Towards Variable Length Addressing for Scalable Internet Routing

Feng Wang*, Xiaozhe Shao†, Lixin Gao†, Hiroaki Harai‡ and Kenji Fujikawa‡
*School of Engineering and Computational Sciences
Liberty University, Lynchburg, VA 24515, USA
†Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003, USA
‡National Institute of Information and Communications Technology
Tokyo 184-8795, Japan

***Abstract*—The Internet is facing the accelerating growth of routing table size. Backbone routers' routing table has already reached 512k entries, which has a negative effect on the scalability of the Internet. Hierarchical addressing and locator/ID separation solutions have been proposed to address the scalability issue. However, there has been little focus on how to efficiently represent hierarchical location addresses for a large scale distributed network, such as today's Internet. In this paper, we present a variable-length address encoding method to represent hierarchical location addresses. Our analysis and evaluation results show that 1) it is difficult to use fixed-length encoding to represent hierarchical location addresses for a large scale network; and 2) the proposed variable-length addresses could guarantee the scalability property of hierarchical addressing, and alleviate the inefficiency of address space due to fixed-length addresses.**

I. Introduction

The Internet is facing the accelerating growth of routing table size. In the current Internet, an address acts as both an identifier (ID) and location for an end host. A multi-homed host will need to make its address visible through multiple Internet service providers. Therefore, it is not possible for a provider to aggregate all of the addresses belonging to its customers. This will seriously limit the scalability of the control plane. There are already more than 512K active entries in IPv4 BGP table, which is a big burden for edge routers. Since the Internet has become integrated into our daily lives, the scalability of the Internet has become extremely important.

There are two trends in solving the scalability issue: locator/ID separation solutions and hierarchical addressing scheme [1]–[9]. For example, in the locator/ID separation based solutions, such as LISP, Shim6, ILNP, HIP and MILSA [1]–[6], [10], the separation of node identification and location can support mobility and multi-homing. On the other hand, hierarchical automatic addressing provides another way to solve the scalability problem due to multi-homing [11], [12]. Because every AS inherits prefixes from its providers, hierarchical addressing allows providers to aggregate their customers' addresses and announce the aggregated prefixes. Thus, the routing table can be reduced significantly.

Some existing addressing schemes have incorporated both methods, such as HANA [13]. In those addressing schemes, the address format follows locator/ID separation technology, while the location allocation is based on a hierarchical routing topology. However, there has been little focus on how to efficiently represent hierarchical location addresses. On the surface the question seems trivial. A direct way is to use fixed-length addresses, but fixed-length addresses do not scale well. First, it is very difficult to pre-assign address length to accommodate a large-scale hierarchical addressing network, like today's Internet. Second, it is difficult to automatically allocate fixed-length locators with minimal human intervention.

In this paper, we first employ a prefix-based labeling scheme to label hierarchical location addresses. Then, as a baseline, a naive encoding method based on *fixed-length encoding* is applied to implement the prefix-based labeling scheme. For comparison, we propose an efficient encoding method based on *variable-length encoding*. In order to understand the degree to which the Internet can benefit from the proposed encoding methods, we compare the performance of the two methods in terms of address length and flexibility in using address space. Based on the real routing data, our study shows that the fixed-length (32 bits) representation is not large enough to represent all hierarchical location addresses in today's Internet. A simple solution is to use a large address space, like 128-bit IPv6 addresses. However, the main issue is its inefficiency of using address space because most of the bits are not used [14]. Our analysis and evaluation results show that the variable-length encoding method resolves both the scalability problem and the inefficiency of address spaces. By using variable-length location addresses, the maximum address length is no more than 32 bits. Most importantly, variable-length encoding method can enable automatic address allocation with minimal human intervention, and the address spaces can be extended with low cost operations. Furthermore, the location addresses can work in IPv4 network environments without any modification.

The rest of paper is organized as follows: We introduce hierarchical addressing and present the challenges of implementing the proposed hierarchical addressing scheme in Section II. We present a fixed-length encoding method and discuss its performance in Section III. We propose a variable-length encoding scheme in Section IV. We evaluate the performance of the encoding methods in Section V. We present related work

978-1-5090-5252-3/16/$31.00 ©2016 IEEE

in Section VI. We conclude the paper in Section VII with a summary.

II. Hierarchical Location Addressing

In this section, we first present an overview of hierarchical location addressing, which is extended from our previous work [9], [12], [13], [15]. We then discuss the challenges of implementing the proposed addressing scheme.

A. Overview

We employ a locator/ID address format by separating the location and the identity of a host. An address of each host consists of two parts: a *host ID* and a *locator*. Since the location of a node may be dynamic and change with node movement, a locator is used to describe the network attachment point(s) to which a host connects. Each host is assigned a globally unique number, which would not change even if the host moved to another location. Host IDs are assigned by the administrator of each network. Each host may have multiple locators representing different routes toward it. In this paper, we focus on locator allocation and representation because host IDs are allocated by each AS.

Locator assignment follows the inter-domain hierarchy where each AS obtains a locator from a provider in the upper layer. Mathematically, we use a hierarchical AS graph $G = (V, E)$ to model the Internet AS-level topology, where the node set V represents a set of ASes and the edge set E represents AS relationships between ASes. A directed edge represents a provider-to-customer relationship. In an AS graph, locators are labeled by a prefix-based labeling scheme. For an AS, its locator consists of a *provider label* followed by a *self label*, which are both assigned by its provider. The provider label identifies a provider of the AS, while the self label identifies the AS. We use delimiter '.' to join a set of labels. Note that the delimiter is used to separate a set of labels, instead of octet values used in an IPv4 address.

More specifically, let G be an AS graph with root r. The locator of node n is represented as the form of $provider(n).self(n)$, where $provider(n)$ and $self(n)$ are defined as follows:

1) (**Tier-1 scenario**): If n is directly connected to root r, and is the ith large customer, then $provider(n) = \phi$ and $self(n) = i$, where ϕ represents an empty locator.
2) (**non Tier-1 scenario**): Otherwise, if n is the ith large customer of node v, $provider(n) = provider(v).self(v)$ and $self(n) = i$.

Because Tier-1 ASes do not have any provider, their provider labels are empty. A centralized authority, such as IANA, can assign them an unique self label. So, their self labels are their locators. The value of the self labels implies the size of the Tier-1 ASes. In this paper, a customer's self label is determined based on the number of descendants of the customer. That is, the ith largest customer has a self label of i. For example, the self label of the largest Tier-1 AS in the Internet is assigned "1". For other non Tier-1 ASes, their providers first determine a self label for each customer according to the size of the customer. Then, each customer's locator is generated by concatenating the provider's locator and the customer's self label.

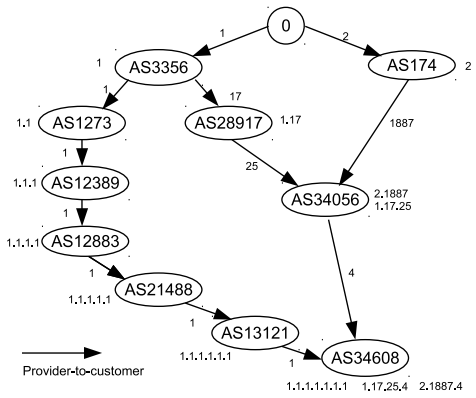


Fig. 1. An example of prefix-based labelling. The label beside an edge represents a self label, and the label beside a node corresponds to its locator.

An example running through this paper is used to demonstrate the prefix labeling scheme. Note that this example comes from the real scenarios in today's Internet, which is derived from CAIDA [16]. In Fig.1, the root node is labeled with an empty locator. AS3356 and AS174 are Tier-1 providers. Suppose that their self labels are `1` and `2`, respectively. AS34056 obtains two locators `1.17.25` and `2.1887`, where locator `1.17.25` is the concatenation of AS28917's provider label `1.17` and self label `25`, which are assigned by AS28917. AS28917's locator `1.17` consists of AS3356's provider label `1` and self label `17`. From this example, we see that a locator can be considered as a sequence of self labels.

According to the prefix-based labeling scheme, customers having the same providers should share a common prefix. It is interesting to point out that the proposed prefix-based labeling scheme does not have a restriction on the length of self labels. We can use an infinite number of integers to represent self labels.

An inter-domain location routing protocol, which can be extended from BGP or HANA [9], could be used to propagate routing information about locators. Due to the space limitation, we only introduce the policies related to locator announcements, and do not discuss the technique details here. A customer announces the locators that are inherited from its provider to the provider. That means, the locators assigned by other providers cannot be advertised to the provider. Furthermore, a provider does not need to inform customers of all its locators. It can announce aggregated locators to its non-provider neighbors, such as customers and peers. Providers shall aggregate their customers' locators and announce the whole locator block rather than the specific locator for each customer.

Note that each provider should determine which locator will be assigned to a customer. A provider could assign all of its locators, or just a subset of locators. However, if a provider assigns all of its locators to customers, this may result in some customers inheriting a large number of locators. As a

result, routing tables will be unacceptably large. As posted by previous work [17], this problem can impact the scalability of Internet routing. Thus, each provider should negotiate with its customers to determine the locators that the customers prefer. For example, in the previous example shown in Fig.1, all the providers except the Tier-1 providers allocate a subset of their locators to customers. The locator selection could be determined by customer's traffic engineering or fault tolerance needs. For instance, a provider could assigned two disjointed locators to a customer for fault tolerance purpose. Our future work will focus on the guidelines for locator selections.

B. Challenges of Implementing Hierarchical Location Addressing

As described in our previous work [12], we believe that implementing the proposed addressing scheme has the following two challenges:

1) automatic locator allocation and configuration: we want to automatically allocate locators with minimal human intervention.
2) address space efficiency: we want to use the locator spaces efficiently, and extend the locator spaces with low cost operations. In addition, we want to implement routing table updates with low overhead.

Addressing the above challenges is critical to achieving the scalability of hierarchical location addressing. Since our previous work [9] has proposed a solution to provide an automatic allocation and configuration tool, we focus on the latter issue in this paper.

To overcome the second issue, we should not assume the number of self labels that a provider can allocate to customers. The implementation should not require a reserved space of self labels. Otherwise, if all reserved self labels are used up, inserting a new customer will require extending the spaces. It requires the provider to recompute the locators of all the existing customers and the new customer. In addition, when a customer changes its provider to another one, this change will cause renumbering its locators and locators of its customers.

To implement the proposed hierarchical addressing, we present two locator encoding schemes, which store locator labels as binary numbers, *fixed-length encoding* and *variable-length encoding*. Here, fixed-length representation is used as a baseline method, which is compared with the variable-length representation. In the next section, we present fixed-length encoding and its performance. A variable-length encoding scheme will be introduced in Section IV.

III. Fixed-length Locator Encoding

Fixed-length encoding means that each locator is represented by a fixed-length binary string, such as an IPv4 (32 bits) or IPv6 (128 bit) address. Much research in hierarchical addressing employs fixed-length encoding schemes to encode locators. For instance, previous work [18] uses IPv6 addresses with variable-length subnet masks to represent locators. Our previous work HANA uses IPv4 addresses to represent locators [9].

A. Encoding Approaches

As we described in the previous section, a locator could be considered as a sequence of self labels (integers) separated by delimiter '.'. By using fixed-length locator encoding, each AS can represent self labels in a fixed-length format. However, not only each self label but also every delimiter should be encoded. Otherwise, without the delimiters, different locators might have the same encoded value. For example, suppose that AS A's locator is `1.2.2`, and AS B's locator is `1.10`. Assume that their providers use two bits to represent number `1` (01) and `2` (10). If AS A still uses 2 bits to represent its self label `2`, its locator is encoded as "011010" without the delimiters. If AS B uses 4 bits to represent `10` (1010), its locator is "011010" without the delimiters. So, the two encoded locators have the same value.

There are several ways to explicitly encode the delimiters. For example, a special byte, like UTF8 [19] can be used to represent the delimiter. Another method is to use two bits "00", "10", and "11" to represent '.', '0', and '1' respectively. However, encoding the delimiters is a waste of address space. At the same time, processing the delimiters during packet forwarding can introduce the overhead of processing the packets.

Therefore, a fixed-length encoding scheme must provide a way to implicitly represent the delimiter. There are two ways to encode a fixed-length locator with implicitly represented delimiter:

- **Fixed self label**. This method is to use a fixed number of bits to represent a customer's self labels. If a provider has n customers, the length of each of the self labels must have $\lceil log_2(n) \rceil$ bits. For example, four two-bit binary numbers can represent four customers, or four self labels.
- **Prefix mask**. This method is similar to VLSM (Variable-Length Subnet Mask) employed in today's Internet. Instead of directly representing each self label, a prefix mask is used to indicate the size of self labels [9], [18]. A mask (with $/k$ notation) means that the provider's mask is $/k$ and the customer's mask is $/n$. As a result, the n-bit locator is partitioned into two fixed-length parts: k bits to represent the provider label part and $n-k$ bits to represent the self labels.

The difference between the two methods is that in the latter method self labels may have variable length. In the first method, all self labels allocated by the *same provider* must have the *same* length.

For instance, we use the fixed self label to represent the locators in Fig.1. From the AS relationship dataset in 2015 [16], we know that there are 71 ASes that do not have any providers. We consider them as Tier-1 ASes [1]. We use 7 bits to represent all Tier-1 ASes. AS3356 has 4,148 customers, and AS174 has 4,608 customers. Therefore, the two ASes should use at least 13 bits to represent their self labels ($2^{13} = 8196$). Since AS34056 has 6 customers, its self label should have at least 3 bits. As a result, locator `2.1887.4` is represented

[1]Some of them may belong to the same Tier-1 ISPs.

as: "0000010-0011101011111-100" (dashes are used to make encoded locators more readable). For another example, one of AS34608's locators `1.17.25.4` is encoded as "0000001-0000000010001-00011001-100".

Just like VLSM employed in today's Internet, the second method is much more complicated than the first one because a prefix mask is used to represent a self label and indicate the size of self labels. For example, in Fig.1, AS3356 has 4,148 customers, and most of them have a small number of customers. In this case, those customers may have a prefix mask $/20$, which indicates the maximum number of their descendants, $2^{32-20} = 2^{12} = 4,096$ (assuming a 32-bit address). For these ASes with a large number of customers, they will have a short prefix mask, for example $/10$, which can assign at most $2^{32-10} = 2^{22}$ self labels for their descendants.

B. Efficiency of Fixed-length Encoding

Fixed-length encoding: the issue with the fixed self label encoding is that it uses locator space inefficiently because the size of self labels must be predetermined. It is possible that some locators are under-utilized because of sparse customers, while others are out of space. Based on the measurement results in Section V, we find that a 32-bit address format is insufficient to represent all locators in the Internet.

For example, suppose that all locators in Fig.1 would be presented in 32 bits. As described before, Tier-1 ASes' locators are presented in 7 bits. Since AS3356 has 4,148 customers, the self labels for its customer are encoded into 13 bits. Consequently, each of AS3356's customers has 12 bits left, which can be allocated further to its descendants. Most of AS3356's customers have a small number of descendants. They do not use all the 12 bits to represent their descendants so that their locator space is wasted. There are two customer ASes, AS1273 and AS12389 (indirect customer), having a large number of customers (247 customers and 586 customers, respectively). After allocating 10 bits to represent AS12389's self labels, there are only 2 bit left for AS12389's customers. If a customer of AS12389 has more than 4 direct customers, or 4 descendants, the address space is not large enough to allocate locators. A simple solution is to use a large address space, such as 128-bit IPv6 addresses. However, previous work has shown that using a large address space is still challenging [14]. In addition, using a large address cannot resolve the inefficiency issue because most of the bits are not used.

Prefix mask encoding: the issue with the prefix mask encoding is the complexity of developing a locator encoding plan. Prefix mask encoding is designed to sacrifice the simplicity of routing for address space efficiency. For example, we can apply the prefix mask encoding to solve the insufficient space issue caused by the fixed self label encoding. In Fig.1, AS3356 assigns different prefix masks to its 4,148 customers. The customers having a small number of descendants are assigned locators with the form "0000001-0xxxxxxxxxxxx/20", where 'x' represents either '1' or '0'. This prefix mask can represent $2^{12} = 4096$ customers, and each of them can allocate 2^{12} locators. For other customers having a large number of descendants, a shorter prefix mask is assigned to each of them: "0000001-1xxxxxx/14". AS3356 can assign this shorter prefix mask to 64 (2^6) ASes. For example, suppose that AS1273 has "0000001-1111111/14". Next, AS1273 applies the same method to assign a short prefix mask to AS12389, for example, "0000001-11111111-100/17". AS12389 assigns "0000001-1111111-100-100/20" to AS12883. Finally, we see that the customers or descendants of AS12883 can be presented by 32-bit numbers. However, this method does not scale well because it is very difficult to maintain and manage the prefix masks, which are subject to human error. Furthermore, it is difficult to dynamically assign locators and troubleshoot address assignment issues.

Another issue with the two encoding methods is the high-cost locator renumbering. This is because the size of self labels is preassigned to accommodate the expected number of customers. For example, in Fig.1, AS28917 has 128 customers so that it uses 7 bits to present the self labels. Let's assume that there is a new customer. In order to accommodate this new AS, AS28917 has to enlarge the size of its self labels to 8 bits. Before assigning a new locator, AS28917 must renumber all the locators that are already assigned. Furthermore, this change might cause other customers to renumber their descendants.

Hence, there is is a need for a simpler and more automatic allocation of locators. In the next section, we introduce a variable-length encoding scheme to achieve this goal.

IV. Variable-length Locator Encoding

The idea of using a variable-length representation is to assign each self label a variable-length number. Instead of encoding the entire locator, each AS locally encodes its own self labels into locally unique binary numbers. The size of each self label is essentially determined by a specific variable-length encoding method.

Since a provider's locator labels will appear in its descendants' locators, this implies that Tier-1 ASes' self labels will appear more frequently than that of other ASes. Therefore, the occurrence of self labels is not uniformly distributed, which motivates us to design an efficient encoding method to encode the more frequent labels with shorter codewords, which would potentially reduce the expected length of self labels. In addition, by using a variable-length encoding method, an infinite number of integers can be used to represent self labels. That is, it does not have a restriction on the size of self labels.

A. Self Label Distribution

To determine an appropriate encoding method, we need to understand the frequency distribution of self labels in today's Internet. The frequency is based on CAIDA dataset [16]). We first use the AS relationships from the dataset to build an AS level graph. Then, we use the labeling method described in Section II to assign self labels to the ASes in the graph. At last, we calculate the occurrence of self labels. Note that the locators have not been encoded into binary numbers.

In Fig. 2 x axis represents the value of self labels, and y axis represents the frequency of occurrence (in percentage) of

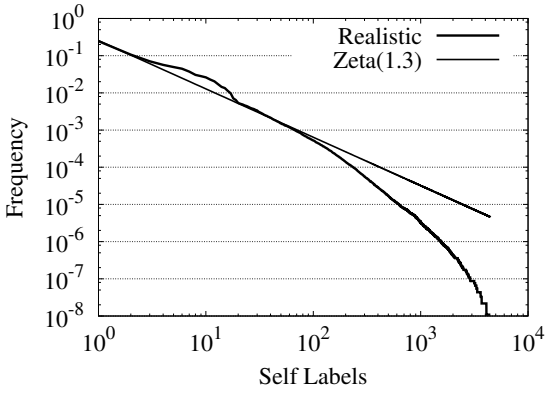


Fig. 2. The frequency of occurrence of self labels in the Internet.

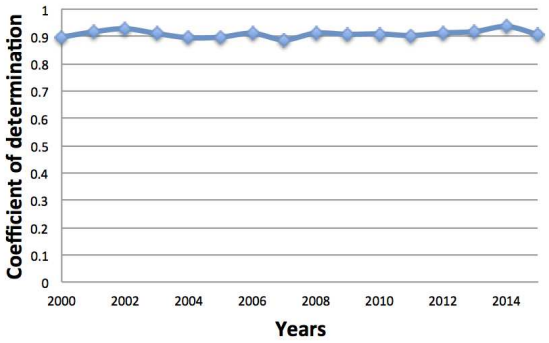


Fig. 3. The coefficient of determination of the frequency of occurrence of self labels.

each self label. The figure shows that the frequency follows Zeta distribution, which is a discrete version of power-law distribution [20]. In the figure, we use the probability mass function of Zeta distribution with $\alpha = 1.3$ as the regression function to fit the distribution of frequency:

$$Z_\alpha[x] = \frac{1/x^\alpha}{\sum_{i=1}^{\infty} \frac{1}{i^\alpha}},$$

Thus, the frequency distribution of self labels can be described as a power law distribution, and the frequency of the xth self label is $1/x^\alpha$.

Furthermore, based on the CAIDA datasets, we calculate the coefficient of determination to understand how well the frequency fit the Zeta distribution with $\alpha = 1.3$ over 10 years (from 2005 to 2015). The result is shown in Fig. 3. The figure shows that the value of coefficient of determination varies between 0.96 and 0.98. That means, the frequency distribution of self labels keeps following the same Zeta distribution through years.

B. Prefix Codes

In general, a prefix code is used to encode self labels. A prefix code is a code system that there is no codeword is a prefix of any other codeword in the system. For example, codewords "0110" and "10" are "prefix-free" codewords. Codewords "0110" and "011" are not because "011" is a prefix of "0110". As a result, without explicitly encoding the delimiters, a prefix code can uniquely encode locators and does not cause ambiguity in the binary representation.

TABLE I
AN EXAMPLE OF BAER CODE 0 CODEWORD TABLES

Values	Baer0	Values	Baer0	Values	Baer
1	000	6	0111	11	101000
2	001	7	10000	12	101001
3	0100	8	10001	13	101010
4	0101	9	10010	14	101011
5	0110	10	10011	15	101100

There are many variable-length prefix codes that could be used to encode self labels. One of the best known variable-length codes is Huffman Coding. Huffman Coding is the minimum redundancy prefix-free codeword. However, Huffman code requires precisely known probability distributions of symbols. In other words, the actual distribution of the self labels must be known. Otherwise, it cannot produce an optimal code to match the distribution. Hence, Huffman code is not feasible because we may not know the actual probability distribution of self labels before we can allocate them.

C. Universal Codes

Universal codes [21] are particularly suitable for our purpose because each universal code assumes an implied probability distribution. The implied distribution could be an estimation of Zeta distribution as we described above. In general, a universal code encodes a set of self labels as follows. Given positive integers $1, 2, \cdots, n$ and a set of probabilities $p_1, p_2, \cdots, p_n$. A universal-code is a prefix-code $c_1, c_2, \cdots, c_n$ that satisfies two conditions: 1) i is mapped to c_i for every $i = 1, \cdots, n$, and 2) if the probability distribution is monotonic, $p_i \geq p_{i+1}$ where $1 \leq i \leq n$, then there is a constant, k, such that the optimal-code $c_1^o, \cdots, c_n^o$ with $p_1, \cdots, p_n$ satisfies $\frac{|c_i|}{|c_i^o|} \leq k$ where $1 \leq i \leq n$ and $k \geq 1$.

Self labels can be encoded by universal code. If we know that self label $self_1$ has the highest probability and $self_2$ has the next largest one, we can assign the shortest codeword to $self_1$ and the next longer codeword to $self_2$. In other words, the self label with index 1 implies the largest probability, the self label with index 2 implies the next highest one, and so on.

There are several possible universal codes for self labels: Elias' γ, Elias' δ, Elias' ω codes [22], Fibonacci code [23], Baer code [24], Exponential-Golomb code [25], and Zeta code [26]. Table I shows part of Baer0 codewords. Table II shows the length of three Elias Codes [27] and Baer code. From the table, we see that Elias' δ and Elias' ω code might generate shorter codes than others if providers have a large number of customers. Baer code will generate shortest locators than others if providers have a small number of customers.

Next, we use Baer code as an example to illustrate how to encode self labels. Note that we do not claim that this code

TABLE II
LENGTHS OF THREE ELIAS CODES AND BAER CODE.

Values	Elias's γ	Elias's δ	Elias's ω	Baer
1	1	1	2	2
2	3	4	3	3
3	3	4	4	4
4	5	5	4	4
5-7	5	5	5	4-5
8-15	7	8	6-7	5-7
16-31	9	9	7-8	7-9
32-63	11	10	8-10	9-11
64-88	13	11	10	11-12
10^2	13	11	11	12
10^3	19	16	16	16
10^4	27	20	20	25
10^5	33	25	25	32

is the optimal code for self labels. The Baer code is defined as [24]:

$$c_0(i) = \begin{cases} 00, & i = 1, \\ 010, & i = 2, \\ 011, & i = 3, \\ 1c_0(\frac{i-2}{2})0, & i = 4, 6, 8, ..., \\ 1c_0(\frac{i-3}{2})1, & i = 5, 7, 9, ... \end{cases}$$

The term $c_0(i)$ denotes the Baer code of the ith codeword. For example, $c_0(17) = 1c_0(7)1 = 11c_0(2)11 = 1101011$. Compared to other codes, Baer code is computationally efficient. Table I shows some part of Baer codewords.

For example, we use Baer code to encode the locators shown in Fig.1. Based on each AS's size, which is determined by the number of descendants, we show the encoded locators in Fig. 4. From this figure, we see that the maximum length of AS34608's locators is 26 bits, which can be packed in an IPv4 address.

Finally, we present the following steps for each provider to encode its self labels:

- **Step 1**: each provider first estimates the implied probability distribution of its self labels. Based on the implied probability distribution, the provider chooses an encoding method. For example, Elias' γ code can be used when the self labels distribution follows power laws of the exponent close to two. Elias' δ code is used when the exponent is close to one [28]. One of Baer codewords [24] is used when the exponent is around 1.2. A universal code will be the optimal code when the distribution of the self label frequency is exactly the same as the implied probability distribution.
- **Step 2**: each provider needs to know the size of its direct customers. The size can be determined at different granularity level. Providers can simply determine the size by using the number of descendants of each customer. The number could be at AS level or host level.
- **Step 3**: each provider sorts the size of its direct customers in non-increasing order. According to the selected coding method, encoded self labels are assigned to its customers.

As we mentioned before, a more detailed description of selecting an optimal universal code for self labels is out-of-range of this paper. Since we are interested to know the feasibility and benefits of variable-length locators, we do not intend to present an optimal prefix code for self labels.

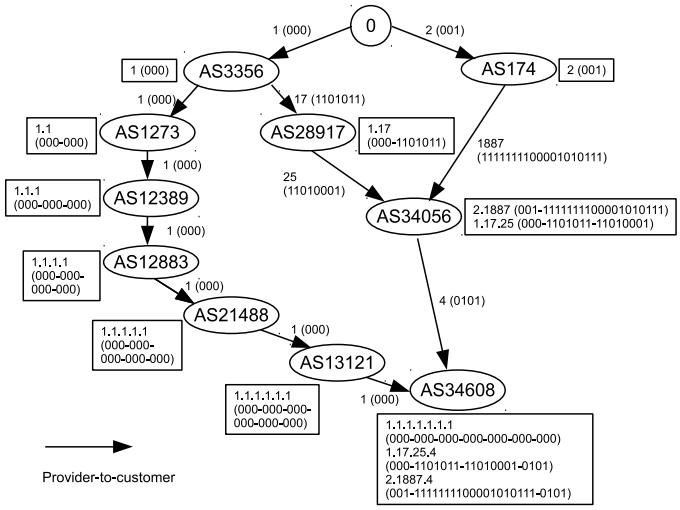


Fig. 4. An example of Baer encoded locators. The number in parentheses represents an encoded locator or self label. Dashes are used to make encoded locators more readable.

D. Flexible Locators Encoding

The variable-length encoding method provides the flexibility for providers to select their own universal codes. Providers are not forced to use the same encoding scheme. Each provider could employ their own encoding schemes so that different variable-length prefix codes could be used to encode locators. For example, a provider may use Elias' δ to encode its self labels, while its customers may use Baer code. Even though a locator might be encoded by different universal codes, it is still an unique locator. Note that to maintain this flexibility, Tier-1 ASes' locators must be prefix-free if they are encoded by different encoding schemes. This is not difficult to implement because a centralized authority, such as IANA, can guarantee the uniqueness of the locators.

Furthermore, there is no need to decode the variable-length self labels. That means, the data plane still uses the variable-length locators to make forwarding decisions.

E. Renumbering Variable-length Locators

Because there is no length restriction, the variable-length encoding method can extend the locator address space without renumbering locators. To accommodate a new AS, a provider can locally assign and encode a new self label for the AS. The provider and other ASes do not need to change the their locators. To delete an AS, a provider does not need to delete the self label, just simply remove the association between the self label and the AS.

Most importantly, as the network or the AS relationship changes, it is useful to have an inexpensive way to renumber locators. The proposed variable-length encoded method should reduce the number of messages exchanged when renumbering locators. To achieve this goal, we can separate the distribution

of a locator into two separated processes: *renumbering encoded provider labels* and *renumbering encoded self labels*. That means, provider labels and self labels could be propagated to a customer jointly or separately.

Renumbering encoded provider label: When a provider's locator label is changed, the new locator label must be distributed to its descendants. All of its descendants are impacted by this renumbering. However, the existing self labels should not be affected by the locator label change. Therefore, whenever a locator of a provider at the upper level is changed for renumbering, the customers at the lower level only need to change their provider labels.

Renumbering encoded self labels: When the size of a customer is changed, a new self label should be assigned to the customer. However, only the customer and its descendants should renumber their locators. Other customers do not need to change. In addition, we can avoid renumbering self labels by exploiting the *staircase* property of universal codes. That is, consecutive codewords may have the same length, for example, the codewords in Table I. Before replacing a self label with a new one, the new label should be compared with the previous one. If the new one has the same length as the previous one, the new self label will be suppressed. Therefore, self label renumbering is the result of a change in the length of a self label, instead of its size.

V. Performance Evaluation

In this section, we first investigate if today's Internet can be represented in a 32-bit fixed-length locator. Second, we compare the performance of the fixed-length encoding and variable-length encoding methods in terms of locator length. Finally, we attempt to find a universal code that can represent the locators in today's Internet in the minimum number of bits.

A. Fixed Length Encoding

We investigate if a 32-bit locator is sufficient to encode all locators in the Internet in fixed-length encoding. We use the AS relationships dataset [16] from CAIDA to derive an AS level graph, which can be used to represent today's Internet hierarchy. We use the AS relationship dataset from 2015, and divide the graph into several layers. We consider the worst case, in which the AS at each layer has the maximum number of customers. First, we determine the self labels of Tier-1 ASes. In the dataset, there are 71 ASes that do not have any providers, and we consider them as Tier-1 ASes. 7 bits are used to represent the self labels of the Tier-1 ASes. Second, we determine the self labels of Tier-2 ASes. To determine the self labels, we consider the worst case, in which a Tier-1 AS has the maximum number of customers. In the dataset, one of the Tier-1 ASes has 4,461 customers. Therefore, at least 13 bits ($2^{13} = 8196$) must be used to represent those customers. Assume that one of the customers is a Tier-2 AS. The provider label of the Tier-2 AS has 20 bits, and only 12 bits are left. Third, we determine the size of self labels for Tier-3 ASes. According the dataset, we find that a Tier-2 AS can have more than 859 customer. As a result, the Tier-2 AS has to use 10 bits for its self labels. So far, totally 30 bits are used to represent a Tier-3 AS. Unfortunately, the Tier-3 AS has only 2 bits left so that it can have at most four customers. From the dataset, we find that a Tier-3 AS may have more than 130 customers, and today's Internet can have more than 5 layers.

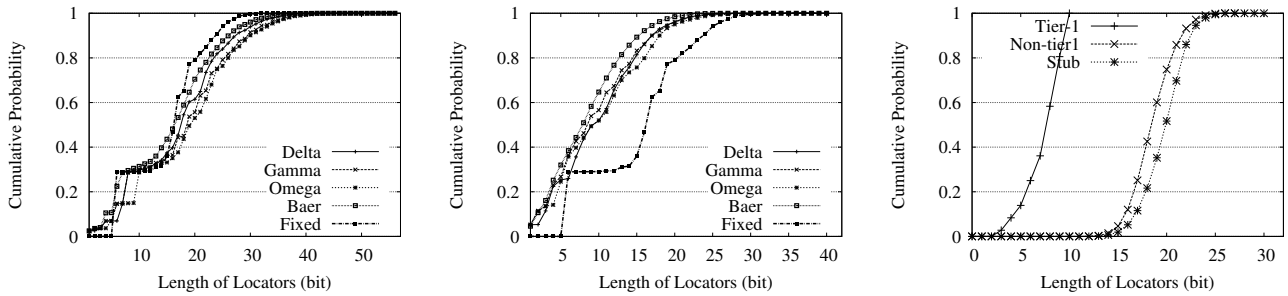
Thus, we believe that a fixed 32-bit locator cannot represent all locators in today's Internet. Furthermore, From the analysis, we find that the size of locators is impacted not only by the number of direct customers, but also by the number layers in the Internet hierarchy.

B. Variable Length Encoding

Based on the AS graph, we assign locators to each AS. With a given encoding method, we assign variable length numbers to self labels in two ways: random assignment and ranking-based assignment. Random assignment means that we randomly assign a number to a self label, and encode the number using one of the codes. Ranking-based assignment means that we assign a self label according to the rank/size of a customer. Because we are interested in keeping a scalable routing table on the control plane, the size is determined by the number of descendants of each customer (on AS level). Then, we sort the self labels according to their values in non-increasing order. Based on its order, a codeword is assigned to each self label.

We present the results in Fig. 5. We compare the length of encoded locators with that of fixed-length locators. In fixed length encoding, we use the total number of customers to determine the number of bits for self labels. From Fig. 5(a), we find that the locator length will be almost the same if we randomly select a codeword. The length distribution is also similar to that of fixed-length locators. We find that in the worst case, the locators encoded by the four prefix codes can be longer than the fixed-length method. However, as shown in Fig. 5(b), if we employ ranking-based assignment, the locators encoded by Elias' δ and Baer codes are always shorter than the others, including fixed-length method. In Fig. 5(c), we redraw the length distribution of locators encoded by Baer code. It shows the length at different hierarchical layers. We can see locators at different layers might have different length distribution, and the majority of the locators are 10 to 25 bits long. In addition, the average length of variable-length locators is about 19 bits, while the average length of the fixed-length locators is more than 30 bits. Hence, the variable-length locators are much shorter than fixed-length locators.

After we know the length distribution of locators, we continue to examine the length distribution of self labels. Here, we focus on using the variable-length encoding methods: Huffman code, three Elias' codes and three Baer codes, to encode self labels. We calculate the distribution of locators encoded by those methods. The distribution implies the best encoding method, which can generate the shortest self labels. To derive the best encoding method, we compare different universal codes with Huffman code. The length distribution of the best encoding method should be the closest one to that of the Huffman encoding method. From Table III, we see that



(a) Random Assignment (b) Ranking-based Assignment (c) Baer Code based Assignment

Fig. 5. Length distribution of locators encoded in different codes.

among all candidate codes Baer0 code (in bold font) always generates the shortest encoded self labels over those years.

VI. Related Work

The most recent network outage shows that routing table size can impact forwarding performance [29]. BGP tables have been increasing much faster than a linear rate for more than a decade [30]–[32], and BGP entries approached 500,000 in 2013 [33]. Measurement studies in [30], [34] pointed out that multi-homing, IP address fragmentation and load balancing are the major contributors to BGP table growth. Realizing the exponential growth of forwarding tables, a flurry of research studies have focused on designing new architectures based on the idea of Locator/ID separation [1]–[6].

Another direction to address the scalability problem of multi-homing is to employ prefix aggregation [9], [11], [35]–[37]. It proposes a hierarchical addressing structure and forces every AS to inherit one sub-prefix from each of its providers. For example, in HANA [9], a locator consists of three parts: prefix, midfix, and suffix.

Others have focused on designing new architectures based on the idea of Locator/ID separation to solve the scalability issue of the current routing system [1]–[6], [10], [11], [38], [39]. For example, in LISP [1], the IP address serves as Endpoint Identifier (EID), which is only routable inside a LISP-domain. Routing Locators (RLOCs) are the public IP-addresses of the border routers of a LISP domain, and are globally routable. The Host Identity Protocol (HIP) [5] decouples the identity and the location by using the Host Identity Tag (HIT) as the identifier and the IP address as the locator. HIT allows two end hosts to establish a secure IP layer session. SHIM6 [2] is proposed to support multi-homing in IPv6 networks. In SHIM6, one of available IP addresses is chosen as the upper layer ID (ULID), which is used for session identification. The remaining addresses are considered as locators. Identifier Locator Network Protocol (ILNP) [3], [4] implements the locator/identifier separation by employing address rewritting. Mobility and Multihoming support Identifier Locator Split Architecture (MILSA) [6], [10] implements the locator/identifier split at different hierarchies in the network. RANGI [40] implements the locator and identifier separation by introducing a host identifier layer between network and transport layers. The host identifier has an organizational structure, and the locators are based on IPv4 addresses. Heterogeneity Inclusion and Mobility (HIMALIS) [7], [38], [41] implements locator and identifier separation by introducing an Identity Sublayer between network and transport layers. HIMALIS can use any kind of addressing scheme for locators and supports security features based on asymmetric keys.

Hierarchical addressing has been proposed to resolve the scalability problems of the Internet. In [42], Leonard Kleinrock et al. evaluate the performance of hierarchical addressing for large networks. Their results show the tradeoff between the routing table size and network path length. In NIRA [18], hierarchical addressing is used to enable end users to choose their own routes. Several previous works have focused on the performance of hierarchical addressing. HANA [9], [11], [35] solves the scalability problem of multi-homing by using prefix aggregation. It enforces every AS to inherit one sub-prefix from each of its providers. In consequence, a multi-homed AS possesses multiple prefixes, and each host in the multi-homed AS has multiple IP addresses. Work [43] measures the efficiency of hierarchical addressing at the inter-domain level. Work [17] investigates the impact of prefix assignment and advertisement on the scalability of hierarchical addressing.

VII. Conclusion

In this paper, we propose a variable-length addressing encoding scheme to mitigate the scalability issue of the current Internet. We compare the performance of fixed-length encoding and variable-length encoding by understanding the degree to which the Internet can benefit from hierarchical addressing. Based on the real routing data, our study shows that it is difficult to use fixed-length encoding to implement hierarchical location addresses for today's Internet. Furthermore, fixed-length encoding uses address space inefficiently. Our analysis and evaluation results show that variable-length encoding could resolve both the scalability problem and the inefficiency of address spaces.

Acknowledgment

This work was supported by National Science Foundation grant CNS-1402857 and CNS-1402594 and under NSF-NICT Collaborative Research JUNO (Japan-U.S. Network Opportunity) Program.

TABLE III
THE LENGTH OF SELF LABELS ENCODED BY DIFFERENT ENCODING METHODS

	Avg.	2015	2014	2013	2012	2011	2010	2009	2008	2007	2006	2005
Huffman codes	*4.79*	*4.97*	*4.88*	*4.95*	*4.86*	*4.82*	*4.71*	*4.76*	*4.77*	*4.61*	*4.69*	*4.68*
δ codes	5.37	5.57	5.44	5.56	5.47	5.43	5.27	5.35	5.36	5.19	5.24	5.22
γ codes	5.19	5.41	5.29	5.40	5.29	5.24	5.07	5.16	5.17	4.96	5.06	5.04
ω codes	5.57	5.78	5.64	5.78	5.67	5.63	5.43	5.53	5.54	5.35	5.46	5.43
Baer0 codes	**4.82**	**5.02**	**4.92**	**5.00**	**4.90**	**4.85**	**4.73**	**4.79**	**4.79**	**4.62**	**4.71**	**4.69**
Baer1 codes	4.90	5.06	5.00	5.04	4.96	4.92	4.84	4.88	4.88	4.73	4.81	4.80
Baer2 codes	5.20	5.33	5.28	5.31	5.24	5.20	5.15	5.18	5.18	5.06	5.15	5.13

REFERENCES

[1] L. Iannone, D. Saucez, and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning," *RFC*, January 2013.

[2] A. García-Martínez, M. Bagnulo, and I. Van Beijnum, "The Shim6 Architecture for IPv6 Multihoming," *Comm. Mag.*, vol. 48, pp. 152–157, September 2010.

[3] L. Iannone, D. Saucez, and O. Bonaventure, "Implementing the Locator/ID Separation Protocol: Design and experience," *Comput. Netw.*, vol. 55, pp. 948–958, March 2011.

[4] R. Atkinson, S. Bhatti, and U. S. Andrews, "Identifier-Locator Network Protocol (ILNP) Architectural Description," *RFC*, 2012.

[5] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture," *RFC*, May 2006.

[6] J. Pan, S. Paul, R. Jain, and M. Bowman, "MILSA: A Mobility and Multihoming Supporting Identifier Locator Split Architecture for Naming in the Next Generation Internet," in *GLOBECOM*, pp. 2264–2269, 2008.

[7] V. P. Kafle, H. Otsuki, and M. Inoue, "An ID/locator Split Architecture for Future Networks," *Comm. Mag.*, vol. 48, pp. 138–144, February 2010.

[8] L. Kleinrock and F. Kamoun, "Hierarchical Routing for Large Networks, Performance Evaluation and Optimization," *Computer Networks*, vol. 1, pp. 155–174, January 1977.

[9] K. Fujikawa, H. Tazaki, and H. Harai, "Inter-AS Locator Allocation of Hierarchical Automatic Number Allocation in a 10,000-AS Network," in *SAINT 2012*, 2012.

[10] J. Pan, R. Jain, S. Paul, M. Bowman, X. Xu, and S. Chen, "Enhanced MILSA Architecture for Naming, Addressing, Routing and Security Issues in the next Generation Internet," in *Proceedings of the 2009 IEEE international conference on Communications*, ICC'09, pp. 2168–2173, 2009.

[11] K. Fujikawa, K. Ohira, and M. Ohta, "A Hierarchical Automatic Address Allocation Method Considering End-to-end Multihoming," in *IEICE Tech Rep*, vol. 109, 2009. IA2009-56.

[12] K. Fujikawa, A. H. A. Muktadir, Y. Fukushima, H. Harai, X. Shao, F. Wang, and L. Gao, "Design and Implementation of Variable-Length Locator Allocation Protocol for Scalable Internet Addressing," in *APCC*, October 2015.

[13] R. Gummadi and R. Govindan, "Practical Routing-Layer Support for Scalable Multihoming," in *in IEEE Infocom on Computer Communications*, 2005.

[14] P. Tsuchiya, "Efficient and flexible hierarchical address assignment," in *INET92*, pp. 441–450, 1992.

[15] F. Wang, L. Gao, S. Xiaozhe, H. Harai, and K. Fujikawa, "Compact Location Encodings for Scalable Internet Routing," in *IEEE INFOCOM*, 2015.

[16] CAIDA, "The IPv4 Routed /24 AS Links Dataset." http://www.caida.org/data/active/ipv4_routed_topology_aslinks_dataset.xml.

[17] X. Lu, W. Wang, X. Gong, X. Que, and B. Wang, "Empirical Analysis of Different Hierarchical Addressing Deployments," in *Communications (APCC), 2013 19th Asia-Pacific Conference on*, pp. 208–213, Aug 2013.

[18] X. Yang, D. Clark, and A. Berger, "NIRA: A New Inter-Domain Routing Architecture," *Networking, IEEE/ACM Transactions on*, vol. 15, pp. 775–788, Aug 2007.

[19] F.Yergeau, " UTF8: A Transformation Format of ISO 10646." Request for Comments (RFC) 2279, 2003.

[20] S. Shakkottai, M. Fomenkov, R. Koga, D. Krioukov, and K. Claffy, *Evolution of the Internet AS-Level Ecosystem*, vol. 5. Springer Berlin Heidelberg, 2009.

[21] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[22] P. Elias, "Universal Codeword Sets and Representations of the Integers," *IEEE Transactions on Information Theory*, vol. 21, pp. 194–203, Mar 1975.

[23] A. S. Fraenkel and S. T. Klein, "Robust Universal Complete Codes for Transmission and Compression," *Discrete Applied Mathematics*, vol. 64, pp. 31–55, 1996.

[24] M. Baer, "Prefix Codes for Power Laws," in *ISIT 2008. IEEE International Symposium on Information Theory*, pp. 2464–2468, July 2008.

[25] I. Richardson, *The H.264 Advanced Video Compression Standard*. Wiley, 2010.

[26] P. Boldi and S. Vigna, "Codes for the World Wide Web," *Internet Mathematics*, vol. 2, no. 4, pp. 407–429, 2005.

[27] D. Salomon, *Variable-length Codes for Data Compression*. 2007.

[28] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[29] M. Palladino, "Growing Pains of the Internet Global Routing Table." http://www.internap.com/2014/08/18/growing-pains-internet-global-routing-table/, Auguest.

[30] T. Bu, L. Gao, and D. Towsley, "On Characterizing BGP Routing Table Growth," in *In Proceedings of IEEE Global Internet Symposium*, 2004.

[31] G. Huston, "Analyzing the Internet's BGP Routing Table," *The Internet Protocol Journal*, vol. 4, March 2001.

[32] L. Budzisz, R. Ferrús, A. Brunstrom, K. J. Grinnemo, R. Fracchia, G. Galante, and F. Casadevall, "Towards Transport-layer Mobility: Evolution of SCTP Multihoming," *Comput. Commun.*, vol. 31, pp. 980–998, March 2008.

[33] "BGP table size." http:bgp.potaroo.netas2.0bgp-active.html.

[34] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing." RFC 4984 (Informational), September 2007.

[35] K. Fujikawa, M. Ohta, and H. Harai, "The Basic Procedures of Hierarchical Automatic Locator Number Allocation Protocol HANA," in *AINTEC'11*, pp. 124–131, 2011.

[36] D. Saucez, J. Kim, L. Iannone, O. Bonaventure, and C. Filsfils, "A Local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers," IFIP'12, pp. 397–408, 2012.

[37] Y. Song, L. Gao, and K. Fujikawa, "Resilient Routing under Hierarchical Automatic Addressing," in *GLOBECOM'11*, pp. 1–5, 2011.

[38] K. V. P. and M. Inoue, "ID/Locator Split-Based Mobility Scheme for Heterogeneous New Generation Network," *Wireless Personal Communications*, vol. 61, no. 4, pp. 753–764, 2011.

[39] A. Ford, C. Raiciu, S. Handley, M.and Barre, and J. Iyengar, "Architectural Guidelines for Multipath TCP Development,"

[40] T. Li, "Recommendation for a Routing Architecture," 2011.

[41] R. Atkinson, S. Bhatti, and S. Hailes, "Implementation and Evaluation of ID/Locator Split-based HIMALIS Network Protocol Stack," in *IEICE Tech. Rep., vol. 112, no. 392, NS2012-153, pp. 69-74, Jan. 2013*.

[42] L. Kleinrock and F. Kamoun, "Hierarchical Routing for Large Networks, Performance Evaluation and Optimization," *Computer Networks*, vol. 1, pp. 155–174, January 1977.

[43] Y. Zhuang and K. L. Calvert, "Measuring the Effectiveness of Hierarchical Address Assignment," in *GLOBECOM*, pp. 1–6, 2010.