SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2018, Vol. 61 092109:1–092109:15
https://doi.org/10.1007/s11432-017-9298-3

GAGMS: a requirement-driven general address generation and management system

Ying LIU[1,2], Lin HE[1,2] & Gang REN[1,2*]

[1]*Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing* 100084, *China;*
[2]*Tsinghua National Laboratory for Information Science and Technology, Beijing* 100084, *China*

Received 19 September 2017/Accepted 22 November 2017/Published online 8 June 2018

Abstract IPv6 address generation is closely related to the manageability, security, privacy protection, and traceability of the Internet. There are many kinds of IPv6 address generation and configuration methods in the area of Internet standards and research that may cause certain problems, including the mixed operation problem of multiple IPv6 address generation schemes, the synchronization problem of the change in IPv6 address, the efficiency problem of processing large-scale concurrent IPv6 address requests, and the general model problem for mapping IPv6 addresses to other requirement spaces as identifiers. In this paper, we consider generating and managing IPv6 addresses according to network requirements. After conducting a requirement analysis of most proposed address generation schemes, we propose a general address generation model and a general address management system, which are the cores of the general address generation and management system (GAGMS). This system solves the above problems under the premise of maintaining the diversity and flexibility of the existing IPv6 address generation and configuration methods and allows networks to utilize different address generation schemes according to different requirements in different scenarios. Finally, we design a prototype system and evaluate our GAGMS to demonstrate its effectiveness, manageability, and scalability, and we have conducted trial deployment in campus networks and are trying to standardize this work in IETF.

Keywords IPv6 address generation, IPv6 address management, requirement analysis, DHCPv6, SLAAC

Citation Liu Y, He L, Ren G. GAGMS: a requirement-driven general address generation and management system. Sci China Inf Sci, 2018, 61(9): 092109, https://doi.org/10.1007/s11432-017-9298-3

1 Introduction

1.1 Problems in IPv6 address generation and configuration

In the area of Internet standards and research, many kinds of IPv6 address generation and configuration methods are proposed. They bring not only flexibility and diversity but also some problems. Because IPv6 address generation is closely related to the manageability, security, privacy protection, and traceability of the Internet, it is important to solve these problems. The problems are as follows.

1.1.1 *Mixed operation problem*

The first problem is a mixed operation problem of multiple IPv6 address generation schemes. Currently, hosts in the same network can use different address generation schemes for stateless address

*Corresponding author (email: rengang@cernet.edu.cn) auto-configuration (SLAAC) [1] to obtain addresses or fetch addresses assigned from dynamic host configuration protocol for IPv6 (DHCPv6) servers [2, 3]. For SLAAC, different operating systems have different default address generation mechanisms [4–7]. However, some address generation mechanisms are not fit for particular network scenarios. For example, some organizations disable the use of temporary addresses even at the expense of reduced privacy [7]. For DHCPv6, there already exist identity association for non-temporary addresses (IA_NA) and identity association for temporary addresses (IA_TA) options which show the different requirements of clients. Requirements also exist for DHCPv6 servers to configure address generation mechanism for different local networks. To conclude, schemes for configuring address generation mechanisms on requirements are needed.

1.1.2 *Synchronization problem*

The second problem is a synchronization problem of the change in IPv6 address. There exist many kinds of network functions related to addresses, such as auditing, access control, and source address validation. Once a host updates its addresses, the corresponding function entities should also update their corresponding stored entries. However, uniform address assignment records and processing standards are currently lacking, which causes problems. For example, if the address assignment updates of hosts are not synchronized with the accounting gateway, incorrect auditing and billings will occur. Currently, DHCPv6 active leasequery [8] solves a part of this problem by allowing other network entities to remain up to date on the DHCPv6 leases. However, the part of the problem caused by SLAAC and manual configurations remains unresolved.

1.1.3 *Efficiency problem*

The third problem is an efficiency problem of processing large-scale concurrent IPv6 address requests. One of the great advantages of IPv6 over IPv4 is the 64-bit interface identifier (IID) space. Although the number of current address requests in a subnet is far less than the maximum number that the IID space can provide (address collisions seldom occur), some predictions [9][1)][2)] show that more than 20 billion devices will connect to the Internet by 2020 and the number will grow at a faster and faster pace in the future. Therefore, IPv6 must adapt to the large-scale concurrent IPv6 address requests in a subnet in the future. In that case, routers or switches will use more resources to handle multicast NS messages in duplicate address detection (DAD) [1] process. However, if we have a central entity to manage all types of addresses, for large-scale concurrent requests, the concurrent processing mechanism of server clustering techniques will achieve greater advantages and unicast or anycast messages can be utilized.

1.1.4 *General model problem*

The fourth problem is a general model problem for mapping IPv6 addresses to other requirement spaces as identifiers. IP addresses are not only locators but also identifiers. As identifiers, IP addresses can be mapped to other requirement spaces to support multiple functions, including traceback, auditing, transition, and mobility. Currently, IETF defines the interoperations between DHCP servers and radius servers [10], and researchers also propose the extension schemes of DHCPv6 such as network identity and time generated address (NIDTGA) [11], which introduces authentication service into DHCP exchanges. That is to say, some interoperations between DHCP entities and external service entities are designed to provide precise and fine-grained services. However, there are no general uniform protocol extensions and models. To summarize, we need to work out some general uniform protocol extensions or models for introducing external services into the address assignment process.

1) IoT market forecasts. https://www.postscapes.com/internet-of-things-market-size.

2) Internet of things (IoT): number of connected devices worldwide from 2012 to 2020 (in billions). https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide.

1.2 Requirement-driven address generation and management

In addition to the diversity and flexibility of current address generation schemes, we consider that the solution must have the following properties to solve the above problems.

- Effectiveness: the performance of each scheme in the solution does not degrade compared with the originals.
- Manageability: the solution must allow all kinds of addresses and address generation schemes to be easily managed.
- Scalability: the solution must allow new address generation schemes to be easily added.
- Economy: the solution must be of low cost to deploy.

To our best knowledge, no prior work has addressed the problems from the aspect of requirement analysis and management. Therefore, we propose a requirement-driven general address generation and management system (GAGMS):

- For the mixed operation problem, GAGMS provides a uniform address generation requirement and uniform address storage structure (UASS) to better conduct address management, control, and traceback.
- For the synchronization problem, GAGMS centrally maintains the UASSes of all hosts and multicasts the updates of UASSes to related function entities.
- For the efficiency problem, GAGMS centrally records the UASSes of all hosts and allows hosts to omit the process of DAD to improve efficiency.
- For the general model problem, GAGMS provides a basic operation set to generate new address generation schemes easily, and the layered address management system features excellent scalability to add new schemes.

Our GAGMS is mainly used in access networks and can be useful for many applications. We illustrate this by a typical example scenario. Campus networks often need to conduct address management and identity management, and our GAGMS can help campus networks provide consolidated address management, precise identity management, and better user experience even if both SLAAC and DHCPv6 are used. Moreover, network operators can even analyze the address usage and address distributions to provide better service. We hope this work will help promote the development of IPv6.

1.3 Key contributions

The following are our key contributions.

- We propose the requirement analysis of address generation, which helps people understand and determine the function scope of the requirement and relationship of requirements.
- We propose a general address generation model, which helps people design new address generation schemes.
- We propose a general address management system, which can synchronize updates of UASSes among related network entities and allow networks to utilize different address generation schemes in different scenarios according to different requirements.

1.4 Paper organizations

The rest of the paper is organized as follows. Section 2 describes related studies. Next, we conduct requirement analysis in Section 3. Section 4 explains a general address generation model. In Section 5, we propose a general address management system to manage the address generation schemes. In Section 6, we discuss a prototype system that we have implemented and evaluate its performance. Finally, in Section 7, we give a conclusion and discuss next steps.

Table 1 Schemes classification by requirements

Requirement		Schemes	RFC or used to be
Easy aggregation of routes		GIRO [14], ISPSG [15]	×
Generation from MAC address		IEEE EUI-64 identifier [4, 16]	√
Protection of user privacy		Temporary address [5]	√
Verification of user identity:	solve forgery source addresses in SEND	CGA [6]	√
	solve home address problem in MIPv6	CAM [17], SUCV [18]	×
Traceback of user identity		NIDTGA [11]	×
Transformation of IPv4 addresses into IPv6 addresses		SIIT [19], IVI [20]	√

2 Related work

2.1 Address auto-configuration mechanisms

There are two address auto-configuration mechanisms in IPv6: SLAAC and DHCPv6. SLAAC can configure hosts with one or more addresses composed of network prefixes advertised by local routers and IIDs that embed hardware addresses (e.g., IEEE MAC address) or are randomly generated. DHCPv6 can provide a device with addresses assigned by a DHCPv6 server and other configurations carried in options.

The router advertisement (RA) message specified in neighbor discovery [12] contains M flag and A flag of the prefix information (PI) option. With M flag set, it indicates that DHCPv6 service is available in the network, while with A flag set, it indicates that the prefix can be used for SLAAC.

2.2 Address generation schemes

According to RFC 7421 [13], the notion of a /64 boundary was introduced after the initial design of IPv6, and a fixed identifier length has the advantages of simple network design, easy subnet addition, and easy understanding of router configurations. Most proposed IPv6 address generation schemes are designed based on a /64 boundary except some schemes that transform IPv4 addresses into IPv6 addresses.

Some IPv6 address generation schemes re-encode the prefix of IPv6 addresses (leftmost-64-bit schemes) to support advanced techniques such as route aggregation [14,15]. Most of the proposed methods redesign the scheme used for generating IIDs (rightmost-64-bit schemes) with different requirements, such as economic generation [4, 16], privacy protection [5], and validation [6, 17, 18]. Another type of address called NIDTGA has network identity (NID) and time information embedded into it. With the help of an NID Traceback Server, we can trace back a user's identity, which allows for fine-grained network management. The remaining schemes focus on transforming IPv4 addresses into IPv6 addresses (cross-64-bit schemes), such as stateless IP/ICMP translation (SIIT) [19] and IPv4/IPv6 translation (IVI) [20]. We conclude with a table that classifies schemes by the requirements in Table 1.

3 Requirement analysis

We focus on the requirement of each scheme and aim to clarify the functional scope of each requirement and the relationships among the requirements. Based on the requirement analysis, we can design new schemes that meet new requirements and build the general address generation model and address management system.

3.1 Requirement scope

As we concluded in Section 2, we summarize several common requirements of the proposed IPv6 address generation schemes, including economical address generation, protection of user privacy, verification of user identity, traceback of user identity, transformation of IPv4 addresses into IPv6 addresses, and easy aggregation of routes.

Before further analysis, we introduce a new definition, namely requirement scope, to help us demonstrate the relationship between two requirements. Requirement scope refers to a requirement's functional position within some network layer's identifier under the particular addressing method. For instance, for the network layer's IPv6 address, we can determine the location of a specific requirement's functional position within IPv6 addresses under the current routing system. More specifically, we can formalize the definition as follows:

$$L_r = f(\mathrm{id}, \mathrm{mode}, r), \tag{1}$$

where L_r, id, mode, and r represent the functional position, the identifier of some network layer, the addressing mode of that layer, and the specific requirement, respectively.

To define how to obtain L_r in detail, we must know the functional position L_{id} of id and the functional position L_m of id. For an n-bit id, the L_{id} is defined as

$$L_{\mathrm{id}} = \mathrm{id}(0, n-1). \tag{2}$$

That is, the whole identifier (from bit 0 to bit $n-1$) is the functional position of id. Note that the leftmost bit of all identifiers or feature information in this paper is numbered 0. The definition of L_m is as follows:

$$L_m = \mathrm{id}(k, l). \tag{3}$$

If the addressing mode that mode represents does not require utilizing id, both k and l equal -1 by default. Otherwise, $0 \leqslant k \leqslant l \leqslant n-1$, i.e., the part from bit k to bit l of id is used for routing. Then, L_r can be obtained as follows:

$$L_r = \begin{cases} L_m, & \text{if } r \text{ functions on mode,} \\ L_{id} - L_m, & \text{otherwise,} \end{cases} \tag{4}$$

where "$-$" represents the rest position in the minuend except for the position that the subtrahend represents.

For example, if we want to generate IPv6 addresses economically and quickly, we may choose the IEEE EUI-64 identifiers. According to the definition of the requirement scope, the requirement does not function on the prefix. Therefore, the IID is the requirement scope. Furthermore, we calculate the functional scopes of these requirements as shown in Table 2.

3.2 Requirement relationships

We can clarify the relationships between the requirements through the requirement scopes mentioned above. We define three types of relationships between two requirements.

3.2.1 *Complete compatibility*

If two requirement scopes do not intersect, the two requirements are in complete compatibility, which indicates that they can both function on the same IPv6 address simultaneously.

3.2.2 *Complete conflict*

If two requirement scopes are equal, the two requirements are in complete conflict, which means that they cannot function on the same IPv6 address simultaneously.

3.2.3 *Partial compatibility*

If two requirement scopes intersect partially, the two requirements are in partial compatibility. Probably, they may function on the same IPv6 address.

Table 3 shows the pairwise relationships of several types of requirements. The relationship between two requirements reflects the possibility of two requirements functioning on the same IPv6 address. Mechanisms in complete compatibility can function on the same address completely. However, mechanisms in complete conflict can never function on the same address.

Table 2 Functional scopes of requirements

Goal	Requirement	Functional scope
Identifier	(a) Generation from MAC address	IPv6(64, 127)[a]
	(b) Protection of user privacy	IPv6(64, 127)
	(c) Verification of user identity	IPv6(64, 127)
	(d) Traceback of user identity	IPv6(64, 127)
	(e) Transformation of IPv4 addresses into IPv6 addresses	IPv6(32, 127)
Locator	(f) Easy aggregation of routes	IPv6(0, 63)

a) IPv6(m, n) represents the part of IPv6 address from bit m to bit n.

Table 3 Relationships among requirements [a]

	(a)	(b)	(c)	(d)	(e)	(f)
(a)	–[b]	×[c]	×	×	×	◯[d]
(b)	×	–	×	×	×	◯
(c)	×	×	–	×	×	◯
(d)	×	×	×	–	×	◯
(e)	×	×	×	×	–	⊗[e]
(f)	◯	◯	◯	◯	⊗	–

a) (a)–(f) represent the requirements mentioned in Table 2.
b) "–" represents the same requirement.
c) "×" means complete conflict.
d) "◯" means complete compatibility.
e) "⊗" means partial compatibility.

The requirement analysis of address generation helps us understand the function scope of the requirement and relationships among requirements. Once we know the requirement scopes and requirement relationships, we can design schemes satisfying more than one requirement.

4 General address generation model

We will introduce an important infrastructure, namely basic operation set, to build four types of mappings that are the core of the address generation model. Based on the basic operation set and address generation model, we can write a basic address generation operation library to implement current schemes or design new schemes, which helps solve the general model problem.

We take the generation schemes of IIDs as an example to analyze the essence of IPv6 address generation schemes. When generating complete IPv6 addresses, the process is similar to that of generating IIDs, only adding a prefix before the IID.

4.1 Basic operation set

We extract seven operations to form the corresponding mappings of IPv6 address generation schemes. There may be new operations in future address generation schemes, so we allow new elements to be added into the basic operation set. The operations in the basic operation set can be classified into two types: simple basic operation set and complex basic operation set.

4.1.1 *Simple basic operation set*

We have extracted five simple operations to form the simple basic operation set—i.e., $S = \{$Invert, Insert, Concatenate, Replace, Truncate$\}$.

- Invert(x, n): invert bit n of input x.
- Insert(x, n, s): insert s after bit n of input x.
- Concatenate$(x, y, \ldots)$: concatenate input $[x, y, \ldots]$ sequentially.
- Replace(x, n, m, s): change from bit n to bit m of input x into s. Note that the length of s must be equal to $m - n + 1$. When $n = m$, change only one bit of input x.

• Truncate(x, n, m): truncate from bit n to bit m of input x as the output.

4.1.2 *Complex basic operation set*

We have extracted two complex operations to form the complex basic operation set—i.e., $C = \{\text{Encrypt}, \text{Hash}\}$.

• Encrypt(x, k): use some specific encryption algorithm to encrypt input x with key k. Encryption algorithms can be IDEA, AES, RSA, etc.

• Hash(x): calculate the Hash digest value of input x. Hash algorithms can be MD5, SHA1, SHA256, etc.

Note that the operations in the complex basic operation set are a class of operations; e.g., the Hash operation can be implemented by any secure Hash algorithm.

4.2 Four types of mappings

Feature-embedded schemes allow us to transform feature information into IIDs, which is very similar to mappings, so we utilize mappings to describe the conversion process. Interface identifiers can be seen as the image of the feature information under a mapping—i.e.,$f : C \rightarrow \text{IID}$, where C denotes the feature information set, IID denotes the interface identifier set, and f denotes an address generation scheme. We divide all address generation mappings into four types according to the usage of complex basic operations.

4.2.1 *Simple mappings*

These mappings consist of simple basic operations. We can formulate them as follows:

$$I = \{f | f \in S^n, n \in N\}, \tag{5}$$

where S denotes the simple basic operation set, and S^n denotes that n basic operations in S are combined to form one nesting operation.

For instance, the IEEE EUI-64 identifier scheme transforms MAC addresses into IIDs. First, we insert "0xfffe" after the leftmost 24 bits of the MAC address and then invert the U/L bit. Therefore, the IEEE EUI-64 identifier scheme is equivalent to the following mapping:

$$f_1(\text{mac}) = \text{Invert}(\text{Insert}(\text{mac}, 23, \text{0xfffe}), 6). \tag{6}$$

The input length of this type of mapping is shorter than the length of IID. If its length is longer than the length of IID, we may obtain repeated results. The same is true for encryption mappings and Hash mappings.

4.2.2 *Encryption mappings*

These mappings consist of simple basic operations and Encrypt operations. Encryption mappings are used to encrypt feature information within IIDs. We formulate them as follows:

$$\text{II} = \left\{ f | f \in S^{n_1} \circ \text{Encrypt}^{m_1} \circ S^{n_2} \circ \text{Encrypt}^{m_2} \circ \cdots, n_i, m_j \in N, \sum m_j > 0 \right\}, \tag{7}$$

where "$\circ$" denotes the nesting of basic operations, and "Encrypt^{m_j}" denotes that n Encrypt operations combined with each other to form one nesting operation.

Notice that the output of the encrypt operation must be 64 bits, so we must choose an encryption algorithm with a 64-bit output. For instance, the generation of NIDTGA [11] uses the IDEA algorithm to encrypt the concatenation of NID and time information to obtain an IID. Therefore, its corresponding mapping can be written as follows:

$$f_2(\text{nid}, \text{time}, \text{key}) = \text{Encrypt}(\text{Concatenate}(\text{nid}, \text{time}), \text{key}). \tag{8}$$

4.2.3 *Hash mappings*

The elements in Hash mappings consist of simple basic operations and Hash operations. We can formulate them as follows:

$$\text{III} = \left\{ f | f \in S^{n_1} \circ \text{Hash}^{m_1} \circ S^{n_2} \circ \text{Hash}^{m_2} \circ \cdots, n_i, m_j \in N, \sum m_j > 0 \right\}. \tag{9}$$

Because we use Hash operations whose results are often longer than 64 bits, the mappings should use a Truncate operation to truncate the results equal to or less than 64 bits. For example, temporary addresses in [5] use MD5 to encrypt the concatenation of the EUI-64 identifier and a history/random value to obtain the first 64 bits of the result as a new IID and then change the U/L bit into 0. Thus, the corresponding mapping can be written as follows:

$$f_3(\text{eui64}, \text{history}) = \text{Replace}(\text{Truncate}(\text{Hash}(\text{Concatenate}(\text{eui64}, \text{history}))0, 63), 6, 6, 0). \tag{10}$$

4.2.4 *Complex mappings*

There are another kind of mappings that use both Encrypt operations and Hash operations. We formulate the mappings as follows:

$$\text{IV} = \{ f | f \in S^{n_1} \circ C^{m_1} \circ S^{n_2} \circ C^{m_2} \circ \cdots, n_i, m_j \in N, \text{count}(f.\text{Encrypt}) > 0, \text{count}(f.\text{Hash}) > 0 \}, \tag{11}$$

where count(f.Encrypt) and count(f.Hash) denote calculating the number of Encrypt operations and Hash operations in mapping f, respectively.

These mappings use both Encrypt and Hash operations, so they are slower and more complex. For example, an NIDTGA extension scheme can be designed. We can use a kind of user id (such as a student identifier or a Citizen ID card number) as the input to a Hash algorithm and then encrypt the Hash output in cipher block chaining mode. Its corresponding mapping can be written as follows:

$$f_4(\text{id}, \text{key}) = \text{Encrypt}(\text{Hash}(\text{id}), \text{key}). \tag{12}$$

Note that using both Encrypt and Hash operations does not gain much more security than only using one kind of them by choosing secure algorithms and keeping keys safe.

4.3 Analysis

When designing a new address generation scheme, we can introduce new operations to complete the design if there are no fitting operations in the basic operation set. The address generation model helps us determine the essence of the address generation schemes, guides us to design new schemes, and can even be offered as an operation library to provide convenience for people to implement these schemes.

5 General address management system

In this section, we propose a general address management system to manage different kinds of addresses and allow networks to adopt different address generation schemes in different scenarios according to different requirements. Overall, the system has three logical layers—namely, requirement layer, functionality layer, and resource layer, as depicted in Figure 1. Before we demonstrate the details of each layer, we must introduce uniform address storage structure (UASS), which stores both SLAAC and DHCPv6 address assignments. UASS mainly includes the IPv6 address, valid lifetime, expire time, preference time, and hardware address. DUID and IAID can also be included for DHCPv6 [2,3].

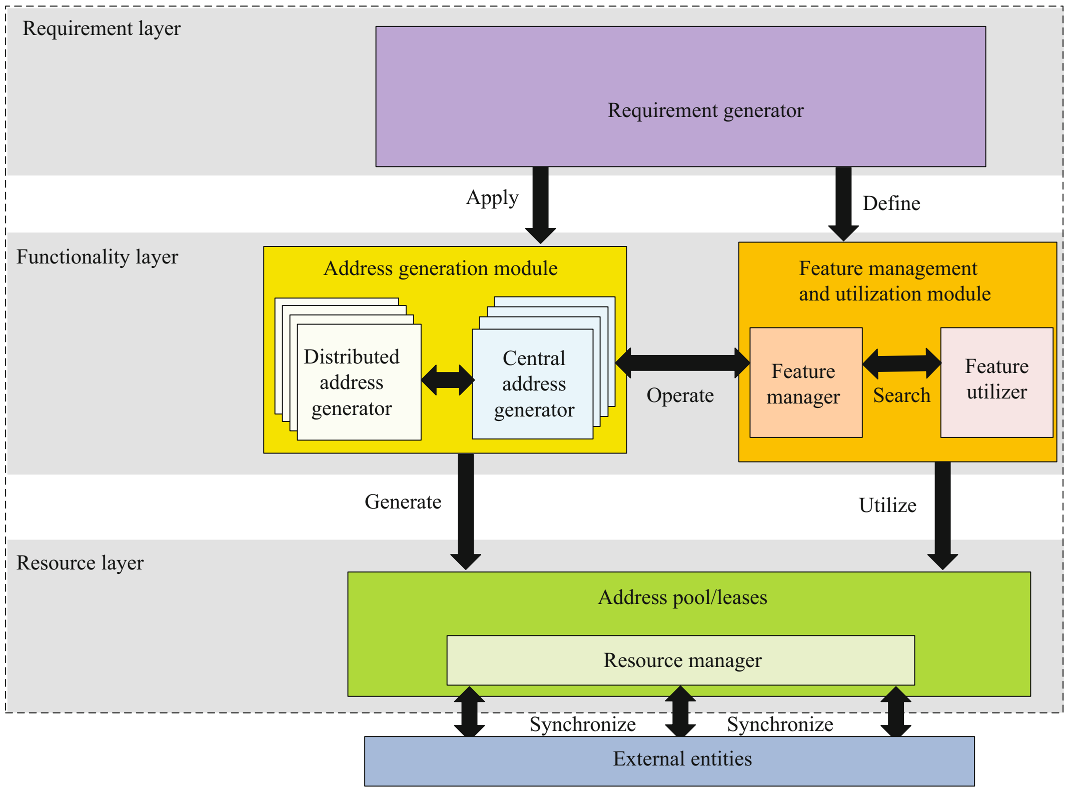

Figure 1 (Color online) Logical structure of GAGMS.

5.1 Requirement layer

The requirement layer comprises a requirement generator (RG). RG determines the requirements for the network, which is the base to solve the mixed operation problem.

RG reads the configuration requirements and informs the functionality layer of the requirement decisions. Note that the requirement decisions from RG should be verified. A necessary verification scheme should be applied to the requirement decisions or some network devices' configurations should be used to guarantee that the requirement decisions are trustworthy.

5.2 Functionality layer

The functionality layer comprises a set of functional modules, each of which has specific functionalities, and is responsible for accomplishing specific tasks. At least, this layer has an address generation module to generate and assign addresses that meet the requirements. Other functional modules can also be added to this layer—e.g., feature management and utilization module.

5.2.1 *Address generation module*

The address generation module is informed by RG of the requirements and generates addresses that meet the requirements. When generating addresses, some schemes may need to interact with third servers, e.g., Feature Manager. Address generators are the core of the address generation in the general address management system. The address generation module can have a set of address generators, which can be classified into two types, namely distributed address generators (DAGs) and central address generators (CAGs).

A DAG is located at a host. It implements the address generation or request functions of the schemes mentioned above as a system process. When a DAG receives requirement decisions from RG, it first verifies the requirement decisions, and then it starts the address generation process or requires addresses from CAG. After generating addresses successfully, it informs resource manager (RM) of its UASSes without performing DAD. If some of the addresses are duplicates in the response message from RM, it restarts the above process. Note that if the DAG has updated its UASSes, it should also inform RM of its UASSes.

A CAG assigns addresses to DAGs. When a CAG receives the requirement decisions from RG, it first verifies the requirement decisions. Then, it starts the address assignment functions. After assigning addresses to DAGs, it informs RM of DAGs' UASSes. If the corresponding address generation scheme of the selected requirement does not have this part, it ignores this information. Note that if the CAG has updated the UASSes of DAGs, it should inform RM of the changed UASSes.

5.2.2 *Feature management and utilization module*

The feature management and utilization module is defined by the user's requirements. Different requirements correspond to different methods of feature management and utilization. The feature management and utilization module includes a feature manager (FM) and a feature utilizer (FU).

An FM is a collection of feature information about the domain. It can interact with CAG—e.g., authenticate a client's identity. It also cooperates with FU to finish the task that RG defines. When FM receives the requirement decisions from RG, it first verifies the requirement decisions and then starts the corresponding management functions. If the corresponding address generation scheme of the selected requirement does not have this part, it ignores this message.

An FU is responsible for extracting or utilizing the feature information from the address. For example, if a user's identity is embedded within the address, the administrator can use the FU to extract the identity. When FU receives the requirement decisions from RG, it first verifies the requirement decisions and then starts the corresponding utilization functions. If the corresponding address generation scheme of the selected requirement does not have this part, it ignores this information. Not anybody can use FU; only authorized users can use it. FU cooperate with FM to meet the requirement that RG defines.

5.3 Resource layer

The resource layer comprises a set of addresses, each of which meets the requirements defined in RG. They are generated by DAGs or CAGs and can also be used to achieve goals that RG defines.

An RM is responsible for managing all UASSes in the network and synchronizing updates of UASSes with external entities. It receives and stores the UASSes from DAGs and CAGs. If the address in UASS is a duplicate, it asks the DAG to restart the address generation process, which omits the DAD to solve the efficiency problem. When external entities request updates of UASSes, RM verifies the identities of external entities and then synchronizes updates of UASSes with external entities.

6 Implementation and evaluation

6.1 Methodology

In this section, we discuss our implementation of a prototype system of GAGMS and evaluate its effectiveness on the address generation time of the tested schemes. Moreover, we analyze the manageability and scalability of the system.

We have implemented RG on the router. Routers send RA messages with M flag or A flag of the PI option set and a new managed address generation scheme option or resource manager address option added.

When receiving RA messages, a DAG checks whether M and A flags are set. If A flag of the PI option is set, it generates addresses according to scheme type2 of the PI option as shown in Figure 2. After generating addresses, the DAG informs RM of its UASSes using the address in the new resource manager address option of the RA message, as shown in Figure 3. We use RA guard to filter rogue RA messages.

If M flag is set, a DAG requires addresses from CAGs, which implement each scheme's specific address generation functions based on the original DHCPv6 server. CAGs assign addresses to DAGs according

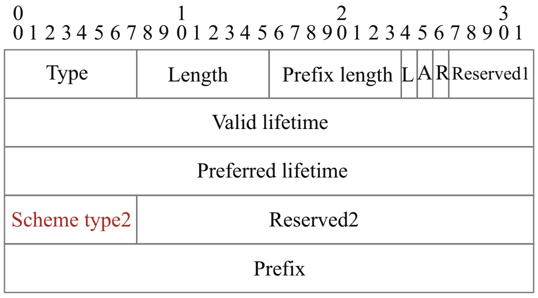

Figure 2 (Color online) Modified prefix information option of RA.

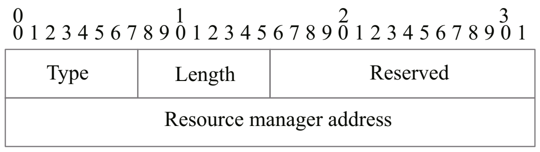

Figure 3 Resource manager address option of RA.

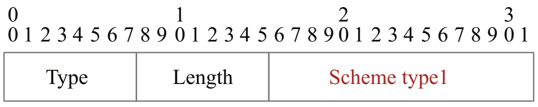

Figure 4 (Color online) Managed address generation scheme option.

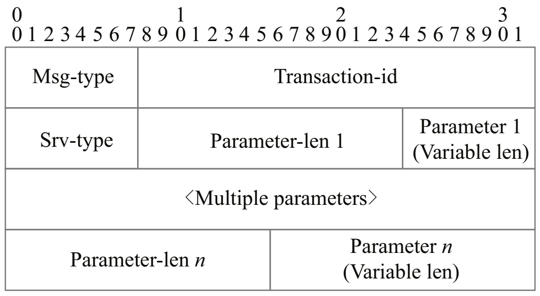

Figure 5 Service message between CAG and FM.

to the scheme type1 of the new managed address generation scheme option in the RA message, as shown in Figure 4. In such cases, a network can deploy a DHCPv6 server to work as both CAG and RM.

The communications between CAG and FM use the message format as shown in Figure 5. The msg-type indicates the type of the message and includes "SERVICE_REQUEST" and "SERVICE_REPLY". The transaction-id are copied from the DHCP message to identify this message exchange. The srv-type shows which type of service is used, e.g., authentication service of NIDTGA.

RM use a multicast address to advertise updates of UASSes to network entities that join the multicast group. As for FM and FU, they are implemented with each scheme's corresponding functions. For example, FM implements the functions of NID Management Server in NIDTGA [11], while FU implements the functions of NID Traceback Server.

Regarding security, secure DHCPv6 [21] can be applied to protect the security of interactions between CAGs and DAGs in GAGMS. At the same time, the messages between other components in GAGMS should be encrypted.

We have written a basic address generation operation library in C++ as described in Section 4. Based on the library, we have implemented three schemes, namely IEEE EUI-64 identifiers, temporary addresses, and NIDTGA from simple mappings, Hash mappings, and encryption mappings, respectively, to show the effectiveness of the system. All GAGMS components are implemented on and run on Dell servers with a 2.9 GHz Inter Xeron E5-2690 CPU and 4 GB of memory. The programs are coded in C/C++. The Internet Assigned Number Authority delegates prefixes to regional Internet Registries which in turn delegate smaller sub-prefixes to ASes. Each AS generates addresses by appending generated IID to the sub-prefix. As GAGMS focuses on the IID generation and is mainly used in access networks, we have deployed GAGMS in five campus networks (Tsinghua University, Peking University, Beijing University of Posts and Telecommunications, Southeast University, and Shanghai Jiao Tong University) of CNGI-CERNET2 to evaluate its performance and accumulate deployment experience. The network topology in each campus network is deployed as shown in Figure 6(a).

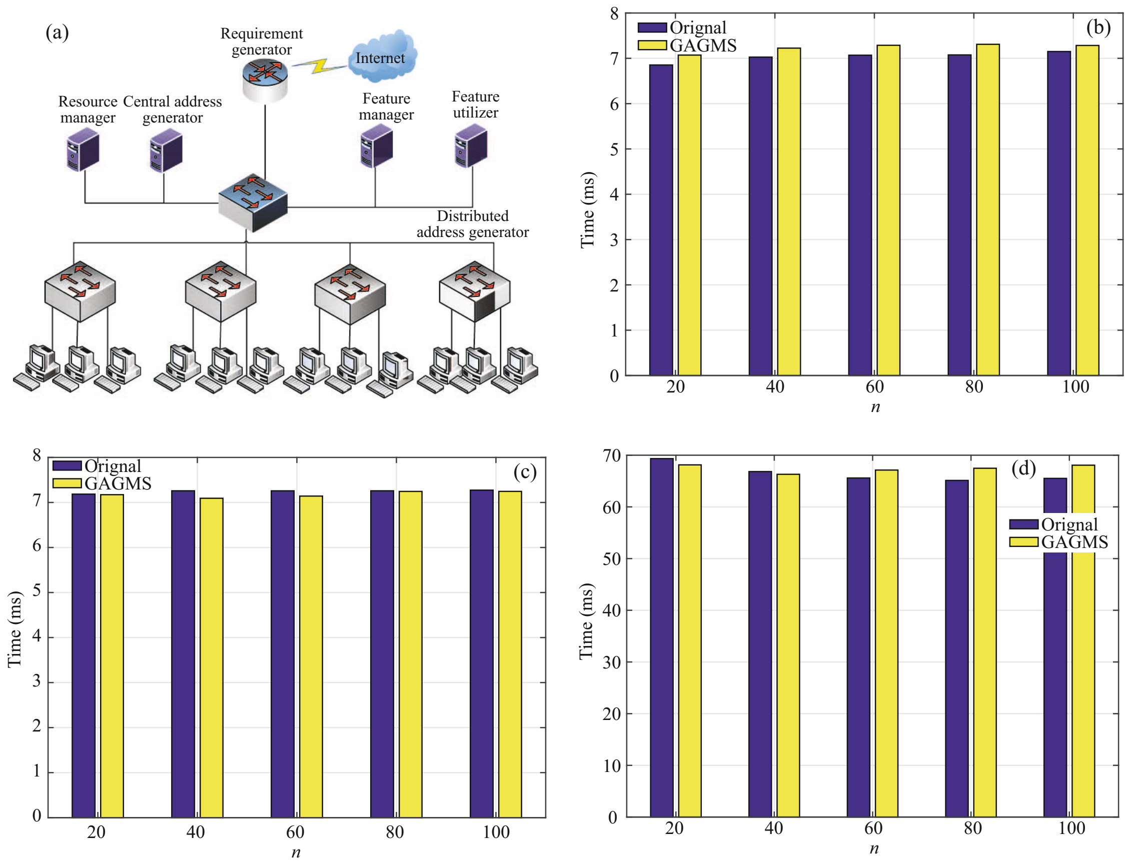

Figure 6 (Color online) Deployment topology and evaluation of effectiveness. (a) GAGMS system deployment topology; (b) IEEE EUI-64 identifiers. (c) Temporary addresses; (d) NIDTGA.

6.2 Effectiveness

If the inputs of an address generation scheme and its corresponding implementation in GAGMS are the same, the outputs are the same. Due to the article length limitation, we assume that each specific address generation algorithm based on the existing IETF standard is implemented correctly, we mainly focus on the evaluation of effectiveness. We have tested the time to configure addresses for interfaces 100 times in each scheme. Figure 6(b)–(d) shows the results of IEEE EUI-64 identifiers, temporary addresses, and NIDTGA. We find that the mean time for configuring the addresses using each scheme and its corresponding implementation in GAGMS is nearly the same, which shows that the performance of an address generation scheme in GAGMS does not degrade.

The lack of synchronization of updates of UASSes makes it error-prone for some function entities to provide services. Network operators usually use unicast or anycast messages to provide address assignment updates for specific network entities according to their experiences, which is inefficient. Some scenarios that need dynamic address assignment updates are even neglected by network operators. In GAGMS, we use multicast messages to dynamically synchronize updates of UASSes with related entities, which improves the correctness of network management and the efficiency of address assignment updates. Figure 7 shows that the update time of UASSes is linearly correlated to the number of UASSes, which is acceptable.

Since GAGMS leverages central address management of all kinds of addresses, it is unnecessary for hosts to perform DAD, which costs much time, especially when there are large-scale concurrent address requests. Although optimistic DAD [22] minimizes address configuration delays, addresses in the optimistic state are still constrained for outgoing connections or overriding entries of neighbor caches.

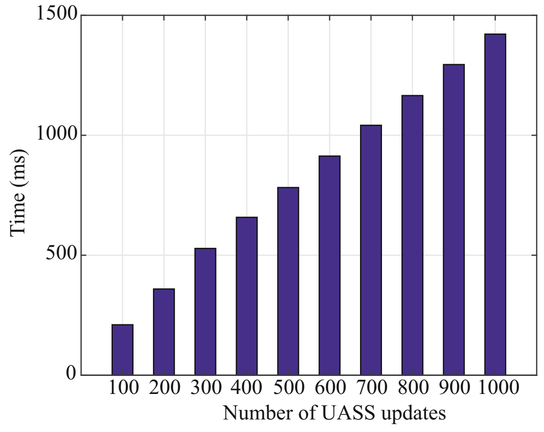

Figure 7 (Color online) Update time of UASSes in GAGMS.

Table 4 Schemes and their corresponding mappings

Type	Scheme	Mapping
Simple mappings	GIRO	f(as_number, geo_loc, sid, subnet_host) = Concatenate(as_number, geo_loc, sid, subnet_host)
	ISPSG	f(as_number, geo_loc, id) = Concatenate(as_number, geo_loc, id)
	SIIT/IVI	f(prefix, IPv4, zeros) = Concatenate(prefix, IPv4, zeros), length(prefix) $\in \{32, 40, 48, 56, 64, 96\}$
	IEEE EUI-64 identifier	f(mac_addr) = Invert(Insert(mac, 23, 0xfffe), 6)
Hash mappings	Temporary address	f(eui64, history) = Replace(Truncate(Hash(Concatenate(eui64, history)), 0, 63), 6, 6, 0)
	CAM	f(pub_key) = Insert(Truncate(Hash(pub_key), 0, 61), 5, 00)
	SUCV	f(imprint, pub_key) = Replace(Truncate(Hash$_1$ (Hash$_2$(imprint), Hash$_2$(pub_key)), 0, 63), 6, 6, 0)
	CGA	f(final_modifier, prefix, collision_count, pub_key, options, sec) = Replace(Replace(Replace(Truncate(Hash(Concatenate(final_modifier, prefix, collision_count, pub_key, options)), 0, 63), 0, 2, sec), 6, 6, 0), 7, 7, 0)
Encryption mappings	NIDTGA	f(nid, time, key) = Encrypt(Concatenate(nid, time), key)
Complex mappings	NIDTGA-extension	f(id, key) = Encrypt(Hash(id), key)

6.3 Manageability

GAGMS allows administrators to switch among the address generation schemes in the administrative domain. Table 4 shows several address generation schemes and their corresponding mappings. Notice that GIRO, ISPSG and SIIT/IVI mappings generate IPv6 addresses, whereas the others are mappings that generate IIDs. Apparently, address generation schemes in complete conflict can be never set as the used mechanisms in the administrative domain simultaneously. As shown in Figure 8, the time for a host to switch from one requirement (A) to another (B) nearly equals the original address configuration time, which means that the switching process costs little.

6.4 Scalability

When referring to scalability, it is important that adding a new address generation scheme to GAGMS will not result in a performance penalty or an increase in management complexity. It is simple to use basic operation set to generate new address generation mappings. Apparently, there are three choices for a new address generation scheme, namely a leftmost-64-bit scheme, a rightmost-64-bit scheme, and a cross-64-bit scheme. We consider a new rightmost-64-bit scheme named the stable and semantically opaque identifier in [7] as an example to explain the scalability of GAGMS. According our general address

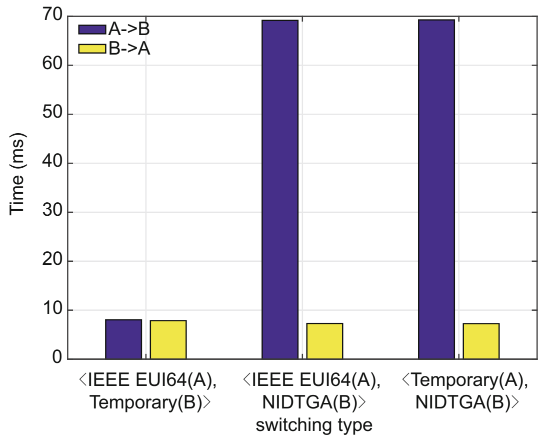

Figure 8 (Color online) Switching time among schemes in GAGMS.

generation model, we can obtain its corresponding mapping as

$$\begin{aligned} &f(\text{prefix}, \text{iface}, \text{network\_id}, \text{dad\_counter}, \text{key}) \\ &\quad = \text{Truncate}(\text{Hash}(\text{Concatenate}(\text{prefix}, \text{iface}, \text{network\_id}, \text{dad\_counter}, \text{key})), 0, 63). \end{aligned} \tag{13}$$

This mapping is a Hash mapping. We must add a new requirement option to RG and add the corresponding address generation scheme to DAGs and CAGs. Regarding FM and FU, we do not need to add anything else. Finally, we finish the necessary communications between any two modules. In short, we must classify the functions of a new scheme into three parts and add these parts into their corresponding modules, respectively. Therefore, the general uniform protocol extensions and models give GAGMS excellent scalability.

7 Conclusion and future work

The address generation and configuration methods of IPv6 cause four problems: mixed operation problem, synchronization problem, efficiency problem, and general model problem. We design a requirement-driven general address generation and management system called GAGMS to solve the problems. GAGMS provides central and uniform address management and synchronization of updates of UASSes and has fine manageability for changing among existing IPv6 address generation schemes and good scalability for designing a new one.

In future work, we will first design and implement the automatic generation of new IP address generation schemes according to requirements. Secondly, we will promote GAGMS to the whole CNGI-CERNET2, which is an IPv6-only network built by China Education and Research Network (CERNET). Thirdly, we have submitted an IETF draft related to this work [23] to DHC working group and are trying to standardize this work.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61402257, 61772307) and Tsinghua University Self-determined Project (Grant No. 2014z21051).

References

1 Narten T, Jinmei T, Thomson S. IPv6 stateless address autoconfiguration. RFC 4862. 2007
2 Droms R, Bound J, Volz B, et al. Dynamic host configuration protocol for IPv6 (DHCPv6). RFC 3315. 2003
3 Mrugalski T, Siodelski M, Volz B, et al. Dynamic host configuration protocol for IPv6 (DHCPv6) bis. draft-ietf-dhc-rfc3315bis-10. 2017
4 Hinden R, Deering S. IP version 6 addressing architecture. RFC 4291. 2006
5 Narten T, Draves R, Krishnan S. Privacy extensions for stateless address autoconfiguration in IPv6. RFC 4941. 2007
6 Aura T. Cryptographically generated addresses (CGA). RFC 3972. 2005

7 Gont F. A method for generating semantically opaque interface identifiers with IPv6 stateless address autoconfiguration (SLAAC). RFC 7217. 2014
8 Raghuvanshi D, Kinnear K, Kukrety D. DHCPv6 active leasequery. RFC 7653. 2015
9 Hosain S Z. Reality check: 50B IoT devices connected by 2020 beyond the hype and into reality. RCR Wireless News. http://www.rcrwireless.com/20160628/opinion/reality-check-50b-iot-devices-connected-2020-beyond-hype-reality-tag10
10 Yeh L, Boucadair M. RADIUS option for the DHCPv6 relay agent. RFC 7037. 2013
11 Liu Y, Ren G, Wu J P, et al. Building an IPv6 address generation and traceback system with NIDTGA in address driven network. Sci China Inf Sci, 2015, 58: 120102
12 Narten T, Nordmark E, Simpson W, et al. Neighbor discovery for IP version 6 (IPv6). RFC 4861. 2007
13 Carpenter B, Chown T, Gont F, et al. Analysis of the 64-bit boundary in IPv6 addressing. RFC 7421. 2015
14 Oliveira R, Lad M, Zhang B, et al. Geographically informed inter-domain routing. In: Proceedings of IEEE International Conference on Network Protocols (ICNP), Beijing, 2007. 103–112
15 Yin X, Wu X, Chon K, et al. ISPSG: Internet service provider-separated geographic-based addressing and routing. In: Proceedings of the Global Communications Conference Workshops, Hawaii, 2009. 1–6
16 Hinden R, Deering S. IP version 6 addressing architecture. draft-ietf-6man-rfc4291bis-09. 2017
17 O'Shea G, Roe M. Child-proof authentication for MIPv6 (CAM). ACM SIGCOMM Comput Commun Rev, 2001, 31: 4–8
18 Montenegro G, Castelluccia C. Statistically unique and cryptographically verifiable (SUCV) identifiers and addresses. In: Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, 2002
19 Bao C, Li X, Baker F, et al. IP/ICMP translation algorithm. RFC 7915. 2016
20 Bao C, Huitema C, Bagnulo M, et al. IPv6 addressing of IPv4/IPv6 translators. RFC 6052. 2010
21 Li L, Jiang S, Cui Y, et al. Secure DHCPv6. draft-ietf-dhc-sedhcpv6-21. 2017
22 Moore N. Optimistic duplicate address detection (DAD) for IPv6. RFC 4429. 2006
23 Ren G, He L, Liu Y. Multi-requirement extensions for dynamic host configuration protocol for IPv6 (DHCPv6). draft-ren-dhc-mredhcpv6-00. 2017