Adaptive Addresses for Next Generation IP Protocol in Hierarchical Networks

Haoyu Song, Zhaobo Zhang, Yingzhen Qu, James Guichard
Futurewei Technologies, USA

***Abstract*—We propose the adaptive addresses under a hierarchical network structure, which can be realized in a newer generation of IP protocol (*i.e.*, IPvn). It minimizes the communication overhead, enables arbitrary address space extension, simplifies both the network data-plane and control-plane, and supports better network security. More importantly, it supports incremental deployment from the network edge and gradual growth towards the core. A clear boundary between IPvn domain and the existing IPv4/IPv6 networks enables transparent cross-domain communication. The clear evolution path makes pre-standard deployment possible. We design both control plane and data plane, prototype the routers within and on the edge of an IPvn domain, and evaluate the performance. We open source the project to encourage further investigation and development.**

I. Introduction

Internet of Things (IoT) and 5G introduce to the Internet a huge number of addressable entities (*e.g.*, sensors, machines, vehicles, and robots). The connected IoT devices are projected to reach 75.44 billion worldwide by 2025 [1]. As of November 2019, the unallocated IPv4 address blocks are depleted [2]. Apart from the interim remedies of Network Address Translation (NAT) and address recycling, an accelerated transition to IPv6 is expected. While the 128-bit address of IPv6 was considered large enough and future-proof, the current IPv6 practice has several issues. First, the long IP addresses inflate the packet header size. 80% of a basic IPv6 header is consumed by addresses. Second, IPv6 addresses are allocated with large blocks (*e.g.*, a perfix length of /48 or /56 is recommended [3]). The extravagance may raise another address exhaustion concern. Third, there is a trend toward extending the meaning of the IPv6 address beyond connectivity (*e.g.*, SRv6 network programming [4]) leading to further address usage.

Address Overhead: In IoT networks, thing-to-thing communication through wireless connections is dominant, which presents several distinct characteristics. (1) The communication pattern is mainly frequent short-message exchanges (*e.g.*, industry robots and networked vehicles). (2) The communication is often energy sensitive (*e.g.*, battery-powered sensors). (3) The communication often requires low latency (*e.g.*, industry control). (4) The precious wireless channels demand high bandwidth utilization (*e.g.*, ZigBee, Bluetooth, Wi-Fi, and 5G). These characteristics render a large header overhead unfavorable and even prohibitive.

The address overhead also takes its toll on Data Center Networks (DCN), especially when large scale containers are deployed and their prevailing communications are comprised of short messages (*e.g.*, key-value pairs) and conducted through virtual switches. The performance hit is hard when packets tunnel through host machines [5].

On the other hand, in IoT and DCN, most communications happen between adjacent and related entities. It is a good practice to locally confine communication, computing, and storage due to performance, efficiency, and security considerations, as advocated by Edge Computing [6]. This pattern provides an opportunity to overcome the overhead problem.

Address Space Extension: A fixed length address scheme only defines one monolithic address space. Each entity (*i.e.*, any Internet addressable device) is assigned a flat address as its global identifier, which is used for communication with other entities. Because of this structure, the only way to combat future address exhaustion is to migrate to a larger address space and reassign longer addresses to all entities. This disruptive process requires an overhaul on applications, protocol stack, and global networks.

A better approach is to allow any entity to keep its base address unchanged even when the address space is expanded. As far as an entity is concerned, nothing needs to be updated and it is business as usual. Address space expansion is transparent to all entities and even to most parts of the networks.

Adaptive Address in Hierarchical Networks: In the era of Internet of everything, it is preferred to contain and organize the directly related entities in isolated and hierarchical networks. Doing so not only ensures a securer network environment but also allows more efficient communications. Although it is necessary for an entity to be globally identifiable, the entity itself does not need to own a full address. A locally unique address (*i.e.*, the global address suffix) is enough for it to communicate with others as long as a strict hierarchical network structure is maintained.

The hierarchical network and address assignment are not unfamiliar (*e.g.*, ATM PNNI [7]). For the Internet. the original IPv4 address is designed to be classful [8]. However, due to the inefficiency of the coarse class granularity, the Internet resorted to Classless Inter-Domain Routing (CIDR)—the design philosophy is also inherited by IPv6—to slow down the address exhaustion [9]. It is no longer possible to maintain a cleanly organized network hierarchy.

However, if we group entities into hierarchical networks based on location, ownership, or logical relationship, we will end up with a clean network architecture and can use addresses as short as possible for communication. All the

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

entities attached to the same network share a same super-net prefix, which can be prepended before their local addresses to make them uniquely addressable in the network one level higher. This recursive process can be repeated until the top-level network. A full and unique global address for each entity only reveals at the top level of the hierarchy.

Such an architecture makes it easy to expand the address space by adding a new level of super-net on top of the original top-level network. Now the original top-level network is demoted to a second level network, and other new networks with independent address space can be created alongside. Meanwhile, the existing network hierarchy remains intact and the existing entities are oblivious to the change, except that the global addresses are effectively extended. Moreover, any entity's local address update is confined in the entity's immediate network and any prefix update is confined in the immediate super-net. Such features are ideal for maintaining a stable and evolvable network infrastructure.

To make such an idea work, however, we need to redesign the Internet protocol header (at least for the address part) and the way routers process and forward packets. We also need to consider the compatibility with IPv4 and IPv6, conceive a feasible deployment strategy, and present provable benefits.In this paper, we provide the system design and P4-based data-plane implementation and evaluation. We demonstrate a clear evolution path for pre-standard incremental deployment, so IoT and DCN can enjoy its benefits instantly.

Related Work: Although ATM PNNI [7] adopts hierarchical network and addressing but full addresses are always used. The link-based IP header compression techniques and protocols [10], [11] are not designed for network-wide application. Cross-layer optimizations such as 6LoWPAN [12] is not general enough for other network environments. Different variable-length IP address schemes was proposed in [13], [14] with the similar motivations, but these works did not describe the network data-plane and control-plane functions, and deployment strategy as in this paper. NAT, L3 tunnelling, SRv6 [15], and RSIP [16] all modify the L3 header or addresses in routers during the packet forwarding, but none of these modifies the address length.

II. Adaptive Address Scheme

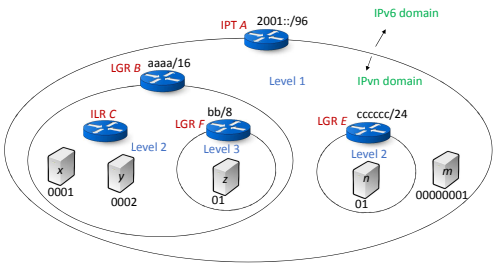

Fig. 1. Hierarchical network and address example.

To begin with, we eliminate the maximum length restriction of a complete address. One may argue that 128-bit address as in IPv6 is more than enough if used with prudence. However, it is better to be flexible in address space size and be prepared for the extensibility in advance when designing new addressing schemes. Consider a scenario that the 128-bit address is used exclusively for entities on earth, and we allocate addresses for extraterrestrial entities in different address spaces. We can consider all the terrestrial entities to be in a single 128-bit network. Another unique super-net prefix needs to be appended to this local 128-bit address before a terrestrial entity can communicate with an extraterrestrial entity.

In an expandable hierarchical network, the "local" address of an entity is the shortest possible address assigned to it which allows the entity to be uniquely identifiable in its immediate network. The entity does not know the number of network levels above it as well as the super-net prefixes.

TABLE I
Example Network Configurations

Host	Address	Length	Level	Gateway	Super-net Prefix
x	0x0001	2B	2	B	0xaaaa/16
y	0x0002	2B	2	B	0xaaaa/16
z	0x01	1B	3	F	0xbb/08
m	0x00000001	4B	1	A	0x2001::/96
n	0x01	1B	2	E	0xcccccc/24

Figure 1 shows a hierarchical network example and Table I summarizes the corresponding configurations. The example contains 5 hosts and 4 networks in 3 levels. The Level 1 network has a 32-bit address space and interfaces with an IPv6 network. The two Level 2 networks below it have a 32-bit and 16-bit address space, respectively. The Level 3 network below the left Level 2 network has an 8-bit address space.

The example shows the variable-length addresses are used. Current IP protocols only use fixed-length addresses. The header of the next generation IP protocol (*e.g.*, IPvn) therefore needs to be amended to support variable-length addresses. In addition to the Source Address (SA) and the Destination Address (DA), it also needs the corresponding Source Address Length (SAL) and the Destination Address Length (DAL). One possible field layout is shown in Figure 2.

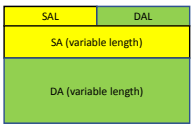

Fig. 2. A possible layout of address related fields in IPvn header.

SAL and DAL have fixed length. To simplify the implementation, SA and DA are preferred to be byte-aligned. It is possible to define the length of address in the unit of byte, nibble, or bit. Each has its own pros and cons. The unit of byte can help reduce the size of the SAL/DAL but results in coarse network granularity which might be inefficient in address allocation. For example, a 4-bit SAL/DAL is enough to encode 16 possible address lengths (one to 16 bytes) for networks. In this design, each super-net is at least 256 times greater than the networks below it. On the other extreme, the

unit of bit allows fine network granularity but requires more space for SAL/DAL. For example, 8-bit SAL and DAL can support an address length up to 256 bits and a super-net is only twice larger than a lower level network below it. With a few bits, it is also possible to design a more sophisticated encoding scheme that supports variable address length steps and adapts to the ideal network sizes at different levels.

III. Router Role and Function

In the hierarchy, each network has one or more Level Gateway Routers (LGR) which are responsible for forwarding packets in or out of this network. The LGRs are the only interface between a network and its super-net.

A network can be in a single L2 domain, which means all the entities in this network (excluding those in lower level networks) and all the network devices (including the LGRs to the super-net and the lower level networks) are L2 reachable.

A network can also be a pure L3 network in which no L2 device is allowed. Each entity in a network is directly connected to either an LGR or some Intra-Level Router (ILR) which is solely responsible for packet forwarding within the network. In this case, the entities need to partially participate in the routing process (*e.g.*, advertising its address).

The scale of an intra-level network can be used to guide the L2/L3 selection. Small networks prefer the L2-based solution and large networks prefer the L3-based solution. In the higher level networks, since the number of entities is usually small, it is free to choose between L2 or L3-based solution. The lowest level network is usually L2-based.

Unlike IPv4 and IPv6, the address related fields in IPvn header can be modified in network by LGRs. An LGR of a network keeps a prefix that can augment the SAs from this network to an address in the super-net. If an LGR needs to forward an internal packet outside (*i.e.*, DAL>SAL), it augments the packet's SA and updates its SAL accordingly. Reversely, if an LGR receives a packet destined for the lower level network it serves from the super-net (*i.e.*, the super-net prefix matches the DA's prefix), it strips off the super-net prefix from the packet's DA and updates its DAL accordingly.

In contrast, within an L3-based level network, ILRs do not modify the address fields. An ILR can decide the packet forwarding direction by examining the DAL. If DAL>SAL, the packet needs to be forwarded to an LGR of this network; otherwise, the packet needs to be forwarded within the current network, and possibly into a lower level network.

We use Figure 1 to illustrate some packet forwarding examples. We allocate 8 bits to encode the address length in bytes. Figure 3 shows that entity x sends a packet to y. This type of communication may account for most of the traffic for IoT or DCN. The size of the address-related fields is kept to the minimum and there is no address transformation involved.

Figure 4 shows a more complicated and interesting example for which x sends a packet to n. Since DAL>SAL, the packet is forwarded to LGR B, at which SA of the packet is augmented and the packet enters the Level 1 network. Now since SAL=DAL, the packet is forwarded in the current

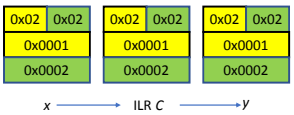

Fig. 3. x sends a packet to y, both in the same L3-based network.

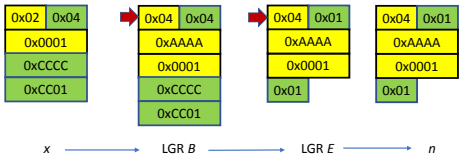

Fig. 4. x sends a packet to n in a different network.

network and eventually reaches LGR E. LGR E prunes DA of the packet and forwards the packet into the Level 2 network and eventually to n. n can directly talk back to x because from the packet, n has acquired x's "global" address.

Interfacing with IPv4/IPv6 Networks: The difficulty of introducing new architectures and protocols into Internet is partially due to the lack of a backwards-compatible incremental deployment strategy [17]. An end-to-end overhaul is economically unjustifiable. Instead, we need to incrementally bring it to the Internet and make it seamlessly work with existing IPv4 or IPv6 networks. Fortunately, this is straightforward.

We discuss two scenarios. First, the whole IPv4 (IPv6) network space is considered a 32-bit (128-bit) lowest level network in IPvn (*i.e.*, IPv4/IPv6 in IPvn). Within the IPv4 (IPv6) network, the communications continue to use the IPv4 (IPv6) protocol. However, if an entity in the IPv4 (IPv6) network needs to communicate with outside entities, they need to switch to use the IPvn protocol at the gateway between the two types of networks. This approach keeps the current IPv4 and IPv6-based networks intact and grows the IPvn-based networks alongside. The second scenario (*i.e.*, IPvn in IPv4/IPv6) allows the support of IPvn within the IPv4 (IPv6) address space. In this scenario, the IPv4 (IPv6) network is considered the top level super-net and the 32-bit (128-bit) addresses cover the entire address space, in which we can construct hierarchical lower level networks to support IPvn. This scenario contains two sub cases.

Case 1: Each top-level IPvn network is assigned a unique prefix to extend the local addresses of the entities in it to full IPv4/IPv6 addresses. The LGR of the top-level IPvn network (named IP Translator or IPT) is responsible for protocol translating between IPvn and IPv4/IPv6. IPT A in Figure 1 illustrates such an example.

Case 2: Each top level IPvn network is assigned one or more public IPv4/IPv6 addresses. The entities in IPvn domain use private addresses. In this case, LGR of the top-level IPvn network conducts both protocol translation and NAT.

The Case 1 is especially attractive because it supports a simple, straightforward, and incremental deployment strategy, that helps to introduce IPvn into the current Internet with the least resistance and the most benefits. For successful protocol

translation, the IPvn header needs to maintain other information than the address-related fields to match the IPv4/IPv6 header. The specification of the IPvn header is beyond the scope of this paper.

IV. Control Plane Design

It is conceivable that all the control plane functions and protocols need to be modified or redesigned due to the hierarchical network architecture of IPvn. Fortunately, the updates are often incremental and the results are usually simpler than their counterparts in IPv4 and IPv6. We discuss a few essential protocols that are sufficient to implement a system prototype.

A. Address Configuration and Resolution

DHCP: An entity can be manually configured or dynamically acquire its address when booting up. Each network may contain a Dynamic Host Configuration Protocol (DHCP) server responsible for assigning addresses to the entities in the same network. The protocol is almost identical to that for IPv4 and IPv6, except that the assigned address length is adaptive to the network size.

DNS: For an entity to acquire the address of a peer entity in order to initiate a communication, Domain Name System (DNS) is still the prominent approach but with a new service model. Any network can provide name service. Each entity is configured with the address of the closest DNS server. The hierarchical network architecture allows a scoped domain name service. That is, a name registered in a network is only valid in this network and the lower level networks covered by it. It is possible that a same name is registered in two networks and one network is the other's super-net. Such name conflict is not a bug but a feature for name reuse, which is transparent to the name query process.

Each network may contain a DNS server (the notation is only logical. The actual implementation may follow the same hierarchical and distributed architecture of today's DNS). Each DNS server knows the nearest DNS server in a higher level network and the nearest DNS servers in lower level networks. This essentially organizes the DNS servers in the network hierarchy into a tree structure rooted at the top-level network. Each named entity in a network is registered with the DNS server that covers its scope, which is basically a subtree.

We have several methods to populate the name to support the scoped name queries, each with different storage and performance trade-off: 1) register the name in all the DNS servers in its scope (*i.e.*, all the subtree nodes); 2) recursively register the name in every parent DNS server until the scope root; and 3) register the name only in the DNS server in its scope root. The address for a name returned by a DNS server is on a "need-to-know" basis. In a network, if the address's prefix matches it super-net prefix, the prefix is removed. This can be easily done by the original or the relay DNS servers. If a query crosses the IPvn domain and enters into the IPv4/IPv6 domain, the protocol translation is also needed.

ARP: In a L2-based network, the operation of Address Resolution Protocol (ARP) or Neighbor Discovery Protocol (NDP) is almost identical to that for IPv4 and IPv6. In an L2-based network, each immediate entity should be configured with a default gateway address to its super-net. If no default gateway is configured, a network LGR should be configured as an ARP proxy to respond to all internal ARP requests for addresses out of the network. Similarly, the LGRs of any lower level network in this network are also needed to be configured as ARP proxy to response all ARP requests for addresses in the lower level network. Due to the multi-homing gateway routers, an ARP request may receive multiple responses. It is up to the requester to determine which one to cache.

Routing Protocol: The address aggregation due to the hierarchical network architecture also benefits the routing protocols. A lower level L3-based network may belong to a single organization or Autonomous System (AS), so the interior gateway routing protocols (IGP) such as OSPF and IS-IS can be used. Other lower level networks in this network can be considered OSPF stub areas or IS-IS levels. A simpler way is that each network run an independent instance of OSPF or IS-IS. Specially, an LGR runs two OSPF/IS-IS instances: one for the super-net and the other for the lower level network.

On the other hand, a higher level L3-based network may contain multiple ASes and forms the backbone of the lower level networks. Like today's Internet, the ASes could come from multiple Internet Service Providers (ISP) with peering relationship. Each lower level network becomes a stub AS. In this case, the exterior gateway routing protocol (EGP) such as BGP can be used. The hierarchical architecture solves the routing protocol scalability issue, and simplifies the protocol implementation by removing unnecessary features. The clean routing scope helps to secure the infrastructure and troubleshoot the networks.

V. Data Plane Design

IPvn Socket for End Entities: To enable IPvn as a new network layer protocol in end entities, we could add the protocol implementation in the OS Kernel and allow applications to invoke the socket API using the address family parameter `AF_INETN`. The L4-L7 protocol stack and the application logic remains the same, allowing direct communication between entities in IPvn domain and in IPv4/IPv6 domain.

Forwarding Table Lookups in Networks: The adaptive address simplifies the router forwarding table structure in L3-based networks. A forwarding table only contains the addresses to local entities and the prefixes to the lower level networks. Since there is no nested prefixes, the Longest Prefix Matching (LPM) is not necessary. The small number of unique prefix lengths allows the prefixes to be grouped on lengths and each group to be implemented as a hash table. A lookup can search all the hash tables in parallel, and at most one table can result a positive match. This design avoids the use of expensive TCAM or other complex trie-based algorithms.

An LGR has two types of interfaces: one faces a lower level network and the other faces the super-net. One LGR may serve more than one lower level network. Hence, an LGR may contain multiple logical forwarding tables, with each for

a network. For a packet in LGR, once its target network is determined and the address related fields are processed, the proper forwarding table can be searched.

VI. Implementation & Evaluation

Prototype: We draft an IPvn header format as shown in Figure 5. The header contains enough information to allow the IP header translation to and from IPv4 and IPv6.

Fig. 5. A proposed header format of IPvn prototype.

We use P4 [18] for prototyping. Compared to IPv4 and IPv6, the variable-length addresses in IPvn pose the main challenge in the implementation due to the limited support of variable-length header in P4 language. One dynamic-size data type offered by P4 is "varbit", but most operations like addition, subtraction, arithmetic shift are not applicable to it, which limits its usability. Alternatively, we use the header stack, an array of fixed-length headers [19]. The dynamic address and padding are split into multiple 8-bit elements, which can be pushed and popped in a header stack, and the complete address can be recovered by concatenating those multiple 8-bit elements. Different lengths supported in the router are all declared in the header (*e.g.*, one-/two-/four-byte addresses), so an address extraction takes just one state transition.

The implementation framework for our P4-based prototype is shown in Figure 6. The routers, including ILR, LGR, and IPT, are written in P4-14 language [18]. Through the P4 compiler, *p4c*, the programs are compiled and installed to the BMv2 software switch. Forwarding tables and switch parameters can be pre-loaded or dynamically changed using *Simple_Switch_CLI* during runtime. The customized switches run on a configured network topology in the Mininet emulation environment [20]. The whole environment runs as a docker container from the open-source tool p4app [21].

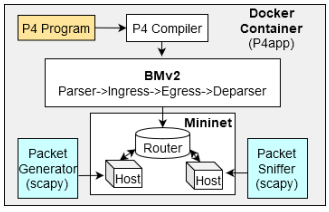

Fig. 6. Implementation and simulation environment.

We open source the entire project [22]. Currently, it contains the network data-plane prototype and simulation environment. New work on the new L3 support in end entities, hardware-based prototypes, and control-plane implementations, will be included in the project.

Environment: The evaluations run on an Ubuntu 18.04 server with 10-core CPU at 2.2GHZ and 64GB RAM. Since BMv2 is not meant to be a production-grade software switch and the whole network simulation runs as a docker process, we focus on the relative performance under different implementations rather than the the absolute performance. Scapy [23] is used to generate and sniff packets. New protocols can be easily defined using the *Packet* class provided by Scapy, and the function *sendp* is used to send layer 2 packets.

Software Switch Performance: We use the timestamps on interfaces from the Pcap Tool to calculate the packet processing time. The forwarding performance of ILR, LGR, and IPT is shown in Fig. 7. The first column is for ILR forwarding within a network; the second one is for ILR forwarding toward super-net (it is faster because no table lookup is needed); the third one is for LGR forwarding to a lower level network; the fourth one is for LGR forwarding to a super-net; the last two are for IPT conversions between IPv6 and IPvn.

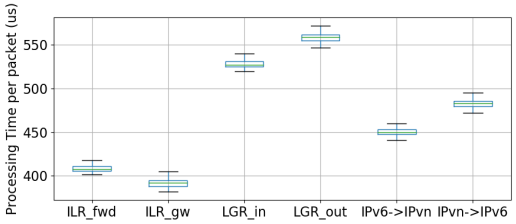

Fig. 7. Processing time per packet of ILR, LGR and IPT

Fig. 8 shows the processing time for one hundred IPv4, IPv6, and IPvn packets sent in sequence. The longer processing time of IPvn is due to the lack of efficient support on variable-length header fields in P4. Our experiments use a small forwarding table with a few entries, so the table lookup cost is negligible. In reality, forwarding table lookup is the most time-consuming for packet processing where IPvn's unique address structure can help achieve a much higher overall forwarding performance.

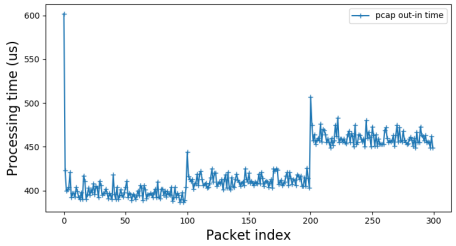

Fig. 8. Packet processing time for IPv4, IPv6, and IPvn.

The above evaluation only applies to the software switch and router. In the more prevailing hardware, the extra header processing in ILR and LGR only needs a few extra pipeline stages which slightly increase the forwarding latency but have no influence on throughput. The chip die size saved due to the compact forwarding table size and efficient forwarding table implementation for IPvn can directly translate into higher I/O bandwidth and/or deeper buffer.

Overhead: Compared with IPv6, the address-related overhead saving by using the adaptive address is from 87.5% to 68.8% when the network size is from 1 Byte (*i.e.*, up to 256 entities) to 4 Bytes (*i.e.*, up to 4 billion entities). If the entire IP header is considered, the overhead saving is between 70% and 60%, as shown in Figure 9.

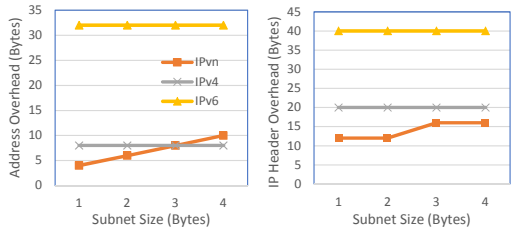

Fig. 9. Overhead comparison: address fields only and IP header.

Power: Networking is responsible for more than 80% of the total power consumption for wireless IoT devices due to the radio and processing [24]. It is difficult to measure the actual networking power consumption without a production environment. However, it is well established that, at a bandwidth C, the networking power consumption is, $P(C) = P_i + E_b C$, in which P_i is the idle power and E_b is the energy for per-bit transmission [25]. P_i usually consumes less than 10% of the power [24] and for a highly shared IoT networking device, P_i is negligible [25]. As shown in Figure 10, given the majority packet payload size ranges from a few bytes to a few tens of bytes, the power saving is between 20% to 50%.

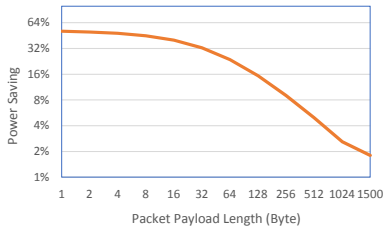

Fig. 10. Power saving over IPv6-based IoT, with the assumption of a 14-byte MAC header and up to 64K IoT entities in a network.

VII. Conclusion

To combat the Internet ossification, a new layer 3.5 was proposed to provide an evolution path by allowing new protocols at AS edges [17]. Our principle differs in that we start the evolution from the network edge, because each edge network is by nature a single management domain and has a clean interface with external networks. Our solution does not stop at the edge. We provide a clear evolving path to expand the scope of the new protocol towards the core and make the entire Internet extensible. Such an approach allows the pre-standard deployment at edge and the use of IPT gateways to interface with existing IPv4 or IPv6 networks in the core. The places where the new address scheme is mostly appreciated can enjoy the benefits immediately.

The advantages of the new address scheme include, but not limited to, power saving, effective bandwidth improvement, simplified data plane, simplified control plane, and boundless address space extension. The benefits of a hierarchical network architecture, because of the adaptive IP address, is more profound. It introduces scope to the domain name system (DNS) and hierarchy to the autonomous system (AS), which provide better system scalability, security isolation, policy management, and network robustness. Due to space limitation, more details can be found in [22]

The change of Internet architecture, even just partially and incrementally, is a daunting task. This paper just scratches the surface. It leaves many issues untouched (*e.g.*, IP mobility and multi/broad/anycast), and many details yet to be explored. By releasing the open source project and sharing our preliminary results, we hope to trigger more debates as well as quality research and development from academia and industry.

References

[1] Statista Research Department, "Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025." https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, 2019.

[2] RIPE NCC, "The RIPE NCC has run out of IPv4 Addresses." https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/the-ripe-ncc-has-run-out-of-ipv4-addresses, 2019.

[3] T. Narten, et al., "IPv6 Address Assignment to End Sites," RFC 6177, IETF, 2011.

[4] A. Mayer, et al., "The Network as a Computer with IPv6 Segment Routing: a Novel Distributed Processing Model for the Internet of Things," in *NGOSCPS*, 2019.

[5] D. Zhuo, et al., "Slim: OS kernel support for a low-overhead container overlay network," in *NSDI*, 2019.

[6] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[7] ATM Forum, "Private NetworkNetwork Interface Specification, Version 1.0," *af-pnni-0055.00*, 1996.

[8] ISI, "Internet Protocol," RFC 791, IETF, 1981.

[9] Y. Rekhter, et al., "An Architecture for IP Address Allocation with CIDR," RFC 1518, IETF, 1993.

[10] V. Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links," RFC 1144, IETF, 1990.

[11] M. Degermark, et al., "IP Header Compression," RFC 2507, IETF, 1999.

[12] J. Hui, et al, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," RFC 6282, IETF, 2011.

[13] S. Ren, et al., "Routing and Addressing with Length Variable IP Address," in *NEAT*, ACM, 2019.

[14] J. Tang, et al., "A Flexible Hierarchical Network Architecture with Variable-Length IP Address," in *IEEE International Workshop on New IP: The Next Step*, 2020.

[15] C. Filsfils, et al., "IPv6 Segment Routing Header (SRH)," RFC 8754, IETF, 2020.

[16] M. Borella, et al., "Realm Specific IP: Framework," RFC 3102, IETF, 2001.

[17] J. McCauley, et al., "Enabling a Permanent Revolution in Internet Architecture," in *SIGCOMM*, ACM, 2019.

[18] P4 Language Consortium, "P4 Language and Related Specifications." https://p4.org/specs/.

[19] P4 Variable Length Header. https://github.com/jafingerhut/p4-guide/tree/master/variable-length-header.

[20] Mininet, "An Instant Virtual Network on your Laptop (or other PC)." http://mininet.org/.

[21] P4app Simulation Tool. https://github.com/p4lang/p4app.

[22] Futurewei, "Adaptive Address for Next Generation IP Protocol." https://github.com/Fizzbb/ResearchPaper/tree/master/Adaptive-Addresses-for-NG-IP, 2020.

[23] Python Scapy. https://scapy.readthedocs.io/en/latest/build_dissect.html.

[24] S. Zhao, et al., "Understanding Energy Efficiency in IoT App Executions," in *IEEE ICDCS*, 2019.

[25] C. Gray, et al., "Power consumption of IoT access network technologies," in *Workshop on Next Generation Green ICT, IEEE ICC*, 2015.