Future Generation Computer Systems 92 (2019) 617–627

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A framework for DNS naming services for Internet-of-Things devices

Keuntae Lee [a], Seokhwa Kim [a], Jaehoon (Paul) Jeong [b,*], Sejun Lee [a], Hyoungshick Kim [a], Jung-Soo Park [c]

[a] Department of Computer Science and Engineering, Sungkyunkwan University, Republic of Korea
[b] Department of Interaction Science, Sungkyunkwan University, Republic of Korea
[c] Electronics and Telecommunications Research Institute, Republic of Korea

ARTICLE INFO

Article history:
Received 1 August 2017
Received in revised form 16 December 2017
Accepted 10 January 2018
Available online 31 January 2018

Keywords:
IoT
DNS
Autoconfiguration
Device registration
Device discovery
Service discovery

ABSTRACT

This paper proposes a new naming framework for Domain Name System (DNS) Name Autoconfiguration (called DNSNA) for Internet of Things (IoT) devices in Internet Protocol (IP) version 6 (IPv6) and IP version 4 (IPv4) networks. As the number of IoT devices increases, it can be a burden for users to manually configure the DNS names of the IoT devices. We propose DNSNA to provide an efficient DNS name management for IoT devices, supporting an IoT device's DNS name auto-generation, its DNS name auto-registration, device discovery, and service discovery. A user can easily use the registered DNS information of the IoT devices for both device discovery and service discovery through DNSNA. DNSNA uses IPv6 Neighbor Discovery (ND) protocol for IPv6 IoT devices and Dynamic Host Configuration Protocol (DHCP) for IPv4 IoT devices, respectively, in the IoT DNS name autoconfiguration. With either ND or DHCP, the IoT devices can obtain a DNS suffix and construct their DNS name. Also, through a physical-contact-based authentication of a user's smartphone with an IoT device over Near Field Communication (NFC), an authenticated IoT device can automatically register its DNS name and the corresponding IP address into a local authoritative DNS server in a secure way. Through real experiments and analysis, it is shown that DNSNA can reduce the average number of packets by 60.8% and the accumulated packet volume by 97% in comparison with the existing DNS naming service (i.e., mDNS) for IoT devices in an IPv6 network.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

Nowadays, Internet of Things (IoT) has been researched widely and intensively. According to Gartner's report [1], the number of IoT devices will reach about 26 billion units by 2020. To efficiently manage such a large number of devices, it is recommended that each IoT device should have its own Domain Name System (DNS) name [2]. However, it is not easy for users to manually configure the DNS names of IoT devices because there are too many IoT devices to be configured. Thus, this DNS autoconfiguration will be one of key technologies for the efficient management and operations in a large-scale IoT network in near future.

The software called Bonjour, which was developed by Apple Inc., automatically assigns DNS names to Apple devices through the zeroconf-oriented software called Multicast DNS (mDNS) [3]. In mDNS, each node runs its own DNS server and responds to a DNS query for its DNS name into the corresponding IP address, which has been sent by another node. The naming service provided by Bonjour has the following security problem. Any node can generate its DNS name by itself and share it with other nodes through a multicast protocol. This open sharing allows a malicious node to pretend to be another node by using the DNS name of another node as its own DNS name and responding to a DNS query for the DNS name of another node with its own IP address. This bogus reply from the malicious node lets some node's traffic be directed to the malicious node. Also, since mDNS works based on multicast, it generates a large amount of traffic on the network. Therefore, mDNS is not suitable for naming services for IoT devices in large-scale IoT networks in terms of naming security and traffic efficiency.

In this paper, we propose a new naming framework for DNS Name Autoconfiguration (DNSNA) to support DNS naming services for IoT devices. The ultimate goal of the DNSNA is to provide DNS naming services to IoT devices so that they can be easily recognized by users. This enables the users to monitor or remote-control the IoT devices in their local network or across the Internet. DNSNA supports the DNS naming services (i.e., DNS name auto-generation, DNS name auto-registration, device discovery, and service discovery) of IoT devices without any user intervention or configuration in IP-based IoT environments. First, the DNS name auto-generation is to let IoT devices generate their appropriate DNS names with the

* Corresponding author.
E-mail addresses: keuntaelee@skku.edu (K. Lee), seokhwakim@skku.edu (S. Kim), pauljeong@skku.edu (J. Jeong), sejunlee@skku.edu (S. Lee), hyoung@skku.edu (H. Kim), pjs@etri.re.kr (J.-S. Park).

https://doi.org/10.1016/j.future.2018.01.023
0167-739X/© 2018 Elsevier B.V. All rights reserved.

advertised DNS domain and their device information. Second, the DNS name auto-registration is to let the IoT devices automatically register their DNS name and the corresponding IP address into a designated DNS server (called authoritative DNS server) having the authority for the advertised DNS domain names. Third, the device discovery is to allow a user to retrieve a list of the DNS names of IoT devices in order to discover what IoT devices are available in the user's residing network. Fourth, the service discovery is to allow a user to retrieve a list of the services provided by the IoT devices in the user's residing network. Note that the work in this paper is extended from our previous studies [4–7], where a basic architecture and authentication scheme were introduced.

DNSNA is designed to support efficient DNS naming services for IoT devices in IPv4 or IPv6 networks [4,6] through the unicast-based interaction between IoT devices and a local DNS server designated for IoT DNS naming services. DNSNA uses a unicast protocol for DNS name resolution instead of multicast, and consists of the standard protocols, such as Neighbor Discovery (ND) [8], Node Information (NI) Query [9], and Dynamic Host Configuration Protocol (DHCP) [10,11]. With the proposed method in DNSNA, we can easily register and control DNS names of IoT devices.

An important issue is to prevent an attacker's illegal IoT device from registering its DNS information into a DNS server in a user's network without the user's consent. The attacker might try to connect its unauthorized device to the user's network by exploiting the DNSNA protocol. To prevent an attack that an illegitimate DNS name registration is performed, we propose a secure DNSNA for IoT devices [5] in this paper. This secure DNSNA version authenticates IoT devices by using a physical-contact-based authentication based on near field communication (NFC) between an administrator's smartphone and an IoT device through the interaction with an authentication server. If a user wants to connect his (or her) IoT device to an IoT network, the IoT device should be authenticated through his (or her) smartphone that communicates with the authentication server having his (or her) credential information (i.e., a private key and public key as a signing key and verification key, respectively). The user can explicitly provide his (or her) credential information, which can be used to generate valid DNS name registration messages, using NFC in a secure manner.

Our key contributions are as follows:

- A DNS Naming Framework for IoT Devices: This paper proposes a scalable, efficient, secure DNS naming framework in a large-scale IoT network that is IPv4 or IPv6 (see Section 3.1).
- A DNS Name Format for IoT Devices: A new DNS name format is proposed, which is based on object identifier (OID) [12] having device information (e.g., vendor and model) and physical location (e.g., living room and kitchen) along with a logical DNS domain (e.g., .home and .skku.edu) (see Section 4.1).
- IoT DNS Naming Services: DNSNA supports the following DNS naming services of IoT devices: (i) IoT DNS name auto-generation, (ii) IoT DNS name auto-registration, (iii) IoT device discovery, and (iv) IoT service discovery (see Sections 4.2 and 4.6).
- A Physical-Contact-Based Authentication: A physical-contact-based authentication can authenticate a legitimate IoT device by giving it a signing key for the authentication of its DNS name registration message via NFC between an administrator's smartphone and an authentication server (see Section 4.3).

The remainder of this paper is structured as follows. Section 2 is the literature review of DNS naming services. Section 3 shows the problem formulation of DNSNA. Section 4 describes the design and protocol of DNSNA. Section 5 explains testbed configuration. Section 6 evaluates the performance of DNSNA. Section 7 concludes this paper along with future work.

2. Related work

Many DNS naming services have been studied intensively in industry and academia [3,4,12–14]. Considering the rapidly increasing number of IoT devices, the naming for IoT is getting more important for the management of IoT devices.

As a de facto standardization organization for IoT in the form of Machine-to-Machine (M2M) communication, the oneM2M provides machines (as IoT devices) with a software platform for industry-oriented services such as energy, transportation, smart home, and public safety. The oneM2M's platform allows IoT service developers to easily develop applications to control IoT devices in the Internet. The oneM2M standardized an object identifier (OID) as a globally unique identifier (ID) for an IoT device [12]. An OID as an M2M node's ID consists of two arcs. The first arc is a higher arc that is M2M node indication ID, which is a globally unique identifier for the M2M node. The second arc is a sequence of four lower arcs that are the manufacturer ID, product model ID, serial number ID, and expanded ID, respectively. Each arc is a placeholder for device identification and description. The higher arc is assigned and managed by International Telecommunication Union Telecommunication Standardization Sector (ITU-T) and International Organization for Standardization/International Electrotechnical Commission (ISO/IEC). However, this OID for an IoT device is deterministically constructed with the device information regardless of the IoT device's current location. On the other hand, since an IoT device's DNS name in our DNSNA can include a domain name (e.g., home and skku.edu) and a physical location (e.g., living room and kitchen at home), the IoT device's current location can be reflected, so it is easy to track the IoT device and control it. Our DNSNA minimizes a user's intervention in the DNS naming service. This is possible by allowing the IoT devices to generate their unique DNS names and register them into a designated DNS server in order to provide an IoT user with the DNS name list of the IoT devices.

The oneM2M is focused on standardizing a unified service platform for various kinds of IoT devices, so it accommodates various application layer protocols for control and management, such as Message Queue Telemetry Transport (MQTT) [15], Constrained Application Protocol (CoAP) [16], Hyper Text Transfer Protocol (HTTP) [17], and TCP/UDP socket. Our DNSNA can also accommodate CoAP, HTTP, and TCP/UDP socket.

CoAP [16] is an application-layer protocol to support Representational State Transfer (REST) or RESTful web service of constrained IoT devices operating with low-power and low-capacity hardware. CoAP is designed for networks with a low communication bandwidth and a high packet loss rate. The CoAP's RESTful-based approach allows an easy conversion and interoperability with the existing HTTP protocol. CoAP defines four message types, such as confirmable, non-confirmable, acknowledgment, and reset [16]. CoAP works on top of IEEE 802.15.4 as a MAC protocol, IPv6 as a network-layer protocol, and UDP as a transport-layer protocol. With IP and UDP, CoAP can support both unicast and multicast. IoT devices in oneM2M uses CoAP as a message delivery protocol. Our DNSNA also uses CoAP in order to control IoT devices [18].

Bonjour [3] is Apple's implementation of zero configuration networking to support DNS naming and service discovery. Bonjour can support the search and registration of Apple devices in order to announce the services of Apple devices. Bonjour is based on Multicast DNS (called mDNS) [13] which is used by every Apple device as both a DNS resolver and a DNS server. mDNS is used as an application protocol for device discovery and service discovery. mDNS can support DNS naming services without an authoritative DNS server in a network. DNS-based Service Discovery protocol (DNS-SD) [14] is used to perform service discovery by Bonjour software. DNS-SD can provide a list of service devices and port numbers to hosts by using DNS service resource records. mDNS

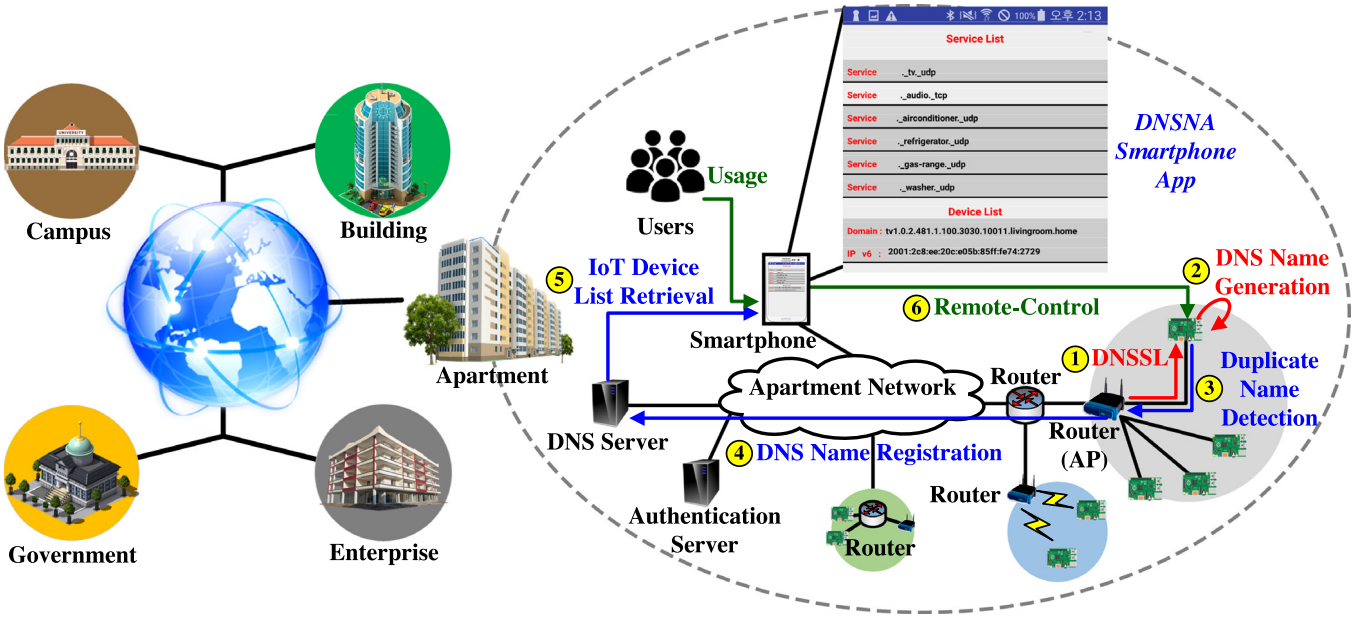


Fig. 1. DNSNA framework.

uses multicast for DNS name resolution in a local network, but this multicasting is not efficient for a large-scale network having multiple subnets because of the unignorable volume of generated traffic for mDNS. The SD feature of mDNS is good, but in the same reason due to multicasting, is not suitable for a DNS naming service in a large-scale network having many IoT devices in distributed multiple subnets. On the other hand, DNSNA is designed to support the efficient DNS naming services in a large-scale network with unicast-based DNS registration, DNS name resolution, and service discovery [4,18].

We have so far discussed DNS naming services for IoT devices. As a baseline protocol for the performance evaluation of our DNSNA in Section 6, we will use Multicast DNS (called mDNS) [13], which is the most popular protocol as a candidate for IoT DNS naming services.

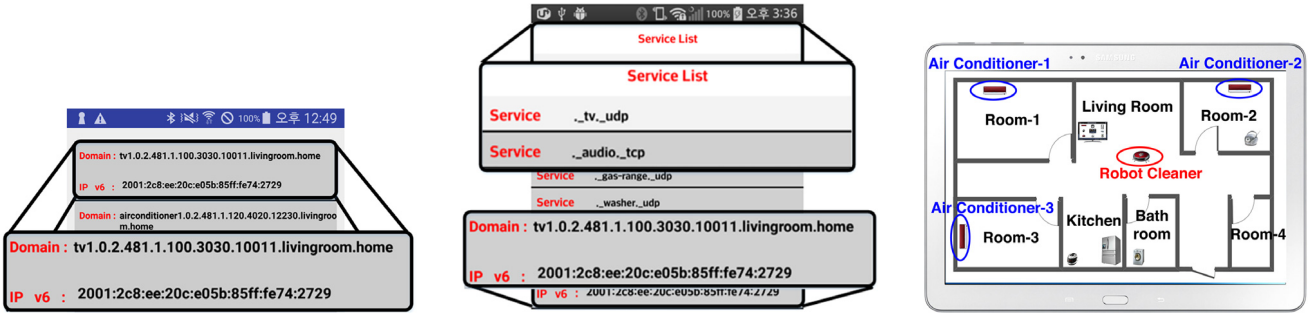
3. Problem formulation

In this section, we formulate the DNS naming service in various networks as follows. Given various networks with routers, our goal is to provide the efficient DNS naming services for IoT devices, while minimizing human intervention for configuration. Using our DNS naming services, such as device discovery and service discovery, we can monitor and remote-control such IoT devices easily.

3.1. DNSNA framework

We describe our DNSNA framework to support DNS naming services in various networks (e.g., campus network, enterprise network, and home network). Fig. 1 shows the DNSNA framework. The DNSNA framework consists of a DNS server, routers (including a DHCP server), an authentication server (AS), IoT devices, and mobile devices.

- DNS Server: A DNS server is a naming server that maintains DNS zone files for DNS names and IP addresses of IoT devices and other nodes. The DNS server has the DNS name list of IoT devices in the same network. The DNS server uses a static global IP address. The IoT devices using our DNSNA scheme can register its DNS name into a designated DNS server via a router or DHCP server. The DNS server can provide a valid user with a DNS name list of IoT devices.
- Router: A router is a gateway including a DHCP server to connect wired or wireless IoT devices to the wired network (e.g., the Internet). The router may be an AP to allow the Internet access of IoT devices. The router advertises an IPv6 prefix for a subnet by prefix option, a Recursive DNS Server (RDNSS) address by RDNSS option, and a DNS Search List (DNSSL) having DNS domains by DNSSL option, as shown in Fig. 1, which are IPv6 Router Advertisement (RA) options [8,19]. The router as a DHCP server can provide the same network information (i.e., prefix, RDNSS, and DNSSL) to IoT devices as DHCP clients. This network information is configured in the router by an administrator or another autoconfiguration protocol.
- Authentication Server (AS): An AS is a server to authenticate IoT devices via a user's smartphone. This server is used in our Secure DNSNA version called SDNSNA. The AS generates a key pair of a signing key and a verification key. The signing key is delivered to an IoT device via the user's smartphone in order to make a digital signature for a registration message for an IoT device's DNS name. The verification key is delivered to a router in order to verify whether the digital signature for an IoT device's registration message is valid or not.
- IoT Device: An IoT device is a target device that we want to manage. The spectrum of IoT devices is broad from low-power, low-capacity devices (e.g., fire sensor and temperature sensor) to high-power, high-capacity devices (e.g., smart TV and air conditioner, and washer). It has an NFC device to communicate with a smartphone and a Wi-Fi device to communicate with an AP.
- Mobile Device: A mobile device is a hand-held device, such as smartphone and tablet. It has an NFC device to communicate with an IoT device and a Wi-Fi device to communicate with an AP. The mobile device can display the DNS name list of IoT devices to monitor and remote-control IoT devices.



(a) Device discovery. (b) Service discovery. (c) Graphical display of devices.

Fig. 2. Device discovery and service discovery of DNSNA.

3.2. Assumptions

Our DNSNA has the following assumptions.

- The DNS server has a parsing program to translate from a zone file of DNS names to a DNS name list of IoT devices.
- The routers are configured with DNS information, such as RDNSS(es) and DNSSL along with IPv6 prefix.
- The AS has the authentication information for legal users. (e.g., an ID and a key pair of a signing key and a verification key per each user).
- The IoT devices have their manufacture information (e.g., vendor, device model, and serial number).
- The smartphone has a remote-control application and authentication application for IoT devices.
- The routers can play a role of a DHCP server for IoT devices as DHCP clients. Also, a separate node other than a router may be a DHCP server.
- For the authentication of IoT devices (e.g., SDNSNA), a secure channel (e.g., Transport Layer Security (TLS) or Secure Shell (SSH)) is established between an IoT device and an AS and another secure channel between a router and the AS.

3.3. Concept of DNSNA

DNSNA is an efficient DNS naming system that is designed for a large-scale network having IoT devices in multiple subnets. Fig. 1 shows the DNSNA framework. Before the start of DNSNA operations, the key distribution for secure DNS autoconfiguration is performed as follows. First, an authentication server gives an administrator's smartphone a signing key for generating a digital signature of its DNS name registration message. Second, it gives a router having the IoT device a verification key for verifying the digital signature of the message sent by the IoT device to the router. Refer to Section 4.3 for the detailed procedure of the naming security. The procedure of DNS naming services consists of the following steps.

1. A router (or DHCP server) provides a DNSSL to an IoT device.
2. The IoT device generates its DNS name with the advertised DNSSL.
3. The IoT device verifies the uniqueness of its DNS name through a Duplicate Name Detection (DND) procedure. This uniqueness verification is subnet-wide DND.
4. The router collects the DNS names of IoT devices and registers them into a DNS server through the verification of the uniqueness of the DNS names. This uniqueness verification is network-wide DND.
5. A smartphone can retrieve the DNS name list of IoT devices from the DNS server having the DNS names and the corresponding IP addresses of the IoT devices. It can perform device discovery of available IoT devices (as shown in Fig. 2(a)) and service discovery (as shown in Fig. 2(b)) of available services. The device discovery can use a localization scheme (e.g., SALA [20]) to recognize the locations of the IoT devices and then display the IoT devices in their positions in a target place (e.g., smart home), as shown in Fig. 2(c).
6. The smartphone can remote-control the IoT devices with the DNS name list and the corresponding IP addresses.

So far, we have explained the DNSNA framework, assumptions, and concept of DNSNA. In the next section, we will explain the design and protocol of our DNSNA.

4. Design and protocol of DNSNA

In this section, we describe the design and protocol of DNSNA. As mentioned before, for a large number of IoT devices, it is inefficient to manually configure their DNS names in a target network. Thus, an automatic DNS name configuration (i.e., DNS name autoconfiguration) is required. The previously announced DNS naming services are not suitable for the IoT devices. We researched on how to provide the DNS naming services for the IoT devices using standard protocols without significantly changing them. Our DNSNA supports DNS naming services in both IPv4 and IPv6 networks along with the authentication for IoT devices.

DNSNA can be divided into three versions largely. The first version is to support the DNS naming services for the device discovery of both IPv4 and IPv6 IoT devices. The second version is to support the authentication of IoT devices through the physical-contact-based NFC. The third version is to provide the service discovery based on DNS service resource records. First of all, we will explain the DNS name format for IoT devices in our DNSNA framework.

4.1. DNS name format for IoT devices

The DNS names of IoT devices can have the following format with OID [12] and location information, which is defined in our Internet Draft [7], as in Fig. 3. This format can contain the physical location of an IoT device. Thus, if the DNS name of an IoT device contains location information, it allows a user to easily identify the physical location of each device. In [7], the location in a DNS name consists of two levels for a detailed location specification, such as a macro-location for the rough position and a micro-location for the detailed position in the macro-location. Note that a hierarchical location specification with more than two levels is left as future work in a DNS name format. To denote both the macro-location (i.e., mac_loc) and micro-location (i.e., mic_loc) into a DNS name, the following format is described as in Fig. 3.

unique_id.object_identifier.OID.mic_loc.mac_loc.LOC.domain_name

Fig. 3. IoT device DNS name format with OID.

- **unique_id**: unique identifier to guarantee the uniqueness of the DNS name in ASCII characters. The identifier may be alphanumeric with readability, e.g., device type (e.g., smart TV, air conditioner, washer, and fire sensor) concatenated by a sequence number.
- **object_identifier**: device's object identifier that consists of a higher arc, that is, M2M node indication ID (i.e., the concatenation of the managing organization, administration, data country code, and M2M node) and a sequence of four arcs (i.e., manufacturer ID, model ID, serial ID, and expanded ID) as defined in [12]. The fields are separated by an underscore '_'.
- **OID**: subdomain for the keyword of OID to indicate that object_identifier is used.
- **mic_loc**: device's micro-location, such as center, edge, and corner.
- **mac_loc**: device's macro-location, such as living room and kitchen.
- **LOC**: subdomain for the keyword of LOC to indicate that mac_loc and mic_loc are used.
- **domain_name**: domain name that represents a DNS domain for the network having the IoT devices.

4.2. DNS name autoconfiguration (DNSNA)

We propose a scheme to support DNS name services using standard protocols and designed for IoT devices in both IPv4 and IPv6 networks.

4.2.1. DNSNA in IPv6 networks

We explain DNSNA for IoT devices in IPv6 networks [4,7]. Fig. 4 shows a system configuration of DNSNA. First, in Fig. 4, either a router or a DHCPv6 server sends IoT devices (i.e., IPv6 hosts) RA DNS options or DHCPv6 DNS options, respectively, which include RDNSS addresses and a DNSSL, to an IPv6 host (i.e., IoT device) [11,19].

Second, once it obtains a DNSSL, the IoT device autonomously generates its own DNS name using the DNSSL and its device information according to the DNS name format in Section 4.1 [4,7].

Third, after generating a DNS name, the IoT device constructs an IPv6 address by hashing the generated DNS name and attaching the 64-bit hashed value to the advertised IPv6 subnet prefix, which is advertised by an RA prefix option [8]. For the subnet-wide duplicate name detection (called DND) of the generated DNS name, it then performs the Duplicated Address Detection (DAD) for the constructed IPv6 address corresponding to the generated DNS name in order to verify the uniqueness of the DNS name [7]. If the IoT device does not receive any response to the DAD message from any other IoT device, the IoT device regards the DNS name as a unique DNS name in its own subnet.

Fourth, the router sends an NI query message [9] to collect a pair of a DNS name and the corresponding IPv6 address of each IoT device [7]. Each IoT device sends the router an NI reply message to register their own DNS name into a designated DNS server for the DNS domain advertised by the DNSSL.

Fifth, for the network-wide DND, after receiving the NI reply messages from IoT devices, the router checks whether the collected DNS names of the IoT devices have already been registered in the DNS server by sending the DNS query messages to the DNS server. If the DNS reply messages do not have the corresponding IP addresses, the queried DNS names are not registered into the DNS server yet. If so, the router registers them into the DNS server by the DNS dynamic update protocol [21]. Otherwise, the router needs to notifies each IoT device with a duplicate DNS name of such a DNS name duplication so that each IoT device can generate another DNS name and try to register its DNS name again via the router. Refer to Section 4.4 for the detailed discussion of DNS name conflict resolution.

Sixth, after this DNS name registration, a user can get a DNS name list of available IoT devices from the DNS server, and then the user can monitor or remote-control the IoT devices with the list.

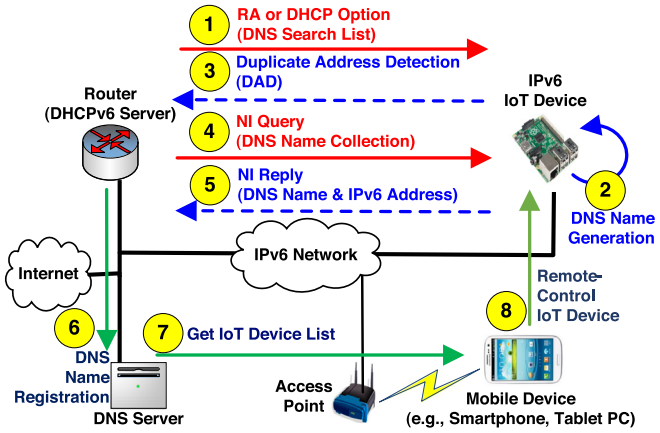


Fig. 4. System configuration for DNSNA in IPv6 networks.

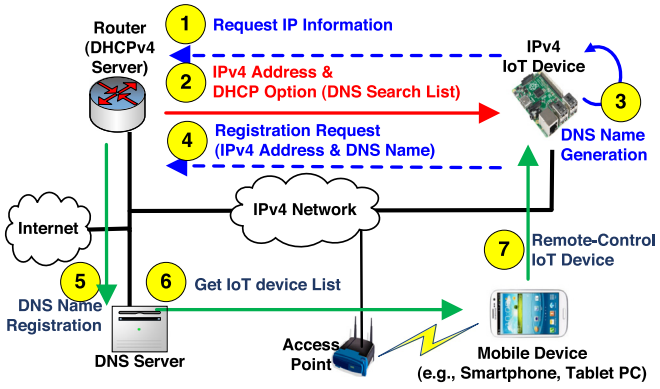


Fig. 5. System configuration for DNSNA in IPv4 networks.

4.2.2. DNSNA in IPv4 networks

We explain DNSNA for IoT devices in IPv4 networks (called DNSNAv4) [6], which is an IPv4 version of DNSNA [4]. Fig. 5 shows a system configuration for DNSNAv4. First, an IoT device in an IPv4 network requests IP information to a DHCPv4 server when it is connected to the network. The DHCPv4 server allocates an IPv4 address and sends the allocated IPv4 address to the IoT device along with RDNSS addresses and a DNSSL through the DNS options [10]. The IoT device then generates its DNS name when the IoT device receives the reply message from the DHCPv4 server. After the IoT device generates its DNS name, it sends the DHCPv4 server a request message to register its DNS name into a DNS server. The DHCPv4 server checks if the DNS name is unique or not, when the DHCPv4 server receives the request message for the DNS name registration. If the message is valid and the DNS name does not

exist in its DNS name cache, the DHCPv4 server registers the pair of the DNS name and the corresponding IPv4 address into a designated DNS server for the DNS domain advertised by the DNSSL [21]. Finally, when the DNS configuration for all the IoT devices is done, a user can get a DNS name list of IoT devices and remote-control the IoT devices through a mobile device (e.g., smartphone and tablet).

In the case where the message is valid, but the DNS name has already been used by another IoT device, which means the duplication of the DNS name, during either the subnet-wide DND or the network-wide DND, DHCP does not have a resolution scheme of such a DNS name conflict [22–24]. Therefore, we need a new DHCP option for the resolution of such DNS name conflicts with a minimal user intervention. Refer to Section 4.4 for the DNS name conflict resolution.

4.3. Secure DNSNA

In this subsection, we describe a Secure DNSNA called SDNSNA [5]. If even a malicious IoT device is connected to a user's network through an AP, the pair of a DNS name and the corresponding IPv6 address of such a device can be registered into a DNS server through the DNSNA protocol [4]. This is because DNSNA allows the registration of any IoT device's DNS name without any authentication. If the malicious IoT device registers its DNS name into a DNS server through an AP or an Ethernet, a hacker can get the various information of IoT devices and their users through the AP. For this security threat, SDNSNA is designed to authenticate IoT devices through the interaction between a valid user's smartphone and an authentication server. SDNSNA authenticates the IoT device with the user's physical authentication via an NFC device [5], thereby providing a defense against an illegal DNSNA registration such that only authorized devices can be registered.

4.3.1. SDNSNA in IPv6 networks

We explain the main idea and framework of SDNSNA for IPv6 IoT networks. Fig. 6 shows the system configuration of SDNSNA. SDNSNA has an additional authentication step for DNSNA in Section 4.2.1.

First, an administrator's smartphone requests authentication information to the AS. The AS then generates a key pair of a signing key and a verification key, and sends the signing key and the verification key to the smartphone and the router, respectively. After the router receives the verification key from the AS, the router will use the verification key to verify the registration message of an IoT device's DNS name. In the meanwhile, the smartphone sends the signing key acquired from the AS to the IoT device via NFC.

Second, after receiving the signing key, the IoT device will be able to make a digital signature for an NI reply to register its DNS name. The router periodically broadcasts an NI query to the IoT devices in its subnet(s) in order to collect the DNS information from them and register them into the DNS server. When the IoT device receives such an NI query, the IoT device constructs an NI reply for its DNS name registration along with a nonce and a timestamp, which are used to prevent a replay attack. It then generates a digital signature for the NI reply, sending the NI reply with the digital signature to the router.

Third, the router receives the NI reply from the IoT device and verifies the reply message with the corresponding verification key. If it successfully verifies the digital signature of the reply message with its verification key, the router makes a DNS name registration request to the DNS server through both the subnet-wide DND and the network-wide DND. After the DNS registration is done, a user can get an IoT device list from the DNS server and remote-control the IoT devices.

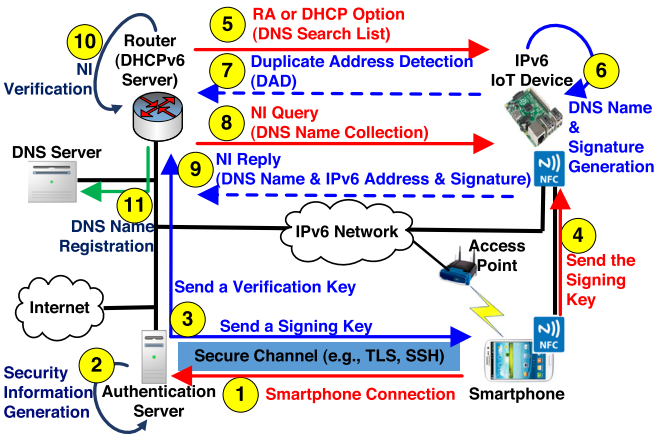


Fig. 6. System configuration for SDNSNA in IPv6 networks.

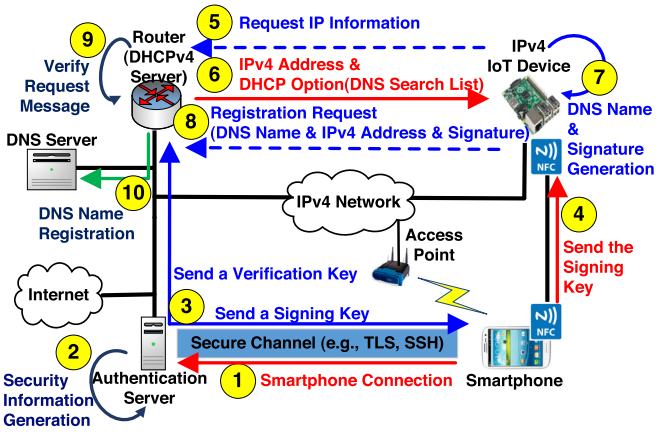


Fig. 7. System configuration for SDNSNA in IPv4 networks.

4.3.2. SDNSNA in IPv4 networks

We explain the main idea and framework of SDNSNAv4 for IPv4 IoT networks. Fig. 7 shows a system configuration for SDNSNAv4. In SDNSNAv4, when the IoT device generates its own DNS name and a digital signature for the registration of its DNS name, it immediately requests the DHCP server to register its own DNS name into the DNS server.

First, in the same way with SDNSNA in Section 4.3.1, the IoT device and the DHCP server can get a signing key and a verification key, respectively, from the AS. The IoT device constructs a DHCP request for the DNS name registration request and generates a digital signature of the DHCP request with its signing key, sending the DHCP server the DHCP request. The DHCP server verifies the DHCP request with the verification key.

Second, through both the subnet-wide DND and the network-wide DND, the DHCP server can register the IoT device's DNS name into the DNS server. A user can retrieve a DNS name list of IoT devices available in the current network for remote-controlling the IoT devices.

4.4. DNS name conflict resolution

Now we explain DNS name conflict resolution in two levels, such as subnet-wide DND and network-wide DND. For the subnet-wide DND for IPv6 IoT networks, both DNSNA and SDNSNA require a new NI type to notify an IoT device of its duplicate DNS name in the subnet. When the IoT device receives such a notification, it

needs to generate a new DNS name with another unique id in the DNS name format, as described in Fig. 3.

For the subnet-wide DND for IPv4 IoT networks, both DNSNAv4 and SDNSNAv4 require a new DHCP option to notify an IoT device of its duplicate DNS name in the subnet. When the IoT device receives such a DHCP option, a new DNS name needs to be generated with another unique id in the DNS name format, as described in Fig. 3.

For the network-wide DND for IPv6 IoT networks, both DNSNA and SDNSNA require a new NI type to notify an IoT device of its duplicate DNS name in the network. When the IoT device receives such a notification, it needs to generate a new DNS name with another unique id in the DNS name format, as described in Fig. 3.

For the network-wide DND for IPv4 IoT networks, both DNSNAv4 and SDNSNAv4 require a new DHCP option to notify an IoT device of its duplicate DNS name in the network. When the IoT device receives such a DHCP option, a new DNS name is generated with another unique id in the DNS name format, as described in Fig. 3.

4.5. Discussion on security issues

In this section, we introduce how to solve security issues of the DNS name autoconfiguration in large-scale networks using SDNSNA. The large-scale networks include smart building, smart factory, and smart transportation, and smart campus. We can classify IoT devices into high-spec and low-spec devices. Therefore, this section discusses scenarios that need to be considered according to device specifications in large-scale networks.

To solve the security issues of the name autoconfiguration in large-scale networks, we should cooperate with companies that produce small IoT devices such as sensors. The reason we need to cooperate with a factory is that we cannot intervene in its production process. Thus, when small IoT devices are produced, an authentication key for DNS autoconfiguration can be included into them in the production process as a factory key management service [25]. The authentication key can be a software key such as a private key as a signing key in asymmetric cryptography. When using such a software key, an administrator receives the key for managing the small IoT devices from the factory, and then (s)he can generate a digital signature of a DNS name registration message with the key as a signing key. Therefore, the administrator can easily register the DNS names of the small IoT devices into its local DNS server through the digital-signature-based authentication via a router having a public key as a verification key.

On the other hand, big IoT devices, such as smart TV, robot cleaner, and air conditioner, can be authenticated through NFC-based physical contact. For these big IoT devices, this approach does not affect their product price because an NFC tag is relatively cheap around USD 24 [26]. As a result, for our Secure DNSNA, we chose NFC tags for a cost-effective secure authentication of big IoT devices. Though ZigBee and Bluetooth are available for wireless communication for IoT devices, their communication ranges are longer than NFC. Note that Bluetooth has a communication range under 10 m and ZigBee has a communication range under 100 m. This longer communication range allows a hacker to eavesdrop the authentication process between an IoT user's smartphone and an IoT device, so, an authentication key of such a valid IoT device can be obtained by the hacker. On the other hand, since NFC tags have a communication range under 20 cm, the eavesdropping of the authentication key can be avoided with a high probability. Since this paper shows NFC as a feasible solution for Secure DNS naming, authentication with other wireless technologies (e.g., ZigBee, Bluetooth, and WiFi) is left as future work.

```
smartTV._tv._tcp.local. 120 IN SRV 0 0 515 smarttv.local.
tabletPC._tv._tcp.local. 120 IN SRV 0 0 615 tabletpc.local.
```

Fig. 8. DNS service resource record format.

4.6. Service discovery

If an IoT user is looking for a printer server on a local network, it is difficult for him (or her) to find it by its host name alone. However, if DNS-based Service Discovery (DNS-SD) [27] is used, service search is possible by specifying a service type. A DNS service (SRV) resource record (RR) can be used to support the Service Discovery (SD) of services provided by an IoT device [28]. This SRV RR specifies a service instance name, consisting of an instance name (i.e., device), a service name, a transport layer protocol, a domain name (i.e., DNS suffix), the corresponding port number, and the DNS name of the device to provide the service. In the DNS-SD, a user can specify a service instance with the SRV RR [27] and discover a list of IoT devices for a specified service type with a DNS pointer record (PTR) [2,27]. Note that PTR is originally used for reverse DNS lookup [2].

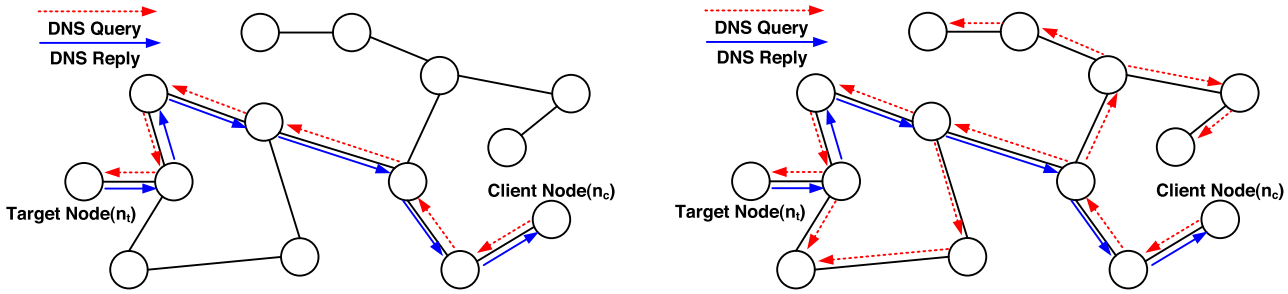
An SRV RR is used to provide detailed service information (e.g., instance name, service name, domain name, and port number) to a user that wants to use a service provided by an IoT device. An important feature of the SRV RR is that you can use "instance name" as a key. For example, even if either the IoT device's IP address or its DNS name changes, the service instance associated with the IoT device remains the same. Thus, you can still find the IoT device corresponding to the service instance.

The format of the SRV RR is described in Fig. 8 as follows. Each field can be separated by a space character. The first field in the SRV RR, which is instance name, has the format of <Instance> . <Service> . <Domain>. For example, as shown in Fig. 8, the SRV RR represents a device that provides a TV service in the local network. As the second field, 120 is used to set the cache period by Time-to-Live (TTL). The third field indicates a class [2], such as the Internet class (i.e., IN). The fourth field of SRV indicates that this RR is an SRV RR. As the fifth and sixth fields, two zeros are weight and priority, respectively. As the seventh field, 515 is used to set the port number. The last field is the DNS name of the device corresponding to the service instance. This DNS name can be resolved into an IP address of the device.

We can find several devices of the same service name if they can support the same service type. For example, a TV service may have more than one service instances. As shown in Fig. 8, "_tv._tcp" indicates a TV service and two TV service instances exist, such as "smartTV_tv._tcp" and "tablePC_tv._tcp". Since there are two IoT devices in the house that are a smart TV and a tablet PC, respectively, they can be found by the service discovery with "_tv._tcp".

The PTR record has only a service name and a domain name in the format of <Service> . <Domain> without any instance name [27]. This PTR record is used to get a list of IoT devices for a specified service type.

After registering the services of IoT devices as SRV RRs through the same procedure with the DNS name registration in Section 4.2, the user can retrieve the available service instances in the target network through a DNS server with SRV RRs. The SRV RR list can be used to monitor or remote-control IoT devices or services according to the server's protocol and domain. Once a desired service is found from the list, it can be used by a user. Thus, even though the IP address of a service device changes, the SRV RR does not change. This is because the service instance name for a device running the service is the same in the DNS server, and the change



(a) DNSNA. (b) mDNS.

Fig. 9. Two naming approaches of DNS name resolution: DNSNA and mDNS.

of either the IP address or the DNS name is reflected in the DNS server. This change is reflected in the A or AAAA resource record for the IPv4 or IPv6 service device, respectively [2,29].

Therefore, the SRV format of DNSNA conforms to the format in the DNS-SD [27] and has the same format in Fig. 8. After the service registration of IoT devices, a user can retrieve the list of SRV RRs, and then the user can monitor or remote-control the IoT devices or their services by the protocol and domain of the servers providing the services.

So far, we have explained the DNS name format, the framework, and service discovery for DNSNA. In the next section, we will explain the experiment of DNSNA in our testbed.

5. Testbed configuration

In this section, we explain our testbed configuration, such as experiment environments and hardware configuration. In DNSNA, IoT devices can be set up through desktop computers, laptop computers, and smart devices (e.g., smartphone and tablet). In our testbed, we used a smartphone as a control device. The communication between a user's smartphone and an IoT device was implemented with UDP-over-IP socket. We also implemented a CoAP-based communication to control IoT devices in our early work for the service discovery and remote-control of IoT devices [18].

Experiment Environments: To verify the effectiveness of DNSNA, we conduct experiments in two different environments. Tests are performed in these two environments. The first environment is an IPv6 network, and the second environment is an IPv4 network. Note that for secure IoT DNS services, SDNSNA and SDNSNAv4 were implemented and tested according to the procedure in Section 4.3.1 and Section 4.3.2, respectively. Refer to our early work of SDNSNA for the detailed design, implementation, and test of SDNSNA [5]. DNSNA and SDNSNA were tested in the IPv6 environment. DNSNAv4 and SDNSNAv4 were tested in the IPv4 environment. In each environment, test-purposed IoT devices were located in the same subnet. Note that the implementation of DNS name conflict resolution in Section 4.4 is left as future work because this paper is focused on the functionality and performance of DNSNA. For the simplified version of DNSNAv4, a proprietary protocol was designed and implemented rather than the extension of DHCPv4 for the DNS name registration with a new DHCP option. The extension of the DHCP for DNSNA is left as future work.

Hardware Configuration: In each environment, we used a Raspberry Pi as an IoT device in our DNSNA testbed, a desktop PC as an authoritative DNS server, a desktop PC as an AS, and a desktop PC as a router (also used as a DHCP server), respectively. As previously mentioned in Section 3, the router and the DHCP server are running in the same equipment. A desktop is modified to act as a router with the "radvd" tool in [30] and as a DHCP server with the "isc-dhcp-server" tool in [31]. Another desktop acts as an AS, which was designed and implemented by us, and another desktop acts as a DNS server with the "bind9" tool in [32]. Those desktops have 3.4 GHz CPU (Intel i7-3770) and 16 GB memory. Raspberry Pi nodes, which have 900 MHz 32-bit quad-core ARM Cortex-A7 and 1 GB memory, are used as IoT devices both in IPv4 and IPv6 environments.

6. Performance evaluation

We compare DNSNA with mDNS [13] in real environments. In performance evaluation in this paper, we use a basic version of DNSNA for fair evaluation because mDNS does not provide security. We show the overall performance comparisons in two different cases, i.e., the number of packets and accumulated packet volume over time.

6.1. Message usage for DNS name resolution

As described in [4], let $G = (V, E)$ be a target network graph where V is a vertex set of vertices (i.e., routers and hosts) and E is an edge set of edges (i.e., wired or wireless links). Fig. 9 shows the target network graph of an IoT network. The graph diameter (d_G) from a client node (n_c) to a designated node (n_t) with the maximum hops is as follows:

$$d_G = d(n_c, n_t), \tag{1}$$

where $d(n_c, n_t)$ is the number of hops between n_c and n_t. The graph diameter of d_G is the maximum number of hops between an arbitrary pair of vertices in the graph G, as shown in Fig. 9(a). In DNSNA, the graph diameter of d_G is assumed to be m hops from a client node (as an IoT device) to a DNS server. The worst-case in DNSNA scheme has the message number of $2 \times m$ because DNSNA uses unicast for DNS queries and replies. Thus, the overhead for DNS name resolution can be represented $O(m)$.

On the other hand, in the worst case, mDNS uses multicast for DNS queries to the whole network with $|E|$ links. For a resolution, mDNS generates queries corresponding to the total number of links ($|E|$) in the target network because it uses multicast. Thus, mDNS generates m reply messages from the target node to the client node. Thus, the overhead for DNS name resolution can be represented $O(|E|)$.

6.2. Packet consumption for DNS name resolution

We have evaluated DNSNA with mDNS using Avahi open source [33]. Actually, in an IoT environment, IoT devices will still turn on, so there are so many packets in the network. We performed an experiment during one day and captured about approximately 2.5 million packets in one IoT device of our testbed

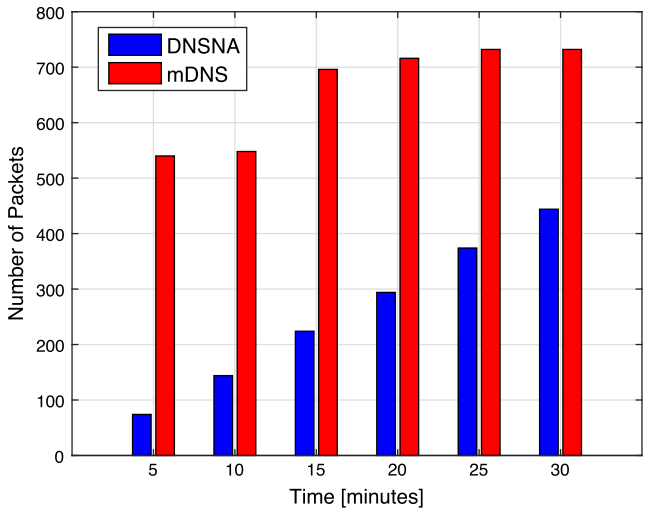


Fig. 10. Comparison between DNSNA and mDNS for the number of packets over time.

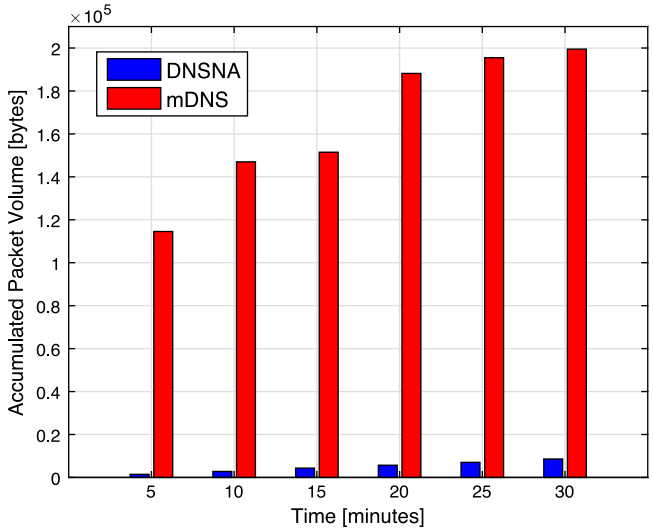


Fig. 11. Comparison between DNSNA and mDNS for accumulated packet volume over time.

environment. The number of packets captured by an IoT device is huge. Thus, we draw a result for 30 min about three IoT devices.

The test configuration is as follows. The testbed network is a campus subnetwork at Sungkyunkwan University. We measured the packet volume of each device for 30 min at 5-minute intervals and summed up the total packet volume. IoT devices were tested three times and all packets were captured by Wireshark [34]. The packet capture criteria are as follows. mDNS packets were related to mDNS protocol, and packets for IoT devices were identified by their IP addresses used in the experiment. Among packets using mDNS protocol, we filtered out a packet if it is not relevant to a pre-configured DNS name registration record for this experiment. DNSNA packets were relevant to DNS name resolution. In the same way with mDNS, we filtered out a packet if it does not have an address for this experiment. In fact, DNSNA was designed based on ICMPv6, but it should not be measured for other ICMPv6-based protocols (e.g., RA and ND).

Table 1
Comparison of two naming approaches.

Approaches	DNSNA	mDNS
Packet forwarding	Unicasting	Multicasting
Packet size	About 100 bytes	About 1500 bytes
Naming scope	Global, Local	Local
Code size	Hundreds	Thousands lines
Authoritative DNS server	Used	Not used
Working layer	Network layer	Application layer
Energy consumption for resolution	$O(m)$	$O(\|E\|)$

Fig. 10 shows the comparison between DNSNA and mDNS for the number of packets over time in our testbed network. As shown in this figure, DNSNA generates a less packet number than mDNS for all the measurement time instants. DNSNA can reduce the average number of packets of mDNS over time by 60.8%. Fig. 11 shows the comparison between DNSNA and mDNS for accumulated packet volume over time in the same network. As shown in this figure, DNSNA generates a smaller accumulated packet volume (in bytes) than mDNS for all the measurement time instants. DNSNA can reduce the accumulated packet volume of mDNS over time by 97%.

mDNS generates more packets for IoT DNS naming services than DNSNA because mDNS lets all IoT devices send their DNS information (including services) to their subnets in multicast. On the other hand, DNSNA lets all IoT devices send their DNS queries for device discovery and service discovery to their designated DNS server in unicast. As shown in Fig. 9(b), if there are many IoT devices in networks, mDNS will generate absolutely a lot of packets due to the multicast property. On the other hand, as shown in Fig. 9(a), since DNSNA uses unicast for DNS name resolution, DNSNA generates as many packets as the number of DNS queries for DNS name resolution. Thus, it is expected that this performance difference will be bigger in a larger network than in our experiment network.

In an IoT environment, IoT devices (e.g., robot cleaner) can move to different subnets at any time. In this case, DNSNA will generate packets for the IoT devices existing in the subnet (i.e., NI Reply and NI Query). On the other hand, mDNS sends a multicast message to other devices every time the information of a service or address is changed or added. Of course, most of IoT devices do not often move to other subnets. However, in mDNS, if the number of moving devices will increase, the number of packets relevant to DNS naming services will also increase very rapidly. Many of these DNS packets put unignorable burdens and energy consumption on low-power and low-capacity IoT devices.

Finally, Table 1 shows the comparison of DNSNA and mDNS. Especially, in Table 1, the packet size can be different by cases, but an IoT device running mDNS sends a packet including its known service information to its neighboring IoT devices located in its subnet in multicast. Therefore, as there are more services in the subnet, the packet size of the advertised DNS information will increase further.

6.3. Analysis and discussion

In the real world, IoT devices such as sensors will be turn on all day long. On the average, there were measured 2.5 million packets per IoT device. Among the measured packets, we filtered out irrelevant packets as described in Section 6.2. Results were different for each experiment. This was an all-day measurement for the one service. We also measured DNSNA in the same way by two cases. Because, for DNSNA, we can decide a time interval (e.g., 30 s and 5 min) for the DNS name collection of IoT devices. In the first case where the time interval is 30 s, DNSNA had about 1800

packets. In the second case where the time interval is 5 min, DNSNA had about 288 packets.

As shown in Fig. 10, mDNS shows that the packet volume did not increase gradually and the volume increase was not very large when it was tested all day long. We investigated why the volume increase was too small. mDNS is based on multicast protocol. Since mDNS sends packets periodically to networks, it generates a big overhead to networks when the time interval is 5 min in Fig. 10. Especially, mDNS generates many packets when a service has just been registered. After some time later, we can confirm that the volume of mDNS packets increased a little gradually in the experiment.

For mDNS, we experimented a new scenario about the DNS name resolution with three IoT devices. The experimental setup is as follows. Let A, B, and C be three IoT devices. For each device A, B, and C, we registered an experimental service with a specific service name. It was observed that each device detected service records of other devices using mDNS protocol. Next, we plugged out the Ethernet cable of Device A, and let the other devices update their service list. We observed whether or not the service information of Device A is deleted from the DNS zone files (i.e., DNS databases) of Devices B and C. However, Devices B and C could not update the service of the removed Device A. In addition, we performed another experiment. We changed the mDNS service record of Device A, which was separated from the Ethernet connection, and then was reconnected to the Ethernet network to join the mDNS group. As a result, it was observed that the removed service of Device A still remained on Devices B and C, and the new service of Device A was newly registered into Devices B and C. For example, assuming that a service called "clean" is registered in Device A, if Device A is intentionally disconnected from the network, and the service name is changed to "wash", then the staleness problem of DNS information mentioned just before in this second experiment. Through the experimental results, it can be concluded that mDNS cannot manage the services of IoT devices efficiently and correctly.

On the other hand, DNSNA checks the aliveness of a device's service with a periodic message polling. Therefore, DNSNA can be considered to be a better protocol in the IoT environments than mDNS, even though the time interval of NI Query of DNSNA, which is used for IoT device discovery and service discovery, is getting smaller and the packet volume is getting larger.

7. Conclusion

In this paper, we propose a framework of DNS Name Autoconfiguration (called DNSNA) for IoT devices in both IPv6 and IPv4 IoT networks. The number of IoT devices is expected to increase in the future. In the management of such a large number of IoT devices, it is inefficient for a user to manually configure their DNS names. In this paper, we describe various versions of DNSNA. Through comparison, it is shown that our DNSNA is more suitable for a large-scale IoT network than the legacy DNS naming scheme called mDNS. As future work, we will develop the security support of DNSNA for various wireless media, such as ZigBee, Bluetooth, and Wi-Fi, other than NFC. Also, we will develop the graphical display of IoT devices at a user's place (e.g., home and office) with a localization scheme (e.g., SALA [20]). In addition, we will make a cloud-based DNS naming services such that an IoT user's smartphone in the Internet can monitor and remote-control his (or her) IoT devices in his (or her) relevant places at any time and at any place.

Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2017M3C4A7065980).

References

[1] Gartner, Inc. Gartner says the Internet of Things installed base will grow to 26 billion units by 2020, http://www.gartner.com/newsroom/id/2636073. Accessed Dec. 13, 2017.
[2] P. Mockapetris, Domain names - implementation and specification, IETF RFC 1035, Nov. 1987.
[3] Bonjour, https://developer.apple.com/bonjour/. Accessed Dec. 13, 2017.
[4] S. Lee, J. Jeong, J. Park, DNSNA: DNS name autoconfiguration for Internet of Things devices, in: Proceedings of the 18th International Conference on Advanced Communication Technology, ICACT, Feb. 2016, pp. 410–416.
[5] K. Lee, H. Kang, J. Jeong, H. Kim, J. Park, Secure DNS name autoconfiguration for IPv6 Internet-of-Things devices, in: Proceedings of the 7th International Conference on ICT Convergence, ICTC, Oct. 2016, pp. 564–569.
[6] K. Lee, S. Kim, J. Jeong, DNSNAv4: DNS name autoconfiguration for Internet-of-Things devices in IPv4 networks, in: Proceedings of the 31st International Conference on Advanced Information Networking and Applications Workshops, Mar. 2017, pp. 347–351.
[7] J. Jeong, S. Lee, J. Park, DNS name autoconfiguration for internet of things devices, IETF Internet-Draft, draft-jeong-ipwave-iot-dns-autoconf, Oct. 2017.
[8] T. Narten, E. Nordmark, W. Simpson, H. Soliman, Neighbor discovery for IP version 6 (IPv6), IETF RFC 4861, Sep. 2007.
[9] M. Crawford, B. Haberman, IPv6 Node information queries, IETF RFC 4620, Aug. 2006.
[10] R. Droms, Dynamic host configuration protocol, IETF RFC 2132, Mar. 1997.
[11] R. Droms, J. Bound, B. Volz, C. Perkins, M. Carney, Dynamic host configuration protocol for IPv6 (DHCPv6), IETF RFC 3315, Jun. 2003.
[12] J. Kim, Y. Park, S. Choi, J. Song, Adding M2M device identifier, oneM2M, ftp://ftp.onem2m.org/Meetings/ARC/2014. Accessed Dec. 13, 2017.
[13] S. Cheshire, M. Krochmal, Multicast DNS, IETF RFC 6762, Feb. 2013.
[14] S. Cheshire, M. Krochmal, DNS-based service discovery, IETF RFC 6763, Feb. 2013.
[15] MQTT, http://mqtt.org/. Accessed Dec. 13, 2017.
[16] Z. Shelby, K. Hartke, C. Bormann, The constrained application protocol (CoAP), IETF RFC 7252, Jun. 2014.
[17] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext transfer protocol –HTTP/1.1, IETF RFC 2616, Jun. 1999.
[18] S. Kim, K. Lee, J. Jeong, DNS naming services for service discovery and remote control for Internt-of-Things devices, in: Proceedings of the 8th International Conference on ICT Convergence, ICTC, Oct. 2017, pp. 1157–1162.
[19] J. Jeong, S. Park, L. Beloeil, S. Madanapalli, IPv6 router advertisement options for DNS configuration, IETF RFC 8106, Mar. 2017.
[20] J. Jeong, S. Yeon, T. Kim, H. Lee, S. Kim, S. Kim, SALA: Smartphone-assisted localization algorithm for positioning indoor iot devices, in: Springer Wirel. Netw., Jan. 2018.
[21] P. Vixie, S. Thomson, Y. Rekhter, J. Bound, Dynamic updates in the domain name system (DNS UPDATE), IETF RFC 2136, Apr. 1997.
[22] M. Stapp, T. Lemon, A. Gustafsson, A DNS resource record (RR) for encoding dynamic host configuration protocol (DHCP) information (DHCID RR), IETF RFC 4701, Oct. 2006.
[23] M. Stapp, B. Volz, Y. Rekhter, The dynamic host configuration protocol (DHCP) client fully qualified domain name (FQDN) option, IETF RFC 4702, Oct. 2006.
[24] M. Stapp, B. Volz, Resolution of fully qualified domain name (FQDN) conflicts among dynamic host configuration protocol (DHCP) clients, IETF RFC 4703, Oct. 2006.
[25] Maxim Integrated's Factory Key Management Service,https://www.maximintegrated.com/en/products/digital/one-wire/factory-key-management-service.html.Accessed Dec. 17, 2017.
[26] Vulcan RFID Custom NFC Tag, https://www.atlasrfidstore.com/vulcan-rfid-custom-nfc-tag/. Accessed Dec. 17, 2017.
[27] S. Cheshire, M. Krochmal, DNS-based service discovery, IETF RFC 6763, Feb. 2013.
[28] A. Gulbrandsen, P. Vixie, L. Esibov, A DNS RR for specifying the location of services (DNS SRV), IETF RFC 2782, Feb. 2000.
[29] S. Thomson, C. Huitema, V. Ksinant, M. Souissi, DNS extensions to support IP Version 6), IETF RFC 3596, Oct. 2003.
[30] RADVD, http://www.litech.org/radvd/. Accessed Dec. 13, 2017.
[31] DHCP, https://www.isc.org/downloads/dhcp/. Accessed Dec. 13, 2017.
[32] BIND, https://www.isc.org/downloads/bind/. Accessed Dec. 13, 2017.
[33] Avahi, https://www.avahi.org/doxygen/html/index.html. Accessed Dec. 13, 2017.
[34] Wireshark, https://www.wireshark.org/. Accessed Dec. 13, 2017.

Keuntae Lee is an M.S. Student in the Department of Computer Engineering at Sungkyunkwan University since 2016. His MS advisor is Professor Jaehoon (Paul) Jeong. He got a B.S. degree from Department of Computer Engineering at KyungHee University (KHU) in 2015. His research interests include Internet of Things (IoT), Wireless Sensor Networks, Software-Defined Networking (SDN), and Network Functions Virtualization (NFV).

Seokhwa Kim is an M.S. Student in the Department of Computer Engineering at Sungkyunkwan University since 2017. His MS advisor is Professor Jaehoon (Paul) Jeong. He got a B.S. degree from Division of Information & Communication at Baekseok University (BU) in 2017. His research interests include Internet of Things (IoT) and indoor positioning systems.

Jaehoon (Paul) Jeong is an assistant professor in the Department of Software at Sungkyunkwan University in Korea. He received his Ph.D. degree in the Department of Computer Science and Engineering at the University of Minnesota in 2009. He received the B.S. degree in the Department of Information Engineering at Sungkyunkwan University and the M.S. degree from the School of Computer Science and Engineering at Seoul National University in Korea, in 1999 and 2001, respectively. His research areas include Internet of Things (IoT), Software-Defined Networking (SDN), Network Functions Virtualization (NFV), vehicular networks, wireless sensor networks, and network security.

Sejun Lee received the M.S. degree from the Department of Computer Science and Engineering at Sungkyunkwan University in Korea, under the supervision of Professor Jaehoon (Paul) Jeong. He received the B.S. degree from the School of Information and Communication Engineering at Mokpo National Maritime University in Korea in 2014. He is currently a researcher with the Ericsson-LG. His research areas include Vehicular Ad Hoc Networks (VANET), Internet of Things (IoT), and cloud.

Hyoungshick Kim received a Ph.D. in the Security Group at the Computer Laboratory, University of Cambridge with Prof. Ross Anderson, having written his thesis on complex network analysis for secure and robust communications. After completing his Ph.D., he worked as a post-doctoral fellow working in the Laboratory for Education and Research in Secure Systems Engineering (LERSSE), Electrical and Computer Engineering at the University of British Columbia (UBC). He also worked in Samsung Electronics on research projects for secure home networking and content protection before pursuing a Ph.D. His research interests are security engineering, usable security and software security.

Jung-Soo Park works for Electronics and Telecommunications Research Institute (ETRI) as the Principle Researcher since 1994. He got his Ph.D. degree from the Department of Electronics Engineering at Kyungpook National University (KNU) in 2013. He got the B.S. degree and the M.S. degree from the Department of Electronics Engineering at Kyungpook National University (KNU) in 1992 and 1994, respectively. His research interests are network security, IoT, machine to machine (M2M), network functions virtualization (NFV), vehicular networks, wireless sensor networks, and mobile ad hoc networks.