Adaptive Address for Next Generation IP Protocol in Hierarchical Networks

Haoyu Song, Zhaobo Zhang, Yingzhen Qu, James Guichard

Futurewei Technologies, USA
Motivation

- Short messages
- Energy sensitive
- Delay sensitive
- Resource limit
- Independent address space
- Identity retention
- Nondisruptive upgrade

Header Overhead

Address Extensibility
Observation

- Hierarchical network architecture
- Most communication happens between adjacent entities
- Communication possible if entities can be uniquely identified *mutually*
Adaptive Address in Hierarchical Networks

**Entity Address** = *Network ID* + *Entity ID*

- **Delegate** Network ID maintenance and operation to networks
- Entity only knows and uses its own Entity ID
- For communication, entity only get the peer’s partial address up to the first network level they share
  - E.g., in the lowest level network, entity ID is enough
Address Fields in Header & Border Router’s Function

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Address (variable)</td>
<td>Destination Address (variable)</td>
</tr>
</tbody>
</table>

- LGR keeps a prefix (network ID) for the network below it
  - Egress packet: add the prefix to the source address
  - Ingress Packet: remove the prefix from the destination address
- ILR for intra-level traffic forwarding
Interface with IPv4/IPv6 Networks

IPv4/IPv6 Network as a lowest level network in IPvn

IPv4/IPv6 Network

Lower levels of IPvn Networks

IPvn networks within IPv4/IPv6 network

IPv4/v6 Network

IPvn Network Hierarchy

IPvn Network Hierarchy

- IPvn with private addresses (NAT)
- **IPvn with assigned public address block**
  - LGR → IPT (IP Protocol Translator)
Control Plane Design

• DHCP
  • Entity ID assignment

• DNS
  • Hierarchical tree architecture
  • Name scoping and name overriding

• ARP/NDP
  • LGR as ARP proxy for its prefix

• Routing Protocol
  • Low level network can be a single AS
  • High level network can contain multiple ASes
    • With lower level network as stub AS
Data Plane Design

• End Entity
  • New IPvn socket supporting IPvn address family
  • Header convertible to IPv4/IPv6
  • Same L2, and L4-L7 protocol stack

• Routers
  • Smaller and simplified forwarding table
    • No nested prefixes
    • Network level address aggregation
  • New functions in LGR
    • Source and destination address manipulation
  • New functions in IPT
    • IP protocol translation
### Implementation

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver(8)</td>
<td>8-bit version number</td>
</tr>
<tr>
<td>Header Length</td>
<td>Length of header in bytes</td>
</tr>
<tr>
<td>ToS/TC</td>
<td>Type of Service/Transport Control field</td>
</tr>
<tr>
<td>Next Header</td>
<td>Next header type</td>
</tr>
<tr>
<td>Hop Limit/TTL</td>
<td>Time to live/hop limit</td>
</tr>
<tr>
<td>Payload Length</td>
<td>Length of payload data</td>
</tr>
<tr>
<td>SAL</td>
<td>Source Address Length field</td>
</tr>
<tr>
<td>DAL</td>
<td>Destination Address Length field</td>
</tr>
<tr>
<td>SA</td>
<td>Source Address</td>
</tr>
<tr>
<td>DA</td>
<td>Destination Address</td>
</tr>
<tr>
<td>Padding</td>
<td>Padding bits</td>
</tr>
</tbody>
</table>

#### IPVn header format

#### IPVn header parse graph in P4

#### Implementation and Simulation Environment

**Docker Container (P4app)**
- P4 Program
  - P4 Compiler
  - BMv2
    - Parser
    - Ingress
    - Egress
    - Deparser

**Mininet**
- Packet Generator (scapy)
- Packet Sniffer (scapy)
- BMv2
- Host
- Router
Evaluation I

Per-Packet Processing Time in Different Types of Routers

- Reflect relative performance in software implementation
- Insufficient support of variable length header in P4
- Not consider the performance for address lookups
Evaluation II

Overhead Comparison

IoT Power Saving over IPv6
Conclusion

- Efficient addressing scheme for IoT and data center networks
- Future-proof extensible address space
- Incrementally deployable from the edge
- Fully interoperable with existing IPv4/IPv6 network and end entities
- Simplified network control/data plane protocol & implementation
- Open source
Thank You.