VSDM: A Virtual Service Device Management Scheme for UPnP-Based IoT Networks

Golam Kayas
Dept. of Computer & Info. Science
Temple University, USA
golamkayas@temple.edu

Mahmud Hossain
Dept. of Computer Science
University of Alabama at Birmingham, USA
mahmud@uab.edu

Jamie Payton
Dept. of Computer & Info. Science
Temple University, USA
payton@temple.edu

S. M. Riazul Islam
Dept. of Computer Engineering
Sejong University, South Korea
riaz@sejong.ac.kr

***Abstract*—The ubiquitous nature of IoT devices has brought new and exciting applications in computing and communication paradigms. Due to its ability to enable auto-configurable communication between IoT devices, pervasive applications, and remote clients, the use of the Universal Plug and Play (UPnP) protocol is widespread. However, the advertisement and discovery mechanism of UPnP incurs significant overhead on resource-constrained IoT devices. In this paper, we propose a delegation-based approach that extends the UPnP protocol by offloading the service advertisement and discovery-related overhead from resource-limited IoT devices to the resource-rich neighbours of a UPnP-enabled IoT network. Our experimental evaluations demonstrate that the proposed scheme shows significant improvement over the basic UPnP, reducing energy consumption and network overhead.**

***Index Terms*—UPnP, Internet of Things, Service Discovery, Service Advertisement, Virtual Service Device, Delegation.**

I. Introduction

The past decade has seen substantial growth in the manufacturing of Internet of Things (IoT) devices. As the availability of a range of different types of sensor- and actuator-enabled IoT devices at the network's edge has increased, a number of IoT deployments have emerged, including smart home systems [1] that integrate home security [2], management, and convenience for home owners; health and wellness services [3]–[5] that allow for remote, continuous, multi-modal monitoring of physiological and behavioral characteristics [6]; and intelligent agriculture systems [7] that can optimize crop growth and harvesting. Common to these and future IoT application deployments is the need for an open system that supports fluid, dynamic, opportunistic integration of potentially mobile clients and services supported by IoT devices at scale, with minimal configuration.

Given these needs, service-oriented architectures are well-suited to support IoT-enabled systems, providing the ability for resource-constrained IoT devices to advertise software services that can be discovered and used by applications. The Universal Plug and Play (UPnP) [8] protocol, which implements a service-oriented model and can be used to connect clients and services across a network, has been widely adopted for use to support IoT systems. UPnP supports dynamic service advertisement and service discovery, and supports zero configuration networking. It also offers language independence and interoperability, supporting the incorporation of a wide variety of devices from different manufacturers with varying capabilities and configurations.

However, supporting UPnP on the types of small, resource-constrained devices that are common in IoT networks is not without challenges. In particular, despite recent advances in device and network protocol design, the substantial power consumption required to send and receive messages via a wireless medium makes communication overhead a significant concern for small, battery-powered sensor devices. Specifically, in the UPnP context, an IoT device acting as a service device (SD) typically needs to periodically multicast advertisements to announce the availability of the services it provides. The advertisement message also contains the information needed to access the services provided by the SD. As the IoT devices are resource-constrained and usually battery-powered, this periodic multicast advertisement message incurs considerable energy overhead.

In this work, we build on the observation that resource-constrained IoT devices at the edge of the network are often connected to a more resource-rich neighbours, such as a gateway device. In general, a gateway device act as a bridge between heterogeneous IoT devices within the local area network and remote devices, performing protocol translation, packet fragmentation, and communication bridging between different communication technologies (e.g., BLE [9], Zigbee [10], 6LoWPAN [11]). With respect to supporting UPnP in IoT networks, the gateway can also serve to assist resource-constrained IoT service devices by supporting delegation of service advertisements. Essentially, the gateway device can act as a *Virtual Service Device (VSD)*, which emulates the advertisement behavior of an SD; it handles sending the

978-1-7281-9656-5/20/$31.00 ©2020 IEEE

service advertisement and serves the service discovery requests from interested clients. This approach can reduce the network overhead, and therefore the energy consumption, incurred on IoT service devices in the UPnP-enabled network.

In this work, we extend the UPnP protocol to incorporate service advertisement delegation from IoT service devices to a resource-rich Virtual Service Device (VSD) and propose a Virtual Service Device Management (VSDM) scheme. The contributions of this work are summarized as follows:

- We identify the energy consumption of UPnP service devices to advertise the targeted services of an IoT network.
- We propose a Virtual Service Device Management (VSDM) scheme, delegating the service advertisement overheads to a resource-rich network member for resource constrained IoT devices.
- We implement a prototype of VSDM and conduct an experimental evaluation. Our experiments focus on exploring the energy consumption and network overheads in basic UPnP and VSDM-enabled UPnP in an both IoT network.

II. Background

A. UPnP based IoT Systems

Figure 1 shows an example UPnP network with IoT devices. In a UPnP network, devices can be located in different types of networks such as BLE, Zigbee, and 6LoWPAN. The participants from different communication mediums can interact with each other to perform UPnP operations. For example, a smart phone that uses WiFi can act as a CP to discover a service provided by an IoT device in the 6LoWPAN network. The gateway device in responsible in bridging different communication technologies.

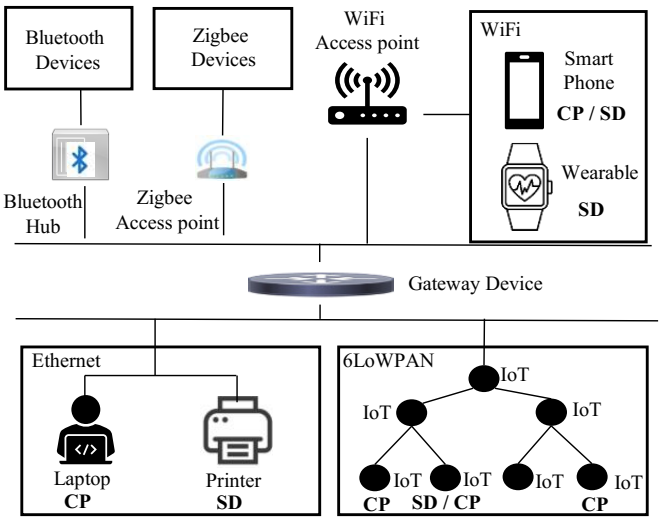


Fig. 1: A UPnP Network with IoT devices.

B. UPnP Components

The components of a UPnP based system are classified into three categories: Service Device (SD), Control Point (CP), and Service.

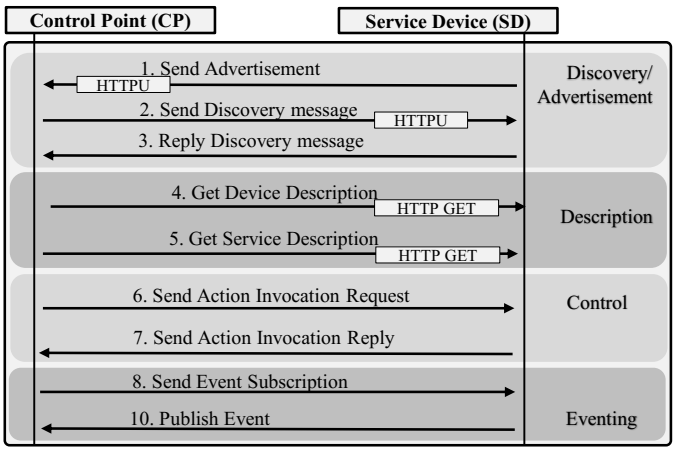


Fig. 2: Interaction between a CP and an SD.

Service Advertisement	Service Discovery
NOTIFY * HTTP/1.1 HOST: 239.255.255.250:1900 CACHE-CONTROL: *expiration time* LOCATION: *URL for description* NT: *notification type* NTS: ssdp:alive USN: identifier for the *advertisement* Other optional fields: *value*	M-SEARCH * HTTP/1.1 HOST: 239.255.255.250:1900 MAN: ssdp:discover MX: *seconds to delay response* ST: *search target* USER-AGENT: *optional field*

Fig. 3: UPnP Advertisement and Discovery messages.

1) Service Device (SD): In UPnP, an SD functions as a server that provides useful services to the clients upon request. In IoT-based scenarios, SDs are embedded with sensors that collect contextual information, actuators that can perform actions in response to sensed information, and radio transceivers for communication, with the support of a real-time operating system. For example, a service device may be a smart refrigerator, doorlock, or security camera in a smart home network.

2) Control Point (CP): A CP acts as a client that requests and consumes the services provided by the SDs. A CP can discover the available services, control the services, and request updates on the state change of the services.

3) Service: A service is a unit of functionality implemented by an SD located on the edge of the network. For example, a smart refrigerator device may offer a service to check the temperature of the vegetable drawer or to add an item to a shopping list.

C. UPnP Phases and Operations

Figure 2 shows the operations performed in UPnP communication by the SDs and the CPs in different phases. As shown in the figure, there are four essential phases of the UPnP protocol: Advertisement, Discovery, Description, Control, and Eventing.

1) Advertisement: In the UPnP protocol [12], an SD periodically sends advertisement messages to the network by

multi-casting to a standard address and port. Figure 3 shows an advertisement message sent by an SD. In the advertisement message (Figure 3), the SD provides a URI (via the `LOCATION` field) that allows the CP to retrieve additional information about the device and its services. An SD will also unicast a similar message in response to a discovery message received from a CP.

2) Discovery: A CP searches a desired service in the network by broadcasting a discovery message. Figure 3 shows a sample discovery message sent by a CP. A CP defines the target service of the discovery message using the `ST` field (See Figure 3). In reply of a discovery message, the CP receives a message similar to the advertisement revealing the description of the services.

3) Description: After the advertisement and discovery, a CP only knows a URI location that provides the details of the services by a particular SD. In the description phase, the CP requests the URI location to retrieve the device description of the SD.

4) Control: After retrieving the description of the services in the previous phase, a CP knows the name of the actions supported by the service, their parameters, and the way to invoke them. The CP sends control request to the service to perform the targeted actions.

5) Eventing: Additionally, a CP also knows all the state variables of a service from its description and can subscribe to monitor the state variables. A subscription is when the targeted state variable of the specific services changes, every subscriber (in this case the CP) gets a notification from the SD.

For IoT devices, periodic multicasting of advertisement message can be prohibitively costly in terms of energy consumption due to communication overhead. Additionally, IoT service devices are vulnerable to service discovery flooding attacks, in which a malicious CP repeatedly sends service discovery messages; at a minimum, the IoT service device will incur significant communication overhead (and therefore, energy consumption) in responding to high volumes of service discovery requests and the IoT service device can be overwhelmed by the requests, effectively resulting in a a denial of service. We discuss these limitations of UPnP service advertisement in further detail in Section III to motivate our proposed scheme.

III. Proposed Scheme: VSDM

To support UPnP in IoT networks, an IoT SD multicasts periodic advertisement messages, which consumes significant energy for a resource-constrained device. The IoT SD must also reply to discovery requests of the CPs. As such, IoT service devices are vulnerable to large volumes of service requests, whether originating from legitimate control points or malicious actors. In any case, so called "discovery flooding" incurs significant communication overhead on the IoT SD, which must respond to the high volumes of service discovery requests, and can ultimately exhaust the energy in a battery-powered IoT device. The IoT service device's buffer and processor can also be overwhelmed by the requests, effectively resulting in a different type of a denial of service, impacting the performance of the IoT system. Moreover, IoT devices are often deployed in low power and lossy networks (LLN) like 6LoWPAN. LLN networks are very constrained, and, in case of discovery flooding, these networks become congested, resulting in yet another type of performance impact.

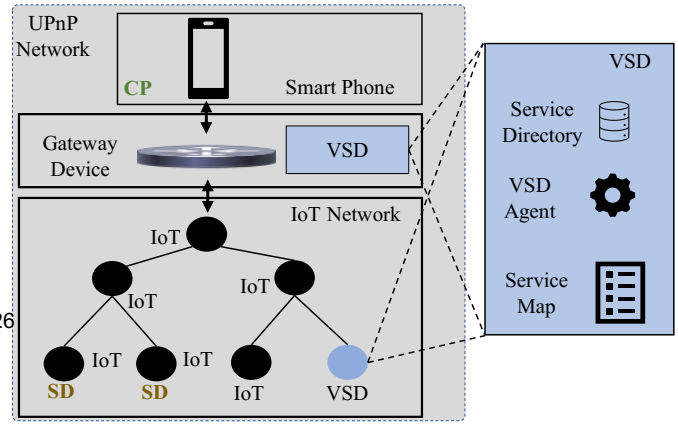


Fig. 4: VSDM System Components.

In response to aforementioned limitations of deploying UPnP service advertisement in IoT networks, we propose a Virtual Service Device Management (VSDM) scheme to offload the advertisement and discovery reply to a resource-rich device in the IoT network. In VSDM scheme, the periodical advertisement and discovery reply is delegated to a *virtual service device (VSD)* which is a resource-rich member of the UPnP enabled IoT network. The VSDs take care of the advertisement and discovery reply, offloading the resource constrained IoT SDs. For later UPnP phases after discovery and advertisement, such as action request (control) and event subscription (eventing), the VSD redirects the requests to the SD. In this work, we focus on presenting a solution that extends the UPnP protocol to incorporate a single VSD in an UPnP-enabled IoT networks. However, we note that multiple VSDs can be deployed in a VSDM-enabled UPnP solution for IoT networks. Such a solution can offer additional benefits, including load balancing across VSDs and fault tolerance.

A. VSDM Components

Figure 4 shows the system components of VSDM scheme. The `Virtual Service Device` (VSD) is implemented in the gateway device or in a resource-rich device in the same communication network. As most IoT networks are accompanied by relatively resource rich gateway devices, they are excellent candidates to implement a VSD. However, the gateway device is not the only candidate to become a VSD; any other participants that have enough resources can eventually implement the characteristics of the VSD. Every VSD implements a service named `VSD-Agent`, keeps a `Service Directory`, and maintains `Service Map`.

1) VSD-Agent: The VSD-Agent is the service implemented by every VSD to support delegation. The description of the VSD-Agent service is maintained in a service description

document. Figure 5 shows the JSON envelope of the service description document of the VSD-Agent service. This description document provides the URL to send action invocation requests, and exposes the name and arguments of the actions to be invoked. There are four actions implemented by the VSD-Agent service:

- **Service-add:** `Service-add` is invoked by a SD when it tries to enroll itself to the VSD for delegation or it has a new service to provide. This action takes a list of service information as a parameter. Each entry of the service information list is a tuple like, `<service-name, service-type, description-location-url, control-url, event-url>`. `Service-add` retrieves the service description from the original SDs and stores them in the Service Directory. It also inserts an entry to the Service-map mapping the delegation information.
- **Service-remove:** `Service-remove` is invoked by a SD when a previously advertised service is no longer available or the SD is leaving the UPnP network. This action also takes a list of service information as a parameter. `Service-remove` removes the delegated service description stored in the Service Directory and the Service-map entries for the associated services.
- **Service-update:** When a SD has an update for exiting service, it invokes the `Service-update` action. This action retrieves the new service description document from the SD and updates the Service-Directory and Service-map with the updated information.
- **Discovery-reply:** This action is invoked by the VSD that implements the VSD-Agent service upon receiving a discovery request from a CP. `Discovery-reply` takes the name of the targeted service and the address of the CP that issued the discovery request as a parameter. This action finds the targeted service name from the Service Map, constructs a delegated discovery reply, and sends the discovery reply to the CP.
- **Multicast-Advertisement :** This action is also invoked by the VSD periodically multicasting the advertisement for the available services. `Discovery-reply` iterates through the Service Map, constructs delegated advertisement messages for each entry, and multicasts the advertisements in the network.

2) Service Directory: VSD retrieves the service descriptions of the SDs and stores them into the Service Directory to perform the delegation. While advertising the service provided by an SD, VSD uses the corresponding service description stored in the Service Directory as the delegated service description location.

3) Service-Map: Service-Map keeps the mapping of a service name and type to the description, control and event URLS and the Owner SD that provides the original service. The Service Map is implemented as a hashtable with separate chaining. Figure 6 shows an example of the `Service Map` data structure. There the description location (`Location`) is a

```
{
 Type            :  "VSD",
 UUID            :  VSD-UUID
 ServiceList     : [
   {
    ServiceName : "VSD-Agent",
    Control-URL : <VSD-IP:PORT>/ACTIONS,
    Actions     : [
     {name: "Service-add",       args: <list-of-service-info>},
     {name: "Service-remove",    args: <list-of-service-info>},
     {name: "Service-update",    args: <list-of-service-info>},
     {name: "Multicast-Advertisement", args:<target-service>},
     {name: "Discovery-reply",   args:<CP-address, target-service>}
    ]
   }
  ]
}
```

Fig. 5: The JSON envelope of the VSD service description document.

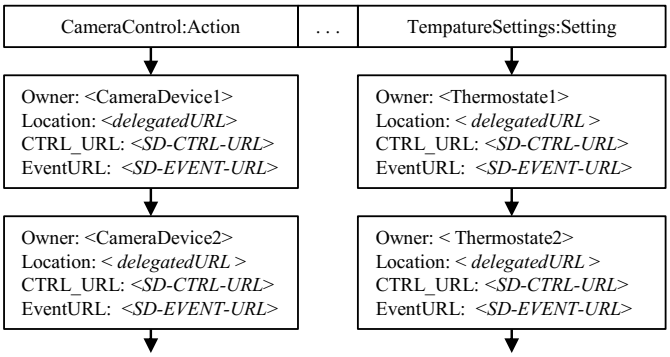


Fig. 6: Structure of the Service Map.

delegated URL pointed to a URL stored in Service Directory. The control URL (`CTRL_URL`) and the event URL are the absolute URL's provided by the SD.

B. SD Enrollment

Figure 7 shows the enrollment process for a SD. **Step 1:** When a new SD enters the UPnP network, it multicasts a VSD discovery message in the network to find the VGD-Agent service. The VSD discovery message is based on the `SSDP M-SEARCH` message like UPnP discovery, where the search target (`ST`) is (`VSD:VSD-AGENT`). **Step 2:** The VSD device replies to the SD with the description location of the VSD-Agent service (`LOCATION` field). **Step 3:** The SD sends a `HTTP GET` request to retrieve the description of the VSD-Agent service (see Figure 5). **Step 4:** The SD invokes the `Service-add` action of the VSD-Agent service with a list of the service information it wants to advertise in the network. **Step 5:** The VSD-Agent sends `HTTP GET` request to retrieve the service description documents of the services hosted by the SD. **Step 6-8:** After receiving the service documents of the SD, VSD-Agent stores them it the Service Directory and creates a delegated description URL. Then VSD-Agent

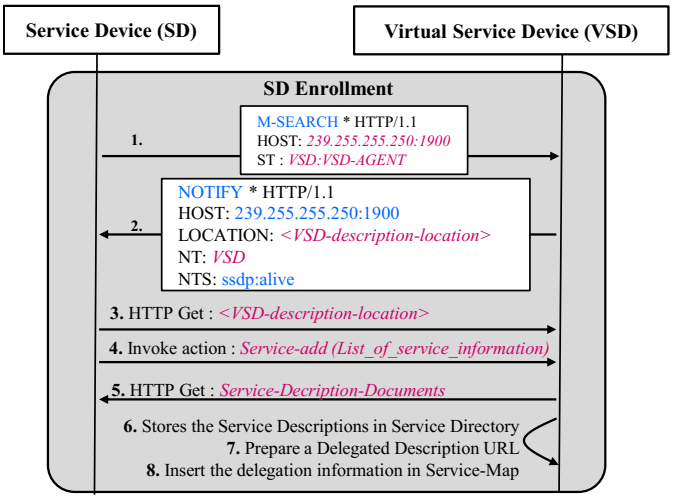


Fig. 7: Enrollment of an SD in VSDM.

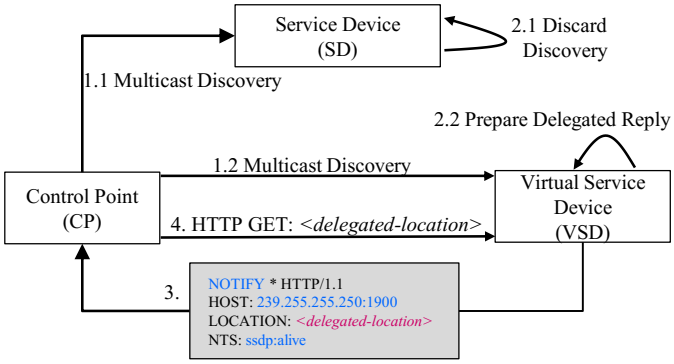


Fig. 8: Delegated Discovery and Description of VSDM.

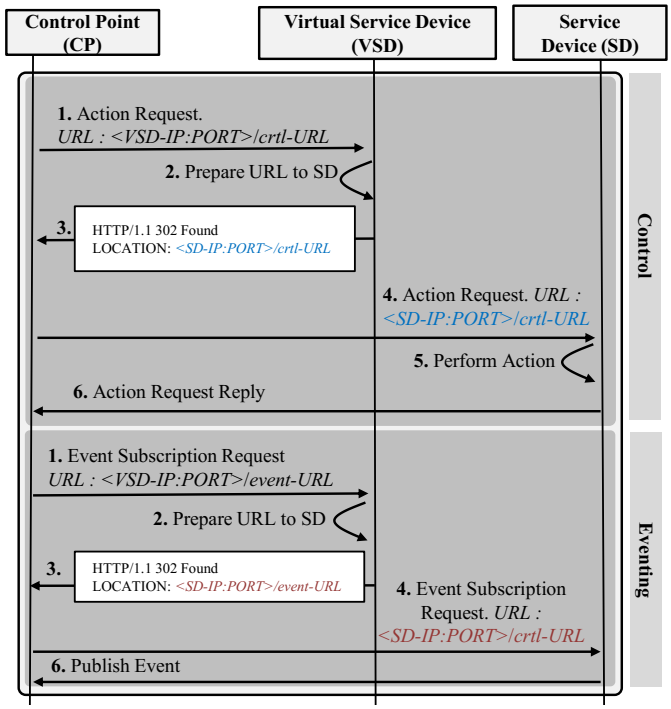


Fig. 9: VSDM Control and Eventing Phase.

inserts an entry to the Service-Map with the service name, type, delegated description URL, control and the event URL of the SD for each service.

C. Delegated Discovery Reply and Description

Figure 8 shows the delegated discovery and the description phases of VSDM. **Step 1:** The CP multicasts a discovery message in the network. **Step 2:** As the discovery message is sent using multicast, both the SD and the VSD receive it. As the SD delegates the discovery reply to the VSD, it ignores the discovery message. **Step 3:** The VDS invokes the `Discovery-reply` action of the VSD-Agent and prepares a discovery reply using the delegated `LOCATION` URL. **Step 4:** The CP uses the delegated description location to get the service description documents from the VSD.

D. VDSM Control and Eventing

Figure 9 shows the control and eventing phases of the VDSM. After getting the description of the services from the VSD, a CP performs action request invocation. As the CP gets the description documents from the VSD, naturally it appends the address of the VSD with the relative control URL provided in the service description to invoke action invocation request (Step 1 Figure 9). The VSD retrieves the absolute address of the control URL to the original SD from the Service-Map (see Figure 6). Then the VSD redirects the control request to the original SD using `HTTP 302` redirection (Step 3 in Figure 9). Thus the action invocation request of the CP is directly served by the original SD.

Similarly, the event subscription request is also redirected by the VSD to the original SD to publish the events to the CP.

IV. Experiment and Evaluation

A. Experimental Setup

The experimental UPnP network imitating a real-world UPnP enabled IoT setting is illustrated in Figure 10. We used Contiki operating system based simulator Cooja [13]. We build an IoT device network that consists of some physical device and a number simulated devices. The simulated IoT network, includes a number of Z1 [14] mote devices, with communications supported by 6LoWPAN and the RPL [15] routing protocol. In addition to the simulated Z1 devices that comprise the simulated IoT network, our experimental setup includes real physical devices; a desktop computer supporting the *VSD* is connected to a Wireless Access Point (WAP) through the Ethernet ($eth0$) interface, which, in turn, is connected to a physical Android smart phone, and a laptop computer. The WAP bridges the $eth0$ interface to the WiFi medium. A bridge is configured to connect the (simulated) 6LoWPAN network and the gateway device using the `Tunslip` utility [16] of Cooja. All the devices in the network use UDP as the transport

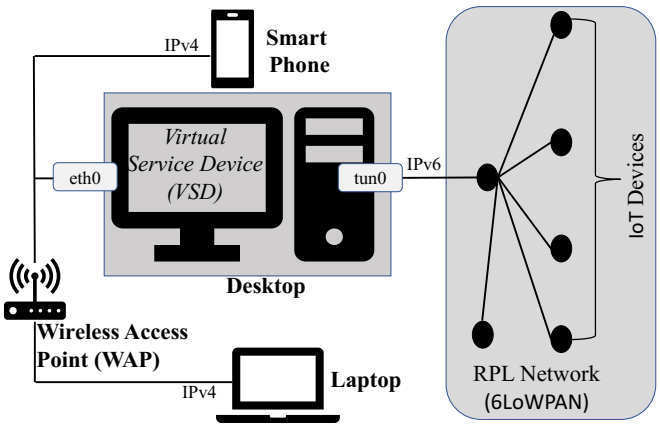


Fig. 10: Experimental Network of VSDM .

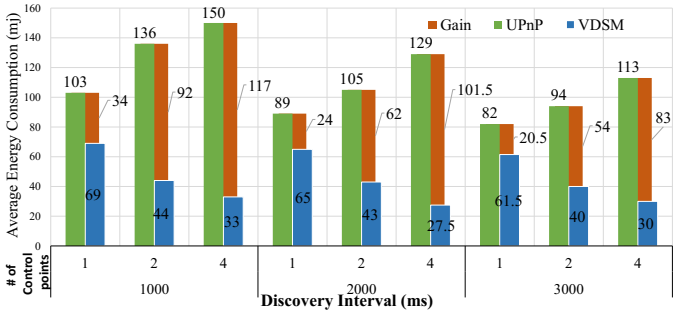


Fig. 11: Gain in energy consumption UPnP vs VSDM

layer protocol to exchange UPnP discovery and advertisement messages.

A simulated device is configured as a resource constrained SD which provides three UPnP services, each providing two actions that can be invoked. The rest of the simulated devices are configured as CPs, which issue discovery messages to search for UPnP services provided by the SD. The smart phone (connected via WiFi) runs an application that is configured as both a CP and a SD. The laptop is configured as a traditional CP and connects via the WiFi interface. A desktop computer serves as the gateway device in the experimental network.

Our experiments evaluated the performances of basic UPnP and VSDM explored the energy consumption of the resource constrained IoT devices. We also analyze the throughput of the IoT network in both tradition UPnP network (WiFi) and constrained IoT UPnP network (6LoWPAN).

B. Energy Consumption Analysis

In our experimental setup, the smart phone acting as a CP that sends discovery messages to the experimental UPnP network, searching for the services provided by the IoT SD located in the IoT network. In addition, we included IoT CPs, varying their number to explore how energy consumption is impacted as the number and source of discovery messages increases. We also varied the frequency of the discovery messages to 1000 ms, 2000 ms, and 3000 ms. We chose a fixed 2 min interval between advertisement messages for a service. Each round of our experiment was 20 minutes long. We repeated each experiment 10 times and reported the averaged outcomes. To explore power consumption, our experiments leveraged the Powertrace library [17] which used the Contiki's energy APIs to measure the power consumption of CPU and radio transceiver in the SD.

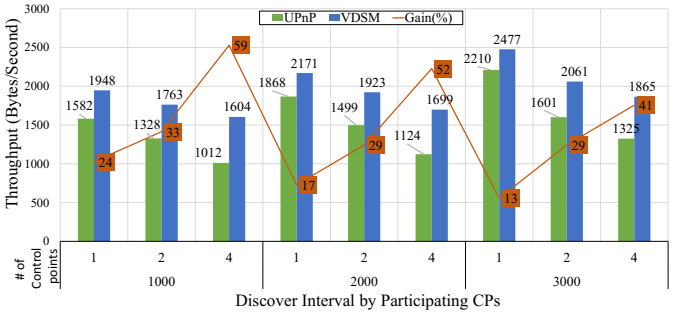
Figure 11 shows the result of the energy-consumption experiments. As expected, as we increase the number of CPs (and therefore, the number of discovery messages), we see an increase in the energy gain of the VSDM over basic UPnP. This is due to the service discovery delegation approach embodied in VSDM . With a VSD, the discovery message send by a CP is not received by the SP; instead the VSD receives and replies to the discovery messages on behalf of the SP. As a result, in VSDM , the SD replies to fewer messages compared to the basic UPnP. We see this benefit increase as the number of CPs who are requesting to discover services increases. For example, with discovery message frequency of 1000 ms, with 1, 2 , and 4 CPs the SD saved 34 mj, 92 mj, and 117 mj of energy, respectively. We also observed, with decreasing discovery frequency, the energy savings provided by VSDM decreases. As with less frequent discovery requests, the SD had less discovery reply to delegate. With 2 participating CPs, the SD saved 67.6%, 41%, and 57% of energy for 1000 ms, 2000 ms, and 3000 ms frequency, respectively. Our energy experiments show that the introduction of the VSD can dramatically reduce energy consumption due to discovery request and advertisement in resource-constrained IoT SDs in the VSDM compared to the basic UPnP protocol.

C. Throughput Analysis

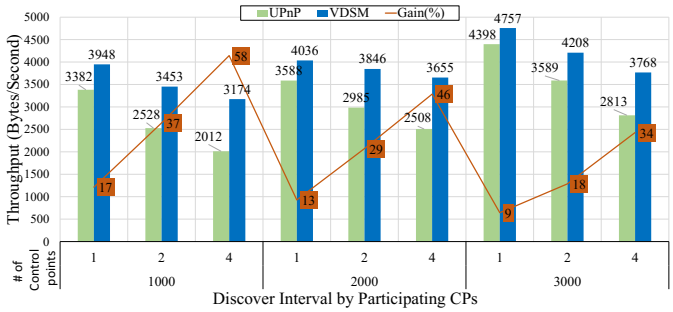
To measure the throughput of an UPnP, we choose a targeted CP and some participating CP. We assumed that the targeted CP already received the advertisement of the SD and know how to invoke service actions from the device and service . Th targeted CP send action invocation requests to the SD using the control messages of UPnP with an interval of 10 seconds. Meanwhile, the participating CPs were sending discovery messages in random intervals (between 1000 ms and 3000 ms inclusive) to the SD searching for the services. As we wanted to measure the effects of the discovery reply messages and advertisements in the service action invocation requests in an UPnP network, we measure the throughout for the targeted CP using the below formula:

$$Throughout = \frac{bytes\ received\ by\ SD\ for\ Service\ Access}{Total\ End\text{-}to\text{-}end\ delay\ to\ send\ the\ bytes}$$

Here, we define service access as an action invocation request (control message) or a event subscription request (eventing) sent by the targeted CP. In the resource-constrained IoT network experiments, the IoT devices act as SDs. The smart phone acts as a CP. Like in the previous experiments, we varied the number of IoT CPs, which sent discovery messages, and varied the intervals at which discovery messages were sent by



(a) Throughput comparison in IoT network.



(b) Through comparison in WiFi network.

Fig. 12: Throughput comparison of VSDM vs UPnP network.

the CPs to the IoT SD. Figure 12a shows the result of these experiments. In the traditional UPnP network experiments, we used the smart phone as the SD and the laptop as the targeted CP. Different number of IoT CPs also took part in the experiment by sending discovery message to the smart phones. Figure 12b shows the result of the experiments for traditional UPnP network.

For both traditional and resource-constrained IoT UPnP networks, VSDM outperforms the basic UPnP protocol. In this VSDM , the VSD served the discovery replies that enabled the SDs to serve the service action invocation requests quickly. We also observed that as the frequency of the discovery message increases, the throughput decreases in both UPnP and VSDM . However, throughput decreases in VSDM at a slower rate than for the basic UPnP. We also find that the throughput gain for VSDM is positively correlated with the number of participating CPs; as more CPs send more discovery messages, the SD offloads more work to the VSD, which results in increased throughput.

D. Comparative Discussion

Yiqin et al. [18] proposes a UPnP based networking solution to monitor and control the home appliances and smart devices by the remotes, smart phones, laptops etc in a home network. Arunachalam et al. [19] proposes an extension of UPnP to improve the interoperability among heterogeneous devices in IoT heavy home networks. The authors introduce an UPnP Application Architecture along with the UPnP application template and UPnP service template to develop applications that run on heterogenous devices. The proposed extension basically enhance the device interoperability of UPnP to application interoperability. In [20], researchers propose new Constrained Application Protocol (CoAP) [21] methods so that the UPnP services offered by a constrained network can be discovered via a CoAP/UPnP bridge co-located on the constrained network gateway. In the proposed method, UPnP messages are translated to the extended CoAP methods at the bridge and vice versa. Another similar approach [22] is proposed to bridge the Zigbee and the UPnP leveraging the low energy footprint of the Zigbee protocol to reduce the energy consumption of the UPnP devices in a constrained network. Researchers [23] propose a new service discovery protocol to make the UPnP service discovery more efficient and compact in IoT based IPv6 home networks. This extension take advantage of the improved design of IPv6 to build a new service discovery method that increase the efficiency of transmission.

The prior works do not recognise that the resource constrained IoT devices can reside with resource rich devices. And the overhead of performing UPnP service advertisement and discovery can be offloaded to resource-heavy counter parts of the constrained networks. Our proposed scheme achieves better energy efficiency by delegating the advertisement and handling the discovery requests in resource rich members of the constrained UPnP enabled IoT network.

s

V. Concluding Remarks

This paper proposes an extension of the basic UPnP protocol that delegates the service advertisements and discovery requests replies from the resource-constrained IoT devices to the resource-rich elements of an UPnP-enabled IoT network. The proposed scheme leverages the insight that the resource-limited IoT devices are often co-located with resource-rich neighbours that have the potential to carry out the UPnP tasks on behalf of the IoT devices. The prototype-based evaluation shows that the proposed extension outperforms the conventional UPnP in terms of energy consumption and network throughput.

In the given solution, we have proposed the VSDM to delegate advertisement and discovery related tasks from IoT service devices. We have argued that the use of multiple VSDs in the delegation process achieve more flexibility in the implementation. In future, it would be worth investigating, the delegation among a number of VSDs that can be optimized by distributing the delegated tasks to get improved performances. The trade-off between the number of VSDs placement and performance in an UPnP-based IoT network can also be subjected to future works.

Acknowledgements

This research was supported in part by the US National Science Foundation (NSF) under Grant No. CNS-1828363 and in part the Sejong University research faculty program under the Grant No. 20192021.

REFERENCES

[1] S. S. Chowdhury, S. Sarkar, S. Syamal, S. Sengupta, and P. Nag, "Iot based smart security and home automation system," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2019, pp. 1158–1161.

[2] M. Hossain and R. Hasan, "P-hip: A lightweight and privacy-aware host identity protocol for internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[3] Y. A. Qadri, A. Nauman, Y. B. Zikria, A. V. Vasilakos, and S. W. Kim, "The future of healthcare internet of things: A survey of emerging technologies," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1121–1167, 2020.

[4] J. Russell and J. Bergmann, "Probabilistic sensor design for healthcare technology," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2019, pp. 0585–0590.

[5] M. Hossain, S. R. Islam, F. Ali, K.-S. Kwak, and R. Hasan, "An internet of things-based health prescription assistant and its security system design," *Future generation computer systems*, vol. 82, pp. 422–439, 2018.

[6] F. Samie, L. Bauer, and J. Henkel, "Hierarchical classification for constrained iot devices: A case study on human activity recognition," *IEEE Internet of Things Journal*, 2020.

[7] D. Glaroudis, A. Iossifides, and P. Chatzimisios, "Survey, comparison and research challenges of iot application protocols for smart farming," *Computer Networks*, vol. 168, p. 107037, 2020.

[8] M. Jeronimo and J. Weast, *UPnP design by example: a software developer's guide to universal plug and play*. Intel Press, 2003.

[9] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[10] zigbee alliance, *Zigbee*, 2020 (accessed September 14, 2020). [Online]. Available: https://zigbeealliance.org/

[11] Z. Shelby and C. Bormann, *6LoWPAN: The wireless embedded Internet*. John Wiley & Sons, 2011, vol. 43.

[12] Open-Connectivity, *UPnP Device Architecture 2.0*, 2020 (accessed April 14, 2020). [Online]. Available: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v2.0.pdf

[13] Contiki, *An Introduction to Cooja*, 2019 (accessed July 14, 2020). [Online]. Available: https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja

[14] *Zolertia*, 2018 (accessed July 14, 2020). [Online]. Available: https://github.com/Zolertia/Resources/wiki/The-Z1-mote

[15] J. V. Sobral, J. J. Rodrigues, R. A. Rabêlo, J. Al-Muhtadi, and V. Korotaev, "Routing protocols for low power and lossy networks in internet of things applications," *Sensors*, vol. 19, no. 9, p. 2144, 2019.

[16] J. Romkey, "Rfc1055: Nonstandard for transmission of ip datagrams over serial lines: Slip," 1988.

[17] Contik, *Contiki APIs for Measuring Energy Consumption*, 2017. [Online]. Available: http://contiki.sourceforge.net/docs/2.6/a00452_source.html

[18] L. Yiqin, F. Fang, and L. Wei, "Home networking and control based on upnp: An implementation," in *2009 Second International Workshop on Computer Science and Engineering*, vol. 2. IEEE, 2009, pp. 385–389.

[19] K. Arunachalam and G. Ganapathy, "Extending upnp for application interoperability in a home network," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 4, p. 2085, 2017.

[20] J. Mitsugi, S. Yonemura, H. Hada, and T. Inaba, "Bridging upnp and zigbee with coap: protocol and its performance evaluation," in *Proceedings of the workshop on Internet of Things and Service Platforms*, 2011, pp. 1–8.

[21] IETF, *COAP*, 2014 (accessed September 14, 2020). [Online]. Available: https://tools.ietf.org/html/rfc7252

[22] S.-J. Kim, H.-M. Seo, W.-C. Park, S.-D. Kim, and Y.-S. Lee, "Seamless network bridge for supporting interoperability upnp-zigbee," in *2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*. IEEE, 2011, pp. 308–313.

[23] Z. Lin and C. Liao, "Icmpv6sd: A compact service discovery protocol supporting plug-and-play in home networks," in *2019 International Conference on Platform Technology and Service (PlatCon)*, 2019, pp. 1–6.