operates at the IP layer where end-to-end reliability is not guaranteed. For scalability multicast messages are preferred in all the discovery protocols. However, reliability in multicast messages requires huge overhead on the transport layer, basically reliability is traded for scalability. There are experimental protocols like the Pragmatic General Multicast reliable transport protocol (RFC 3208), which try to maintain reliability with negative acknowledgement (NACK) packets. NACK packets can reduce the traffic but still the loss of the NACK packet is a problem.

In SDP, like other service discovery protocols, periodic retransmissions are employed to enhance the reception rate but still reception is not guaranteed. Both the SA and the SS messages are retransmitted with increasing intervals in the order of two and after some number of retransmission process ends. The duration of the interval and the number of

retransmissions depend on the medium and background traffic. In a medium with low bit error rate small number of retransmissions are enough.

Another issue is the replies to the SS messages. Since the nodes in the network get the SS message at the same time, their replies may collide. Although the data link layer and MAC protocols avoid collisions, it is still preferred to send the replies after a random back-off time.

3.4.1 Fragmentation and Traffic Shaping

In Fig. 4, it is seen that there are "total" and "sequence" fields which are used to re-assemble the fragmented message. For fragmentation another option is IPv6 fragmentation header. However, if IPv6 fragmentation were used, every retransmission message would be considered as a different message by the stack. When one of the packets of a message drops, IPv6 fragmentation removes the whole message and partial message is not passed to the SDP. By depending on our own fragmentation method, in retransmissions the peer can reassemble a message by gathering packets from different transmissions.

Additionally, SDP employs traffic shaping to decrease the congestion. Especially in wireless networks like 802.11x, multicast messages can easily incur congestion due to limited bandwidth reserved for them. Therefore, SDP puts time delays between the consequent packets to decrease the congestion. The performance increase employing the traffic shaping is presented in Sect. 4.

3.5 Authentication and Integrity

In service discovery, adversaries can inject fake services or alter the definitions of the existing services. Therefore, depending on the requirements of the network, authentication and integrity checks should be devised. As summarized in Table 1, all the carrier protocols aim to maintain the authentication and integrity to establish trust and prevent the denial of service attacks which can easily be done by advertising non-existing services. SDP also ensures the authentication and integrity with Secure Neighbor Discovery (SEND, RFC 3971). Since SDP is an add-on to NDP and SEND is designed to secure NDP, SDP is also covered with SEND.

SEND introduces four new options to neighbor discovery protocol:

- *Cryptographically generated address:* Used to guarantee that address-owner association is valid with the asymmetric key encryption.
- *RSA signature:* The digest of the packet is signed by the private key of the source. The signed digest authenticates the source and guarantee that no other node can alter the packet, any change on the packet data is detected by the peers.
- *Timestamp:* Timestamp option is employed to prevent the replay attacks.
- *Nonce:* In the solicitation-advertisement message pairs, randomly generated nonce values are used for association.

3.6 Confidentiality

Though authentication and integrity is offered by many service discovery protocols, confidentiality is not considered in protocols that employ multicast messages. Any node inside a network can analyze and trace each device with its offered services. In a trusted network such as office and home environment, the risk may be low. However, in an open

network malicious nodes can overhear the discovery packets and easily build an inventory of network.

In open networks, it is advised to depend on a secure overlay network such as virtual private networks (IPSec). With IPSec all the traffic is secured. However, it is a complex protocol which requires detailed pre-configuration and has bootstrapping problems. Service discovery protocols are designed to be simple and have less overhead, therefore security should also be addressed without heavy protocols. All in all, we also omit confidentiality like other protocols and consider service discovery as a case study for the researchers working on multicast group security.

4 Performance Evaluation of SDP

The multicast nature of the all the discovery protocols trades off the reliability of the messages. SDP also operates on unreliable multicast ICMPv6 packets, which incur drops due to congestion and noise in the wireless medium. Compared to other service discovery protocols that operate at higher layers, obviously SDP imposes lesser load in terms of packet headers. However, the metadata payload carried over SDP significantly increases the load. We deal with increased load and its consequence, packet drops by the traffic shaping method defined in Sect. 3.4. The web service based systems like DPWS are expected to carry XML metadata which is close to the size of the metadata of SDP. On the other hand protocols that carry just the URI definitions of the services like DNS-SD and SLP, have lesser load.

Despite of the high load incurred by the SDP on the network with respect to simpler service discovery protocols, the traffic is still less than the load of a web page. For instance, the size of the HTML page in the URI *google.com* is around 11 KB whereas the metadata in Fig. 2 is 1.8 KB.

SDP differs from the other protocols with its design in dealing with heterogeneity and unified discovery. However, still the performance and especially reliability of the protocol should be assessed.

4.1 Experiments on Real Hardware

In order to assess the performance of the protocol and especially to observe the reliability, the protocol is implemented in Ruby programming language which is served as an open-source project.[2] The ICMPv6 structure described in Sect. 2 is created with *type* value assigned to 200/201 which are reserved for private experimentation in the ICMPv6 standard. Different metadata sizes are used in the experiments. For the experiment environment "*Eduroam*", the largest WiFi network of TUDELFT is used. The network represents a crowded office environment and the experiments are carried out between 13:00 and 17:00 while the network is active. The signal levels of the devices that we experimented vary between −80 and −40 dBm; the bit-rates vary between 18 and 54 Mb/s depending on the location. The structure of the experiments is as follows: There are one sender and four receiver laptops. Sender and receivers are being served by different access points. Three of the receivers operate in 802.11a network and the rest is in 802.11g network. The sender laptop either announces its service details or queries for a service and waits for the reply.

[2] github.com/yunus/SDP.

The latency of the announcement and query-response packets are given as a metric for performance.

We start with experiments on the latency of advertisement and solicitation messages with their responses. Then, we compare SDP against SLP and show that SDP does not lead to extra latency while unifying legacy discovery protocols. Lastly, we demonstrate the performance of the traffic shaping.

4.1.1 Results on Message Latencies

First, we experimented the latency of the advertisement messages. The metadata of the services with different sizes are fragmented into SA packets and sent as a batch 100 times to different number of machines. At the receiver side the duration from the first message till the last one is presented in Fig. 7. As the metadata size increases, more packets are transmitted leading to an increase in the duration. Multicasting the traffic provides scalability as the receivers incur similar latencies.

Second, we experimented the round trip delay of solicitation messages and their reply including the semantic data parsing. The solicitation message in Fig. 3 is sent to the peers. The peers run the query and the result is sent back with unicast messages. Both the solicitation and its reply fit into one ICMPv6 packet. In Fig. 8, the time from the start of the solicitation message till the reception of the reply is presented. Even with the semantic data parsing the round trip time is lower than a half second. Unfortunately, there exists variance which is due to the channel conditions.

4.1.2 Comparison of SDP to SLP

One of the novel features of SDP is that it is capable of carrying legacy service messages. The initiator inserts the URI identifier of the legacy protocol in a TLV field of SDP. As shown in Fig. 6, the receiver parses the TLV, uses the APIs of an existing library of the legacy protocol to parse the identifier. If the message is a solicitation, legacy API checks whether the request matches the service or not. If the service is matched in the reply whole service definition is embedded in another TLV field of SDP.

In order to validate our design, we compared our implementation with SLP. The jSLP[3] library is used to send SLP service request and service reply messages. In SDP for parsing and matching the service identifier again the jSLP API is used. In both protocols a service is queried with the identifier "service:test" and the reply contains "service:test:myService://my.host.com". Figure 9 shows the duration in solicitation messages similar to Fig. 8. In this test differently from the previous one, SDP makes an additional API call to the jSLP while also performing a semantic query. When Figs. 8 and 9 are compared, we observe that calling an external library does not lead to an increase in the duration. Moreover, both protocols, SLP and SDP perform similarly. The SDP protocol does not impose higher latencies than SLP, which can lead to timing issues on the legacy services.

4.1.3 Results on the Traffic Shaping Feature

To observe the effect of message size on the packet reception rate, 2000 messages are transmitted and their reception rates are presented. Half of the messages are composed of just one packet and the other half is composed of four packets. As seen in Fig. 10, the

[3] http://jslp.sourceforge.net/.

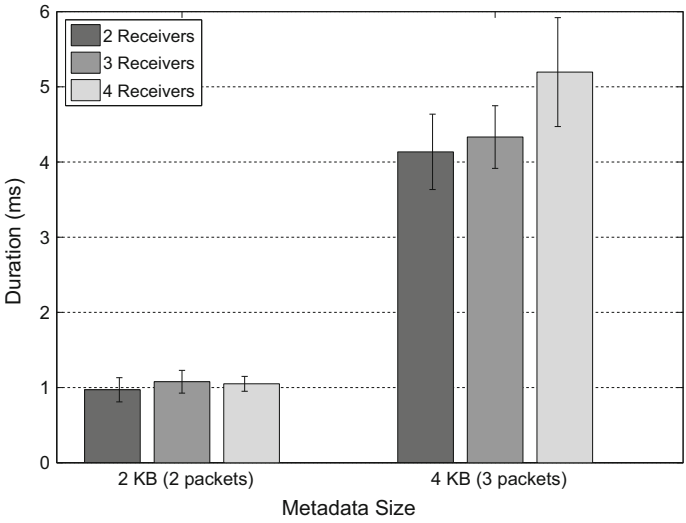

Fig. 7 The durations between the reception of the first message till the last one with different metadata sizes are presented. The MTU is 1500 bytes

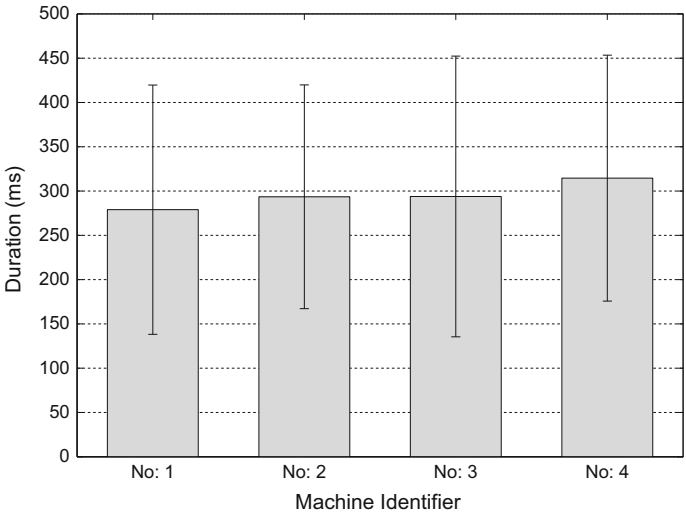

Fig. 8 Duration from the start of a solicitation message until its reply arriving to the initiator

increase in the number of packets decreases the rate of the successful message reception. It is also observed that the second machine has higher drop rate with respect to others. This shows that the retransmission threshold should be determined by considering different devices and the noise levels that they encounter.

Until now traffic shaping is not employed in the experiments. As mentioned in Sect. 3.4, traffic shaping is used to avoid congestion. Fragmented parts of a message are transmitted by inserting sleep intervals among them. Figure 11 presents the effect of the traffic shaping

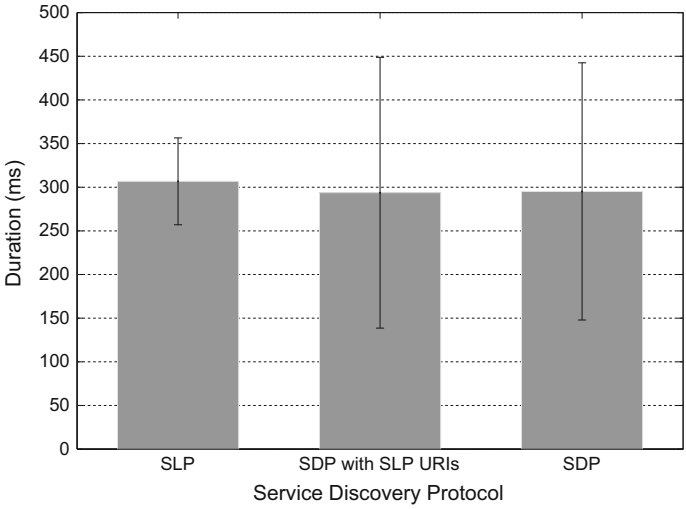

Fig. 9 Comparison of pure SLP solicitation messages and SLP URIs carried in SDP with ontologies

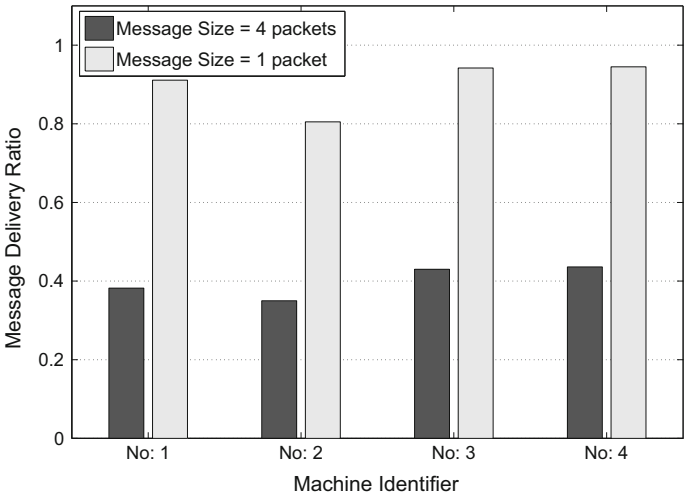

Fig. 10 Successful arrival ratio of the advertisement messages in different machines with different metadata sizes

on the dropped messages. Among the partially arrived messages, packets are grouped according to their sequence ids. Without traffic shaping, as the id increases the reception rate decreases. When traffic shaping is employed the reception rate becomes the same for all. Moreover, the overall successful transmission rate increases from 0.4 to 0.5. Exceptionally, in this example the reception rate for the last packet is larger than the others even if the traffic shaping is enabled. The reason is that last packet is smaller than the others. Other packets use the whole MTU (1500 bytes). However, the last packet carries the rest of the message which is 800 bytes and small packet effected from the noise less than the longer ones.

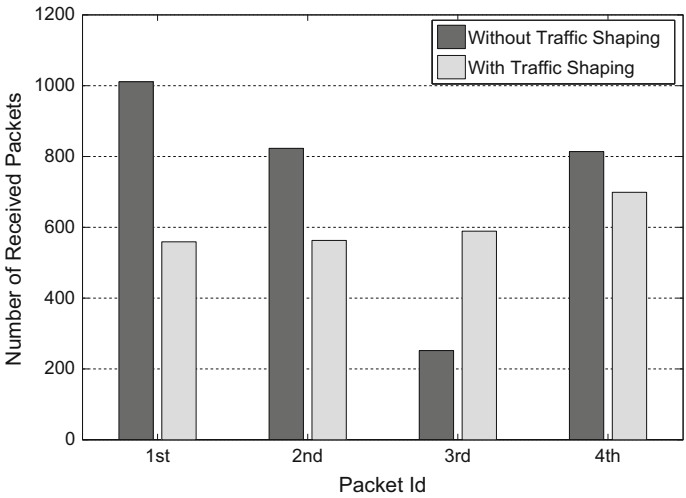

Fig. 11 A message is composed of four packets. Among the dropped messages, the number of received packets with respect to their ids are given. The last packet has smaller size and hence its success rate is higher

5 Existing Architectures

Service discovery is an active research area started in 1990s. There are many proposals up to now. In this section, we present some of the well-known and widely-supported service discovery protocols and compare them with SDP.

5.1 Service Location Protocol

Service Location Protocol (SLP) is a standard defined in RFC 2608 and RFC 3224. The main communication protocol is UDP multicasting. Unicast TCP messages are also used for long messages to improve the reliability. The messages are in URI formats like "service:printer-detector.1234://example.com:8080" which requires exact matching. Three different roles exist:

- *User Agent (UA)*: looks for the services,
- *Service Agent (SA)*: provides the service and announce it,
- *Directory Agent (DA)*: stores the list of services to solve scalability issues. It is optional.

In the absence of the DA, the system works in a distributed fashion. Otherwise, the service announcements are cached by DA and again the search queries are replied by DA.

5.2 Universal Plug and Play

Universal Plug and Play[4] (UPnP) is a set of network protocols which aims at seamless discovery and control of devices without human intervention. Leading companies in the

[4] http://www.upnp.org/.

electronics industry support the research on UPnP and some end products have already been commercialized. The devices such as computers, network printers, smart phones, televisions discover each other when they are attached to the same network. Then, they can exchange data and configuration parameters. It should be also noted that UPnP is more than just a service discovery protocol, it is an architecture in which pervasive devices control and exchange data among each other in a peer-to-peer way.

Simple Service Discovery Protocol (SSDP) [6] is an outdated IETF Draft but adopted by UPnP community as the service discovery protocol. Similar to SLP, it is based on multicast search messages. However, the protocol used for transportation is HTTPU (HTTP over UDP). UDP is preferred for HTTP transmission to reduce the overhead of TCP signaling and to use multicast instead of unicast. SSDP client multicasts an HTTPU discovery message to a predefined multicast channel. The services which listen to the channel replies with unicast HTTPU messages when the queried service matches. Apart from this request-response scheme, services can also announce their presence when they first join into the network.

Unique Service Names (USN) are URIs which uniquely define the services. USNs are used to handle the change of the point of attachment of the services in the network. An example of request and response message from [6] is given in Fig. 12. As it is seen, it is an HTTP message whose payload involves some predefined key, value pairs like "Host".

5.3 Device Profiles for Web Services

Device Profiles for Web Services (DPWS) [9], proposed by Microsoft, is similar to UPnP, designed as a plug-and-play architecture which involves discovery, control, and eventing of the services. Differently from UPnP, every service is considered as a web service and therefore all the standards depends on web services [9]: WSDL 1.1, XML Schema, SOAP 1.2, WS-Addressing, and further comprises WS-MetadataExchange, WS-Transfer, WS-Policy, WS-Security, WS-Discovery and WS-Eventing.

WS-discovery [10] is the service discovery protocol used in DPWS. SOAP over UDP is chosen as the transport protocol and messages are multicast to enable ad hoc mode of operation. Besides the ad hoc mode of operation, there is a managed mode in which a centralized proxy exists to coordinate the traffic. There is also a dynamic mode which combines both ad hoc and managed schemes. Centralized mode is mainly proposed to reduce the multicast traffic load in the network.

5.4 Zero-Configuration Networking

Zero-Configuration Networking[5] (Zeroconf) uses Multicast DNS/DNS-SD (IETF Draft standard) for service discovery. Multicast DNS enables well-known Domain Name System (DNS) application without the existence of a central server. Devices can query the services using the multicast messages. Similar to the other service discovery protocols, a standardized set of URIs are used to identify the services. Mainly, it is being supported by Apple Inc. in the name of Bonjour[6] with Apache 2.0 License and as another open source implementation for Linux and BSD machines in the name of Avahi.[7]

[5] http://www.zeroconf.org/.

[6] https://developer.apple.com/opensource/.

[7] http://avahi.org/.

```
M-SEARCH * HTTP/1.1
S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6
Host: 239.255.255.250:reservedSSDPport
Man: "ssdp:discover"
ST: ge:fridge
MX: 3

----------------

HTTP/1.1 200 OK
S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6
Ext:
Cache-Control: no-cache="Ext", max-age = 5000
ST: ge:fridge
USN: uuid:abcdefgh-7dec-11d0-a765-00a0c91e6bf6
AL: <blender:ixl><http://foo/bar>
```

Fig. 12 Example of SSDP request and reply messages

DSN-SD (DNS-Service Discovery) uses different record schemes for identifying and configuring the services. The Service (SRV) (RFC-2782) records are in the form of "*Instance.Service.Domain*" and defines the target host-port pair of the service address. The DNS Text (TXT) records are "*key=value*" pairs and are used to provide additional information about the service such as the queue name of a printing machine. DNS Pointer (PTR) (RFC 1035) records are used in the form of "*Service.Domain*" to discover available instances of a service.

5.5 Semantic Service Discovery

All of the above protocols employ text-based matching which can be considered as a problem. As mentioned in Sects. 2.1.1 and 2.2, ontologies are better options for representation. Multicast Resource Discovery Protocol (mRDP) [15] is a semantic service discovery protocol. OWL is used as the message format. Only solicitation messages are allowed, advertisement is not considered. The solicitation messages are over multicast HTTPU packets for scalability and the responses are in unicast HTTP packets to guarantee the delivery. Although semantic data is carried as in SDP, the choice of the transport protocol differs significantly. In SDP, instead of heavy protocols like HTTP, ICMPv6 is used that operates on the network layer and low power devices are addressed with legacy support. SDP offers authentication, integrity via SEND. Advertisement messages are supported which helps caching the services and decrease the response time.

UPnP architecture may also be mixed with the semantic languages [13]. UPnP messages can be converted to ontologies on which other devices may infer. The ontology created in [13] may be used in the SDP since SDP does not restrict itself in one language.

INDISS [3] interoperability system and in general the Amigo[8] project uses OWL-S[9] (Semantic markup for web services). In another example, home device interoperability is improved by using ontologies on top of SOA based service discovery protocols such as UPnP and DPWS [5].

There are some works in pervasive and sensor network environments. The proposed service discovery scheme in [16] tries to convert the natural language queries to machine understandable ones for the sensor like small devices. In [12], service discovery algorithms

[8] www.amigo-project.org.

[9] http://www.w3.org/Submission/OWL-S/.

for pervasive environments are proposed which aim to guide service discovery with the context information or personal preferences of the users.

When we move to the web domain, there are many works that concentrate on semantic web services. Researchers try to develop new algorithms for better matching of the services for the requirements of the users. In [11], an ontology framework which categorizes the services according to their functions is proposed. S-MatchMaker [7] improves service discovery by involving quality of the services in the selection process. Semantic web services research supports our research with the tools that are used in the matching of the web services. Most of these algorithms can be incorporated in our work as a back-end system for service matching.

5.6 Comparison with SDP

Many organizations have proposed state-of-the-art protocols like UPnP, DPWS and Bonjour which target service discovery satisfying specific sets of requirements making those protocols better than others in one way or another. Due to the prominent and distinctive features of these protocols and with the competitive support of the companies behind them, there is still no dominant service discovery mechanism. As a result there are disconnected islands of devices that can only interact with compatible ones belonging to the same vendor. Another main incompetency of the present protocols is their inability to infer beyond the shared pieces of service definitions. The operational service discovery architectures are based on simple text matching of the service descriptions. Generally, a URI (e.g., *service:printer-detector.1234://example.com:8080*) is published in a network. Other devices that can look up and match the text are able to recognize and consume the service (the printer in this example). When a new type of service is developed, the standardization bodies must come up with a new URI that define the service, and all the machines should upgrade their data stores. While existing services evolve; new services appear everyday. That is why the devices should embrace the change by inferring the meaning of a service by themselves.

Our proposal, SDP, is a service discovery protocol that carry semantic representations of the resources and the capabilities of the devices which also include the owner information. Comparison of SDP with other service discovery protocols are given in Table 3. Firstly, all the protocols support ad hoc mode operation and use multicast messages for scalability. Ad hoc mode is crucial for the networks without a central authority.

Recent proposals on service discovery have a tendency on employing more expressive representations of the services like XML in WS-Discovery and ontologies in mRDP. Many researchers agree on the expressive capabilities of ontologies and the importance of such intelligent architectures that will minimize human intervention. SDP also motivates the use of semantics. However, main contribution of SDP is the distribution protocol which operates on the network layer to eliminate heterogeneity. SDP can crawl in different network architectures provided that they use IP as the network layer. Fortunately, IP is becoming a convergence point for many different network architectures.

Another contribution of SDP is that it combines different service discovery protocols by carrying their service identifiers together. The TLV fields of ICMPv6 packets can store the identifiers, and when the peer network stack gets the identifier it pushes the identifier to the original handler of the standard. However, in the long run with the involvement of vendors, service representations can converge to ontologies.

Table 3 Comparison of different service discovery protocols

	SLP	SSDP	DNS-SD	WS-Discovery	mRDP	SDP
Carrier protocol	UDP & TCP	HTTPU	DNS	SOAP& UDP	HTTPU& HTTP	ICMPv6
Operating in heterogeneous networks	–	–	–	–	–	✓
Ad hoc mode	✓	✓	✓	✓	✓	✓
Service description	Text based (URI)	Text based (USN)	Text based (DNS-SRV)	Structured text (XML)	Ontology (OWL)	Ontology (OWL) & Text based (ALL)
Evolves for future requirements	–	–	–	–	✓	✓
Unified discovery	–	–	–	–	–	✓
Inference	–	–	–	–	✓	✓
Advertisement Message	SAAdvert	ssdp:alive	–	Hello	–	Multicast SA
Resource information	–	–	–	–	–	✓
Context information	–	–	–	–	–	✓
Personalized	–	–	–	–	–	✓
Reliability	Long messages in TCP	Retransmission	Retransmission	Retransmission	Reply with HTTP	Retransmission
Authentication and Integrity	–	–	DNSSEC	SSL or WS-Sec	Partial SSL	SEND
Confidentiality	–	–	–	–	–	–

6 Conclusions

Devices are not *swiss army knives* that incorporate all the required functionality in one item. They need to cooperate and share their functionalities with others. As a first step towards the cooperative networks of devices, devices should discover each others' services. Operational service discovery protocols that we use today do not adapt to future requirements. They use simple text-based representations of the services instead of more expressive ontologies that allow inferencing on gathered information. Moreover, existing service discovery protocols create their own islands where only devices from the same vendor can participate in communication.

The Smart Discovery Protocol proposed in this paper is a semantic service knowledge discovery protocol that operates at the network layer. Being embedded in the operating system, SDP is independent of the application layer protocols and the communication interfaces. Inspired by the Neighbor Discovery Protocol, ICMPv6 messages carry service definitions and service queries. The resources and capabilities are carried together with context and owner information. In this work we showed that the protocol can scale and reliability can be improved by rate limitation.

Semantic Web tools and vocabularies like OWL and SPARQL are used to describe the services. The URIs used in existing service discovery protocols like SLP are embedded in TLV options of the ICMPv6 packets. In this way, several discovery messages are unified in one message.

SDP still requires extensive testing in heterogeneous networks to determine the parameters like traffic shaping and number of retransmissions. SDP works in ad hoc mode. However, many other discovery protocols employ optional centralized servers that store all the service details, to decrease the number of messages sent in the network. Although we do not expect any issues in home networks, in enterprise networks a storage server may be required for scalability. In such a case a central scheme should be developed. Fortunately, ICMPv6 packets are parsed by the routers. Therefore, routers can act as storage servers.

Lastly, for the widespread use of the protocol, we need to develop an API for the application developers. This API must also provide features for manipulating the semantic data.

Acknowledgments We would like to acknowledge Prof. Geert-Jan Houben for his comments on the use of Semantic Web technologies and acknowledge Prof. Koen Langendoen for helping us in the organization of the paper. This work has been partially supported by Trans-sector Research Academy for complex Networks and Services (TRANS) project and METU BAP-08-11-2014-025. The source code of the Smart Discovery Protocol which is licensed with Apache License, Version 2.0, can be found in "github.com/yunus/SDP".

References

1. Berners-Lee, T., & Connolly, D. (2011). Notation3 (n3): A readable rdf syntax. www.w3.org/TeamSubmission/n3/
2. Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, *284*(5), 28–37.
3. Bromberg, Y. D., & Issarny, V. (2005). Indiss: Interoperable discovery system for networked services. In: Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware, Middleware

'05, pp. 164–183. Springer, New York Inc, New York, NY, USA. http://dl.acm.org/citation.cfm?id=1515890.1515899
4. Cisco: Cisco visual networking index: Global mobile data traffic forecast update, 2011–2016 (2014). http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html. Accessed 11 Sep 2014
5. El Kaed, C., Denneulin, Y., Ottogalli, F. G., & Mora, L. (2011). Combining ontology alignment with model driven engineering techniques for home devices interoperability. In: Software Technologies for Embedded and Ubiquitous Systems, Lecture Notes in Computer Science, Vol. 6399, pp. 71–82. Berlin, Heidelberg: Springer.
6. Goland, Y. Y., Cai, T., Leach, P., Gu, Y., & Albright, S. (1999). Simple service discovery protocol (ssdp). https://tools.ietf.org/html/draft-cai-ssdp-v1-03
7. Lemos, F., Gater, A., Grigori, D., & Bouzeghoub, M. (2012). A framework for service discovery based on structural similarity and quality satisfaction. In M. Brambilla, T. Tokuda, & R. Tolksdorf (Eds.), *Web engineering, lecture notes in computer science* (Vol. 7387, pp. 481–485). Berlin, Heidelberg: Springer.
8. Mutka, M., & Ni, L. (2005). Service discovery in pervasive computing environments. *IEEE Pervasive Computing*, *4*(4), 81–90. doi:10.1109/MPRV.2005.87.
9. Nixon, T., Regnier, A., Driscoll, D., & Mensch, A. (2009). Oasis devices profile for web services (dpws) version 1.1. http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01
10. Nixon, T., Regnier, A., Modi, V., & Kemp, D. (2009). Web services dynamic discovery (ws-discovery) version 1.1. http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html
11. Paliwal, A., Shafiq, B., Vaidya, J., Xiong, H., & Adam, N. (2012). Semantics-based automated service discovery. *IEEE Transactions on Services Computing*, *5*(2), 260–275.
12. Rasch, K., Li, F., Sehic, S., Ayani, R., & Dustdar, S. (2011). Context-driven personalized service discovery in pervasive environments. *World Wide Web*, *14*, 295–319.
13. Togias, K., Goumopoulos, C., & Kameas, A. (2010). Ontology-based representation of upnp devices and services for dynamic context-aware ubiquitous computing applications. In: Communication Theory, Reliability, and Quality of Service (CTRQ), 2010 Third International Conference on, pp. 220–225.
14. Tsarkov, D., & Horrocks, I. (2014). Fact++ description logic reasoner. http://code.google.com/p/factplusplus/. Accessed 11 Sep 2014.
15. Vazquez, J. I., & de Ipia, D. L. (2007). mrdp: An http-based lightweight semantic discovery protocol. *Computer Networks*, *51*(16), 4529–4542. doi:10.1016/j.comnet.2007.06.017.
16. Yang, S., Xu, Y., & He, Q. (2011). Ontology based service discovery method for internet of things. 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, pp. 43–47.

Yunus Durmus received his BS and MS degrees in computer engineering from Bogazici University, Istanbul, in 2007 and 2009. He is a student IEEE member and currently pursues a PhD degree at the Embedded Software Group in University of Technology Delft. His research interests include distributed systems with a focus on cooperation and identity management.

Ertan Onur received the BSc degree from Ege University, Turkey in 1997, and the MSc and PhD degrees in computer engineering from Bogazici University, Turkey in 2001 and 2007, respectively. During the MSc and PhD degrees, he worked as a project leader at Global Bilgi and as an R&D project manager at Argela Technologies, Turkey. After having worked as an assistant professor between 2008 and 2013 at Delft University of Technology, the Netherlands, he has been working as an associate professor at the department of computer engineering of Middle East Technical University, Turkey. Dr. Onur's research interests are in the area of computer networks and network security. He is a member of IEEE and ACM.