VLI: Variable-Length Identifier for Interconnecting Heterogeneous IoT Networks

Gang Liu, Wei Quan, Nan Cheng, Hongke Zhang, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—Long identifier brings low packet forwarding efficiency in Internet of Things (IoT), whereas short identifier may suffer from the exhaustion of identifier space. Compared with fixed-length identifiers (*e.g.*, IPv4 and IPv6), flexible identifiers are expected for *balancing* the packet processing efficiency with the various IoT scales. However, it is challenging to make IoT support the flexible identifier-based forwarding. In this letter, we firstly proposed a novel variable-length identifier (VLI) solution for interconnecting IoT networks. In particular, a VLI datagram header is designed to effectively support a flexible identifier field. Following a basic VLI header, one (multiple) extension header(s) can be added if require. Each extension header includes a fixed-size identifier field. According to the combination of multiple identifier fields, the variable-length identifier can be easily achieved, resolved and supported by the IoT nodes in a flexible way. Experimental results show that VLI can decrease the processing delay effectively.**

***Index Terms*—Flexible identifiers, VLI, processing delay, IoT.**

I. Introduction

INTERNET of Things (IoT) has driven billions of devices connected to the Internet. Along with this growth, more and more IoT nodes can "*talk*" with each other via various existing or new network identifiers. As the most classic identifying system, IPv4 address has been widely applied to identifying each IoT device [1]. Thubert and Cragie presented using IPv6 identifiers over IoT devices by IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [2]. Besides, the 16-bit ZigBee identifier was recommended for wireless dynamic sensor network (WDSN) [3]. However, these identifying systems are with separated address space and protocols, which bring big troubles in compatibility and interoperability.

On the other hand, the aforementioned fixed-length identifiers have drawbacks to meet different requirements in heterogeneous IoT networks. A short identifer is insufficient for vast numbers of devices in large-scale IoT [4]. For example, IPv4 addresses are just 32 bits long, which are inherently limited to 4.3 billion unique combinations before becoming exhausted [5]. Although the Network Address Translation (NAT) technology can relieve this tense situation, it is still challenging to cope with the exponentially increasing IoT devices. IPv6 addresses have a larger identifier space. However, it is wasteful if we use a long identifier for small-scale IoT networks, *e.g.*, 128-bit is too long for identifying only 100 IoT nodes. What is worse, a long identifier would lower packet forwarding efficiency and waste bandwidth. *Therefore, why not to finish an efficient variable-length identifier to adapt to various IoT scenarios and be compatible with each other?*

Flexible identifiers are expected to accommodate different demands in the booming area of IoT. For example, it can balance the packet processing efficiency with the IoT scale. If a light packet header is required in IoT cases [6]–[8], a flexible identifier can transfer to a *short* form itself, which can guarantee the high packet process efficiency. On the contrary, a *long* form can be transferred for a large-scale global network [9]. This also resolves both the scalability problem and the inefficiency of address spaces [10]. In fact, the view of variable length addresses was once discussed in early 1990s, but did not achieve unanimity then [11]. Our previous work also proposed to use the smart identifier mapping to be compatible with heterogeneous networking paradigms [12]. Recently, Ren *et al.* presented a hierarchical structure of flexible address system with an infinite address space [13]. However, there is still many practical issues to consider before deploying flexible identifiers [14]. *One key issue is how to make IoT devices communicate with each other using flexible identifiers and how to design the flexible identifier based forwarding.*

This letter firstly proposes a complete Variable-Length Identifier (VLI) solution, includes the recommended packet header and forwarding mechanism, for interconnecting various-scale IoT networks. Particularly, we focus on designing a novel datagram header (in network layer) to support a flexible field of identifier. A complete VLI identifer is dependent with both basic header and extension header. Each extension header has an identifier field with the fixed size, which can combine to get variable-length identifier. In this way, IoT devices could easily implement the flexible field without updating their fixed firmware. The main contributions are summarized as follows:

- First, we propose a novel Variable-Length Identifier for interconnecting IoT networks. The VLI header and its extension header are also introduced to support a flexible identifier based forwarding.
- Second, we formulate and analyze the VLI extension-header-size optimization problem, and introduce how to splice extension headers.

Manuscript received December 3, 2019; accepted March 8, 2020. Date of publication March 26, 2020; date of current version August 7, 2020. This work was supported in part by the National Key Research and Development Program under Grant 2018YFE0206800 and Grant 2017YFE0121300, and in part by the Fundamental Research Funds for the Central Universities of China under Grant 2019JBM011 and Grant 2019YJS014. The associate editor coordinating the review of this article and approving it for publication was K. Ota. *(Corresponding author: Wei Quan.)*

Gang Liu, Wei Quan, and Hongke Zhang are with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: weiquan@bjtu.edu.cn).

Nan Cheng is with the State Key Laboratory of ISN, Xidian University, Xi'an 710071, China, and also with the School of Telecommunications Engineering, Xidian University, Xi'an 710071, China.

Xuemin Shen is with the Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L3G1, Canada.

Digital Object Identifier 10.1109/LWC.2020.2982641

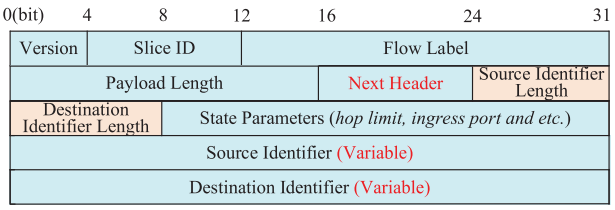


Fig. 1. The VLI datagram header.

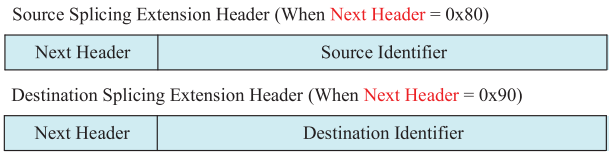


Fig. 2. The VLI datagram extension headers.

- Third, we implement the VLI solution through P4[1] language and share the source codes. Experimental results show VLI decreases the packet processing time by 10.4%.

The remainder of this letter is organized as follows. Section II introduces the VLI header and analyzes an optimal header size. Section III evaluates the performance. Finally, Section IV concludes this letter.

II. VLI Solution

A. VLI Header and Extension Header

This section designs a set of VLI header and extension header to effectively support a flexible field of identifier. Why do we choose this structure? In terms of engineering practice, there is no variable to directly store a variable-length content. Therefore, it is not practical to use a variable for VLI. Motivated by the railway carriages, a variable-length identifier can be comprised with a number of fix-length sub-identifiers. This design can be easily implemented and supported in the resource constrained IoT devices.

Fig. 1 shows a sample of VLI datagram header, in which a Source-Identifier-Length (SIL) field is used to indicate the length of source identifier, a Destination-Identifier-Length (DIL) field is for the length of destination identifier, and a Next-Header (NH) field is used to support the VLI extension header. Source/Destination-Identifier fields store the basic identifiers. As an example, the VLI basic header defines a 32-bit identifier. It is worth noting that the length also can be 8-bit, 16-bit or with other longer size. Fig. 2 depicts the VLI extension header which is followed one by one for a specific length of identifier. The size of each VLI extension header will be introduced and analyzed in the next subsection. Aside from these new features, the state parameters are created for recording state parameters, *e.g.*, a hop limit of a packet, an ingress/egress interface of the packet, and a timestamp of processing the packet. Similar with the IPv6 protocol, the Version field indicates the VLI protocol version, the Slice-ID field carries the information about the types of the traffics, and the Payload-Length field indicates the size of the payload in octets. Moreover, the Flow-Label field is created for better supporting real-time application services. It can also be used to detect spoofed packets.

[1]P4: https://p4.org/.

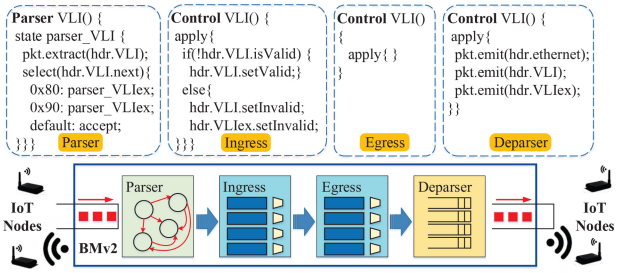


Fig. 3. A simple VLI forwarding process.

B. Extension-Header-Size Optimization Problem

A flexible identifier is realized by following multiple VLI extension headers, the size of which is an open problem. On one hand, if the size is too large, it introduces many redundant bits (*e.g.*, an 8-bit identifier is required, but the extension-header-size is 32-bit), which causes a degraded efficiency. On the other hand, if the size is too small, then many extension headers are followed one by one for a specific length of identifier, which brings long processing delay. Therefore, we can formulate the VLI extension-header-size problem and analyze the optimal solution.

We first model the process of packet forwarding by the P4 software switch (Behavioral Model, BMv2), which is programmable and protocol-independent for easily supporting various novel network protocols. Typically, the switch forwarding process, which is shown in Fig. 3, can be divided into four parts: parser, ingress, egress and deparser. In the parser period, the packet headers (including basic header and extension headers) are parsed, which takes variable time due to different number and size of the extension headers. We denote the parser processing time as $\varphi(x, y, i)$, whereas x is the extension header size, y is the number of extension headers, and i is the instruction cycle time. Assuming a network requires z bits identifiers, the relationship between x, y, z can be denoted as $xy \leq z < x(y+1)$. In the ingress period, the switch assembles each extension header using the Next-Header field in the VLI header, and its processing time can be denoted by $\eta(x, y, i)$. The switch does nothing for the extension headers in the egress period, and the processing time is αi. Moreover, in the deparser period, the switch takes $\gamma(x, y, i)$ time to serialize the basic headers, extension headers and the payload. Therefore, the total time of forwarding a packet in the switch can be denoted by:

$$d_x = \varphi(x, y, i) + \eta(x, y, i) + \gamma(x, y, i) + \alpha i \tag{1}$$

In our designed VLI, the maximum identifier size is 256 bits and the basic header has a 32-bit identifier. Supposing that the number of packets within networks of different z is the same as each other, the average delay of forwarding a VLI header by the switch can be expressed by:

$$\bar{d}_x = \frac{1}{256} \sum_{z=1}^{256} d_x, 8 \leq x \leq 224 \tag{2}$$

and the extension-header-size optimization problem can be formulated as follows:

$$\min \quad (\frac{\bar{d}_x}{\bar{d}_{\max}} + \bar{r}) \tag{3}$$

Pseudocode 1: Solving the Integer Programming Problem

Initialization: $x = 0$, $y = 0$, $z = 0$, $(\frac{\bar{d}_x}{d_{\max}} + \bar{r})[M] = 0$
1. **Procedure** Find_Optimal_Extension_Header_Size()
2. **for** $x = 0 : 8 : 224$ **do**
3. **for** $z = 1 : 1 : 256$ **do**
4. **Let** $y : xy \leq z < x(y+1)$
5. $d[x][z]$ <= switch.getDelay(x,z), $r[z] = (xy - z)$
6. $\bar{d}[x] = AVERAGE(d[x][z])$, $\bar{r} = AVERAGE(r[z])$
7. $\bar{d}_{max} = MAX(\bar{d}[x])$, $(\frac{\bar{d}_x}{d_{\max}} + \bar{r})[x] = (\frac{\bar{d}[x]}{d_{\max}} + \bar{r})$
8. **return** optimal_exheader_size = $MIN((\frac{\bar{d}_x}{d_{\max}} + \bar{r})[x])$
9. **end Procedure**

subject to:

$$\begin{cases} \bar{r} = \sum_{z=1}^{256}(xy - z) \\ xy \leq z < x(y+1) \\ 8 \leq x \leq 224 \\ x \ mod \ 8 = 0 \\ x = 0, \ y = 0, \ if \ z \leq 32 \\ x, \ y, z \in N \end{cases}$$

whereas $\bar{r}$ denotes the average redundant bits of the extension header, $\bar{d}_{\max}$ denotes the maximum average delay using different extension-header-size. Owing to the requirements of the BMv2 switch, each header size must be integer multiple of eight. Besides, there does not need the extension headers when the network scale is smaller than 2^{32} because the VLI basic header has a 32-bit identifier field. The aforementioned problem is an integer programming problem with limited range, therefore, the optimal solution can be found easily by conducting the Pseudocode 1.

C. Splicing Extension Headers

As an example, we assume a 56-bit identifier field in the VLI extension header to implement any length of identifier. When the Next-Header field is 0x80, a VLI extension header named Source Splicing Extension Header (SSEH) is followed with the basic header. The SSEH includes an 8-bit Next-Header field and a 56-bit Source-Identifier field. When the Next-Header field is 0x90, the basic header follows with a Destination Splicing Extension Header (DSEH). The DSEH includes an 8-bit Next-Header field and a 56-bit Destination-Identifier field. Using such two VLI extension headers, any length of an identifier can be implemented. For example, if a 64-bit source/destination identifier is required for a network, a VLI basic header and an SSEH/DSEH are enough for the requirement. If the basic header has a 32-bit identifier, other 32 (64 minus 32) bits of the SSEH/DSEH identifier field are available. One extension header is required. Moreover, if a 140-bit source identifier is required for a network, a VLI basic header and two SSEHs/DSEHs are enough for the requirement.

III. Performance Evaluation

Using the popular P4 language, we implement the VLI mechanism and evaluate the optimal extension-header-size by series of experiments.[2] Besides, the performance analysis are executed and compared in terms of processing delay when using different length of identifiers.

[2]Codes for VLI are available: https://github.com/KB00100100/VLI.

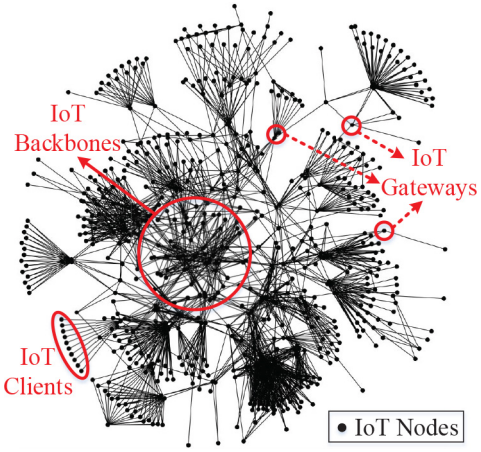


Fig. 4. The experimental topology.

TABLE I
Link Parameters in AT&T Topology

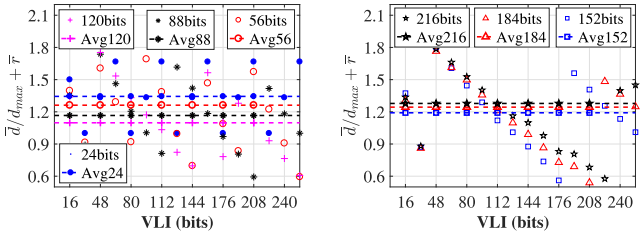
Parameters		Value
Backbone-	Delay	5ms∼10ms
Backbone	Bandwidth	40Mbps∼100Mbps
Gateway-	Delay	5ms∼10ms
Backbone	Bandwidth	10Mbps∼20Mbps
Gateway-	Delay	5ms∼10ms
Gateway	Bandwidth	10Mbps∼20Mbps
Client-	Delay	10ms∼70ms
Gateway	Bandwidth	1Mbps∼3Mbps

A. Experimental Setup

As shown in Fig. 4, the experiment is carried out in a realistic large-scale topology (namely AT&T). Particularly, the AT&T topology has 625 IoT nodes which can be classified into three categories (IoT clients, IoT gateways and IoT backbones) based on different degrees. There are 296 IoT clients in the topology. Each of them has a degree less than four and at least one link connected to the IoT gateway. Those nodes directly connected to IoT clients are classified as IoT gateway (221 such nodes in total). Each of them has at least one link connected to the IoT client and one link connected to other IoT gateways or the IoT backbones. The remaining 108 nodes are called IoT backbone nodes, which have different number of links connected to various IoT gateways. Besides, the links between IoT clients, gateways and backbones have different parameters. The link parameter details are listed in Table I.

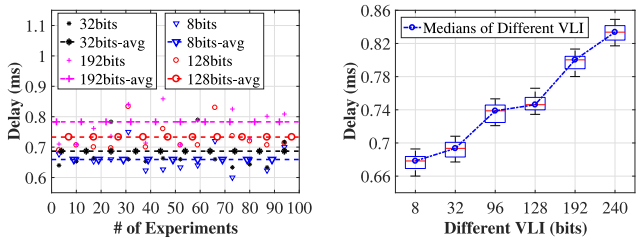
B. Optimal Extension-Header-Size

We conduct a test on the optimal extension-header-size and compare the results with the theoretical analysis introduced in Section II. Besides, there are seven different sizes of VLI extension header, from 16-bit to 256-bit in our experiments. The experimental results are shown in Fig. 5. Particularly, the dots marked with different shapes are the raw data of $(\bar{d}_x/\bar{d}_{\max} + \bar{r})$. It is difficult to find the relationship between those dots. Therefore, we give the average values of different VLI extension-header-size and plot them using various dash lines. The results are shown in dash lines and marked



(a) 24bits∼120bits (b) 152bits∼216bits

Fig. 5. The optimal VLI extension-header-size.



(a) A IoT backbone node (b) All IoT backbone nodes

Fig. 6. The processing delay with different VLIs.

with different types of forms such as circles, asterisks, and squares. Particularly, as Fig. 5(a) shows, the dash lines in 24-bit, 56-bit and 88-bit are all over the dash line marked with '+'. That means the average delay and redundant bits of the 24-bit, 56-bit and 88-bit are larger than the 120-bit ones. In other words, when the VLI extension-header-size is 120 bits, it has a lower average delay and less redundant bits than the VLI extension header with size of 24-bit, 56-bit and 88-bit. Besides, as Fig. 5(b) shows, the ordinate values of dash lines in 152-bit, 184-bit and 216-bit are all over 1.2, which are larger than 120-bit ones (approximatively 1.1 in Fig. 5(a)). That is, the 120-bit VLI extension header has the lowest average delay and redundant bits compared with other sizes of VLI extension header. This experiment shows a method to select the length of extension header.

C. Processing Delay With Different VLIs

To verify the processing delay for different VLIs, a comparison experiment between various length of identifiers is implemented in the AT&T topology. Particularly, the IoT backbone network has 108 nodes, therefore, an 8-bit VLI can satisfy the IoT backbone network requirement ($2^8 = 256 > 108$). Such an 8-bit identifier can be provided by the VLI basic header without using any extension header. We use the ingress and egress timestamps provided by the BMv2 switch to measure the processing delay for 8-bit, 32-bit, 128-bit and 192-bit identifiers. Fig. 6(a) shows the results from a random IoT backbone node. The 8-bit VLI, represented in the blue line, has the lowest processing delay compared with 32-bit, 128-bit and 192-bit VLI. In other words, the larger size is, the longer processing delay has.

Besides, we collect the processing delay data of all IoT backbone nodes and display them using the boxplot. Fig. 6(b) depicts the distribution of processing delay for different length of identifiers. Regarding the medians of processing delay, the values for 8-bit, 32-bit, 96-bit, 128-bit, 192-bit and 240-bit are around 0.67ms, 0.69ms, 0.73ms, 0.74ms, 0.80ms and 0.83ms, respectively. The blue dash line shows the trend of processing delay, which indicates a longer VLI brings a longer processing delay. Compared with IPv6 address with a size of 128-bit, the 8-bit VLI identifier can decrease the processing delay by 10.4%. Above all, it can improve the processing efficiency by using VLIs for different scales of networks.

IV. Conclusion

This letter has designed a novel VLI header to support a flexible field of identifier. Based on this, an extension-header-size optimization problem has been solved to minimize the packet processing delay. Besides, this letter has implemented the VLI header using the popular P4 language, also evaluated its performance in terms of processing delay. In the future work, we will integrate a VLI mapping mechanism to realize the communication between heterogeneous networks with different length of identifiers. Besides, more applications for assigning various length of identifiers will be implemented.

References

[1] R. Herrero, "Dynamic CoAP mode control in real time wireless IoT networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 801–807, Feb. 2019.

[2] P. Thubert and R. Cragie, "IPv6 over low-power wireless personal area network (6LoWPAN) paging dispatch," IETF, Fremont, CA, USA, RFC 8025, 2016.

[3] T. de Almeida Oliveira and E. P. Godoy, "Zigbee wireless dynamic sensor networks: Feasibility analysis and implementation guide," *IEEE Sensors J.*, vol. 16, no. 11, pp. 4614–4621, Jun. 2016.

[4] H. Li, K. Ota, and M. Dong, "LS-SDV: Virtual network management in large-scale software-defined IoT," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1783–1793, Aug. 2019.

[5] *The RIPE NCC Has Run Out of IPv4 Addresses*, RIPE, Amsterdam, The Netherlands, Nov. 2019. [Online]. Available: https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/the-ripe-ncc-has-run-out-of-ipv4-addresses

[6] C. Zhang, X. Sun, J. Zhang, X. Wang, S. Jin, and H. Zhu, "Throughput optimization with delay guarantee for massive random access of M2M communications in industrial IoT," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10077–10092, Dec. 2019.

[7] W. Quan *et al.*, "Adaptive transmission control for software defined vehicular networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 653–656, Jun. 2019.

[8] G. Liu, W. Quan, N. Cheng, H. Zhang, and S. Yu, "Efficient DDoS attacks mitigation for stateful forwarding in Internet of Things," *J. Netw. Comput. Appl.*, vol. 130, pp. 1–13, Mar. 2019.

[9] J.-W. Xu, K. Ota, M.-X. Dong, A.-F. Liu, and Q. Li, "SIoTFog: Byzantine-resilient IoT fog networking," *Front. Inf. Technol. Electron. Eng.*, vol. 19, no. 12, pp. 1546–1557, 2018.

[10] F. Wang, X. Shao, L. Gao, H. Harai, and K. Fujikawa, "Towards variable length addressing for scalable Internet routing," in *Proc. IEEE 35th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2016, pp. 1–9.

[11] S. Bradner and A. Mankin, *The Recommendation for the IP Next Generation Protocol* IETF, Fremont, CA, USA, RFC 1752, 1995.

[12] H. Zhang, W. Quan, H. Chao, and C. Qiao, "Smart identifier network: A collaborative architecture for the future Internet," *IEEE Netw.*, vol. 30, no. 3, pp. 46–51, May 2016.

[13] S. Ren *et al.*, "Routing and addressing with length variable IP address," in *Proc. ACM SIGCOMM Workshop NEAT*, 2019, pp. 43–48.

[14] Y. Meng, W. Zhang, H. Zhu, and X. S. Shen, "Securing consumer IoT in the smart home: Architecture, challenges, and countermeasures," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 53–59, Dec. 2018.