DESIGN, IMPLEMENTATION, AND OPERATION OF IPV6-ONLY IAAS SYSTEM WITH IPV4-IPV6 TRANSLATOR FOR TRANSITION TOWARD THE FUTURE INTERNET DATACENTER

Keiichi SHIMA¹ (Presenter)
Wataru ISHIDA²
Yuji SEKIYA²

¹IIJ Innovation Institute Inc.
²The University of Tokyo
BACKGROUND

• Increasing computing power and wide spreading virtualization technology

• Demand for more flexible configuration of network service backend system and cloud computing technology

• IPv4 address depletion, IPv6 deployment, and operation cost of infrastructure
CLOUD SERVICES FROM DIFFERENT VIEWPOINTS

- Users
  - Provide
  - Require
  - SaaS
  - Provide
  - Require
  - PaaS
  - Provide
  - Require
  - IaaS

- Service providers
  - Provide
  - Require

- Infrastructure providers
CLOUD SERVICES FROM DIFFERENT VIEWPOINTS

Users

Users / Service providers

Users / Service providers

Infrastructure providers

Our target

Provide

Require

SaaS

Provide

Require

PaaS

Provide

Require

IaaS
REQUIREMENTS FOR FUTURE INTERNET DATACENTERS

• Datacenter interconnection for scaling out infrastructure or service beyond geographical limitation

• Transparent resource availability over geographically distributed datacenters

• Migration to IPv6 with less operation cost without losing IPv4 client backend compatibility
REQUIREMENTS FOR FUTURE INTERNET DATACENTERS

• Datacenter interconnection for scaling out infrastructure or service beyond geographical limitation

• Transparent resource availability over geographically distributed datacenters

• Migration to IPv6 with less operation cost without losing IPv4 client backend compatibility

Our focus in this paper
BASIC CONCEPT

- Build infrastructure network only with IPv6
- IPv4 compatibility is necessary only for frontend nodes that interact with users
- IPv6-IPv4 translation is performed between frontend nodes and user nodes
The basic idea is same as DNS64/NAT64

The important design choice and limitation is that one IPv4 and one IPv6 addresses are mapped directly one by one

For redundancy and scaling out

Drawback: IPv4 address utilization (but remember that IPv4 addresses are assigned only to frontend nodes)
HOW TRANSLATOR WORKS

IPv4 Internet → Proposed IaaS system

IPv4 client → IPv4 packet → Translator

Src: Client$_{IPv4}$
Dst: Server$_{IPv4}$

IPv6 packet → IPv6 server

Src: Pseudo prefix::Client$_{IPv4}$
Dst: Server$_{IPv6}$

Server$_{IPv4}$ ↔ Server$_{IPv6}$

Mapping table
IMPLEMENTATION AND DEPLOYMENT

- Frontend nodes and backend network, built on top of the wide area L2 network
- Two exit points located at Tokyo and Osaka
SYSTEM EVALUATION

• Redundancy verification between Tokyo and Osaka

  1. Stop radvd on the translator node at Tokyo
  2. Stop translator function on the translator node at Tokyo
  3. Restart radvd, translator at Tokyo
system evaluation

Figure 1: The overview of the proposed IaaS system

In this proposed system, we assume an IPv4 address and an IPv6 address are mapped one to one. Because the proposed system discussed in this paper is for IaaS providers providing server resources to their users, who are PaaS providers, SaaS providers, and/or ASPs, we cannot design the system to share one IPv4 address by several IPv6 server nodes. We also don’t consider implementing this requirement using application level proxy solution, since service providers have different requirements for their services, we cannot limit the communication layer function to application layers. This may be a problem considering that the IPv4 address is a scarce resource. However, we need to use IPv4 address anyway if we want to support IPv4 clients. And also we need to map addresses one to one if we want to provide services without any modification on the client side. Our best effort to handle this issue is that we do not allocate IPv4 addresses to backend nodes to avoid non-necessary IPv4 use.

The version of an IP protocol used in the backend network can be either IPv4 or IPv6. In the implementation shown in this paper, we use IPv6 as an internal protocol for the backend network considering the trend of IPv6 transition of the whole Internet.

Figure 1 depicts the overview of the proposed system. We said that the frontend nodes provide dual-stack services to the client before, however precisely speaking, these frontend nodes do not have any IPv4 addresses. The mapping information between IPv4 and IPv6 addresses are registered to each IPv4-IPv6 translator node and shared by all the translator nodes. Since the mapping is done in the one to one manner, no translator nodes need to keep session information of ongoing communication. They can just translate IP packets one by one. This makes it possible to place multiple translator nodes easily to scale out the translation service when the amount of the traffic grows.

This feature also contributes robustness of the translation system. When one of the translator nodes fails, we can just remove the failed node from the system. Since there is no shared session information among translator nodes, the service is kept by the rest of the translator nodes without any recovery operation.

Figure 2 shows the overview of the actual system we are operating. The virtual machine resource management system located at the bottom of the figure is the WIDE Cloud Controller (WCC) (WIDE project, 2011) developed by the WIDE project. The two network operation centers (NOC) located at Tokyo and Osaka are connected by a wide area layer 2 network technology. The layer 2 network is a 10Gbps fiber line dedicated to the WIDE project network operation. There are two layer 3 networks in Tokyo and Osaka NOC whose network address spaces are same. These two layer 3 networks are connected by the layer 2 link using VLAN technology. The translator nodes are placed at each location. The routing information of the IPv4 addresses used to provide IPv4 connectivity to IPv4 clients is managed using the OSPFv3 routing protocol in the WIDE project backbone network. Since all the translator nodes advertise the same IPv4 address information using the equal cost strategy, incoming traffic is distributed based on the entry points of the incoming connections. The aggregated IPv4 routing information is advertised to the Internet using the BGP routing protocol. Any incoming IPv4 connection requests from the Internet are routed to the WIDE backbone based on the BGP information, routed to one of the translator nodes based on the OSPFv3 routing information, and finally routed to the corresponding virtual machine based on the static mapping table information stored in the translator nodes. The translation mechanism is described in section 3.2. Failure of either Osaka or Tokyo NOC will result in failure of the OSPFv3 routing information.
In this proposed system, we assume an IPv4 address and an IPv6 address are mapped one to one. Because the proposed system discussed in this paper is for IaaS providers providing server resources to their users, who are PaaS providers, SaaS providers, and/or ASPs, we cannot design the system to share one IPv4 address by several IPv6 server nodes. We also don’t consider implementing this requirement using application level proxy solution, since service providers have different requirements for their services, we cannot limit the communication layer function to application layers. This may be a problem considering that the IPv4 address is scarce resource. However, we need to use IPv4 address anyway if we want to support IPv4 clients. And also we need to map addresses one to one if we want to provide services without any modification on the client side. Our best effort to handle this issue is that we do not allocate IPv4 addresses to backend nodes to avoid non-necessary IPv4 use.

The version of an IP protocol used in the backend network can be either IPv4 or IPv6. In the implementation shown in this paper, we use IPv6 as an internal protocol for the backend network considering the trend of IPv6 transition of the whole Internet.

Figure 1 depicts the overview of the proposed system. We said that the frontend nodes provide dual-stack services to the client before, however precisely speaking, these frontend nodes do not have any IPv4 addresses. The mapping information between IPv4 and IPv6 addresses are registered to each IPv4-IPv6 translator node and shared by all the translator nodes. Since the mapping is done in the one to one manner, no translator nodes need to keep session information of ongoing communication. They can just translate IP packets one by one. This makes it possible to place multiple translator nodes easily to scale out the translation service when the amount of the traffic grows. This feature also contributes robustness of the translation system. When one of the translator nodes fails, we can just remove the failed node from the system. Since there is no shared session information among translator nodes, the service is kept by the rest of the translator nodes without any recovery operation.

Figure 2 shows the overview of the actual system we are operating. The virtual machine resource management system located at the bottom of the figure is the WIDE Cloud Controller (WCC) (WIDE project, 2011) developed by the WIDE project. The two network operation centers (NOC) located at Tokyo and Osaka are connected by a wide area layer 2 network technology. The layer 2 network is a 10Gbps fiber line dedicated to the WIDE project network operation. There are two layer 3 networks in Tokyo and Osaka NOC whose network address spaces are same. These two layer 3 networks are connected by the layer 2 link using VLAN technology. The translator nodes are placed at each location. The routing information of the IPv4 addresses used to provide IPv4 connectivity to IPv4 clients is managed using the OSPFv3 routing protocol in the WIDE project backbone network. Since all the translator nodes advertise the same IPv4 address information using the equal cost strategy, incoming traffic is distributed based on the entry points of the incoming connections. The aggregated IPv4 routing information is advertised to the Internet using the BGP routing protocol. Any incoming IPv4 connection requests from the Internet are routed to the WIDE backbone based on the BGP information, routed to one of the translator nodes based on the OSPFv3 routing information, and finally routed to the corresponding virtual machine based on the static mapping table information stored in the translator nodes. The translation mechanism is described in section 3.2. Failure of either Osaka or Tokyo NOC will result in failure of the OSPFv3 routing information.
In this proposed system, we assume an IPv4 address and an IPv6 address are mapped one to one. Because the proposed system discussed in this paper is for IaaS providers providing server resources to their users, who are PaaS providers, SaaS providers, and/or ASPs, we cannot design the system to share one IPv4 address by several IPv6 server nodes. We also don’t consider implementing this requirement using application level proxy solution, since service providers have different requirements for their services, we cannot limit the communication layer function to application layers. This may be a problem considering that the IPv4 address is scarce resource. However, we need to use IPv4 address anyway if we want to support IPv4 clients. And also we need to map addresses one to one if we want to provide services without any modification on the client side. Our best effort to handle this issue is that we do not allocate IPv4 addresses to backend nodes to avoid non-necessary IPv4 use.

The version of an IP protocol used in the backend network can be either IPv4 or IPv6. In the implementation shown in this paper, we use IPv6 as an internal protocol for the backend network considering the trend of IPv6 transition of the whole Internet.

Figure 1 depicts the overview of the proposed system. We said that the frontend nodes provide dual-stack services to the client before, however precisely speaking, these frontend nodes do not have any IPv4 addresses. The mapping information between IPv4 and IPv6 addresses are registered to each IPv4-IPv6 translator node and shared by all the translator nodes. Since the mapping is done in the one to one manner, no translator nodes need to keep session information of ongoing communication. They can just translate IP packets one by one. This makes it possible to place multiple translator nodes easily to scale out the translation service when the amount of the traffic grows. This feature also contributes robustness of the translation system. When one of the translator nodes fails, we can just remove the failed node from the system. Since there is no shared session information among translator nodes, the service is kept by the rest of the translator nodes without any recovery operation.

Figure 2 shows the overview of the actual system we are operating. The virtual machine resource management system located at the bottom of the figure is the WIDE Cloud Controller (WIDE project, 2011) developed by the WIDE project. The two network operation centers (NOC) located at Tokyo and Osaka are connected by a wide area layer 2 network technology. The layer 2 network is a 10Gbps fiber line dedicated to the WIDE project network operation. There are two layer 3 networks in Tokyo and Osaka NOC whose network address spaces are same. These two layer 3 networks are connected by the layer 2 link using VLAN technology. The translator nodes are placed at each location. The routing information of the IPv4 addresses used to provide IPv4 connectivity to IPv4 clients is managed using the OSPFv3 routing protocol in the WIDE project backbone network. Since all the translator nodes advertise the same IPv4 address information using the equal cost strategy, incoming traffic is distributed based on the entry points of the incoming connections. The aggregated IPv4 routing information is advertised to the Internet using the BGP routing protocol. Any incoming IPv4 connection requests from the Internet are routed to the WIDE backbone based on the BGP information, routed to one of the translator nodes based on the OSPFv3 routing information, and finally routed to the corresponding virtual machine based on the static mapping table information stored in the translator nodes. The translation mechanism is described in section 3.2. Failure of either Osaka or Tokyo NOC will result in failure of the OSPFv3 routing information.
Figure 1: The overview of the proposed IaaS system

In this proposed system, we assume an IPv4 address and an IPv6 address are mapped one to one. Because the proposed system discussed in this paper is for IaaS providers providing server resources to their users, who are PaaS providers, SaaS providers, and/or ASPs, we cannot design the system to share one IPv4 address by several IPv6 server nodes. We also don’t consider implementing this requirement using application level proxy solution, since service providers have different requirements for their services, we cannot limit the communication layer function to application layers. This may be a problem considering that the IPv4 address is scarce resource. However, we need to use IPv4 address anyway if we want to support IPv4 clients. And also we need to map addresses one to one if we want to provide services without any modification on the client side. Our best effort to handle this issue is that we do not allocate IPv4 addresses to backend nodes to avoid non-necessary IPv4 use.

The version of an IP protocol used in the backend network can be either IPv4 or IPv6. In the implementation shown in this paper, we use IPv6 as an internal protocol for the backend network considering the trend of IPv6 transition of the whole Internet.

Figure 2: The actual network configuration of the proposed IaaS system implemented and operated in the WIDE project operation network

When one of the translator nodes fails, we can just remove the failed node from the system. Since there is no shared session information among translator nodes, the service is kept by the rest of the translator nodes without any recovery operation.

Figure 2 shows the overview of the actual system we are operating. The virtual machine resource management system located at the bottom of the figure is the WIDE Cloud Controller (WCC) (WIDE project, 2011) developed by the WIDE project. The two network operation centers (NOC) located at Tokyo and Osaka are connected by a wide area layer 2 network technology. The layer 2 network is a 10Gbps fiber line dedicated to the WIDE project network operation. There are two layer 3 networks in Tokyo and Osaka NOC whose network address spaces are same. These two layer 3 networks are connected by the layer 2 link using VLAN technology. The translator nodes are placed at each location. The routing information of the IPv4 addresses used to provide IPv4 connectivity to IPv4 clients is managed using the OSPFv3 routing protocol in the WIDE project backbone network. Since all the translator nodes advertise the same IPv4 address information using the equal cost strategy, incoming traffic is distributed based on the entry points of the incoming connections. The aggregated IPv4 routing information is advertised to the Internet using the BGP routing protocol. Any incoming IPv4 connection requests from the Internet are routed to the WIDE backbone based on the BGP information, routed to one of the translator nodes based on the OSPFv3 routing information, and finally routed to the corresponding virtual machine based on the static mapping table information stored in the translator nodes. The translation mechanism is described in section 3.2. Failure of either Osaka or Tokyo NOC will result in failure of the OSPFv3 routing information.
SYSTEM EVALUATION

3.2 Design and Implementation of the Translator Software

The server nodes in the IaaS backend system will see all the communication from their clients as IPv6 connections, since the backend system is operated in IPv6 only. All the IPv4 traffic from clients are intercepted by the translator nodes and converted to IPv6 traffic.

Figure 3 shows the translation procedure used in the translator software. IPv4 address information bound to IPv6 addresses of servers is stored in the mapping table. Based on the table, the destination address of the IPv4 packet from client nodes is converted to the proper IPv6 server address. The source address of the IPv6 packet is converted to a pseudo IPv6 address by embedding the IPv4 address of the client node to the lower 32-bit of the pseudo IPv6 address. The pseudo IPv6 address is routed to the translator node inside the backend network. The reply traffic from the server side is also converted with the similar but opposite procedures.

The mechanism has been implemented using the tun pseudo network interface device that is now provided as a part of the basic function of Linux and BSD operating systems, originally developed by Maxim Krasnyansky.

For the reverse direction, the similar procedure is applied. Since the translator nodes advertise the routing information of the pseudo IPv6 address that includes the IPv4 address of the client node is advertised to the IaaS backend network, the translator nodes receive all the reverse traffic.

This translator software is published as map646 open source software (Keiichi Shima and Wataru Ishida, 2011), and anyone can use it freely.

4 System Evaluation

We implemented the proposed IaaS system as a part of the WIDE project service network as shown in figure 2. By focusing on IPv6-only operation, we could be free from IPv4 network management.

For the redundancy, we located two translator nodes in different locations of the WIDE project core network. We sometimes stop one of them for maintenance. In that case, the other running node is working as a backup node. We confirmed that the redundancy mechanism is working automatically in a real operation.

The incoming traffic to IPv4 addresses are load-balanced based on the BGP and OSPFv3 information. For the outgoing traffic, currently a simple router preference mechanism is used to choose an upstream router from IPv6 servers. We are considering using more advanced traffic engineering methods, such as destination prefix based routing in near future.

Figure 4 shows the RTT measurement result from an IPv4 client to an IPv6 server. Initially, both translator nodes are running. At time 35, we stopped the router advertisement function of translator in Tokyo. The traffic from outside to the cloud network still goes to Tokyo node, but the returning traffic will be routed to Osaka. At around time 65, we stopped routing function of the Tokyo node. After this point, all the traffic goes to Osaka and returns from Osaka. We restarted the routing function of Tokyo node, and the system returned to normal operation.

RA stopped at Tokyo
Translator stopped at Tokyo
RA and translator recovered
TRANSLATOR PERFORMANCE

Test network configurations

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Translation (4&gt;6)</td>
<td>![IPv4 Client] - IPv4 - IPv6 - IPv6 Server</td>
</tr>
<tr>
<td>C2</td>
<td>Translation (6&gt;4)</td>
<td>![IPv6 Client] - IPv6 - IPv4 - IPv4 Server</td>
</tr>
<tr>
<td>C3</td>
<td>IPv4 forwarding</td>
<td>![IPv4 Client] - IPv4 - IPv4 Server</td>
</tr>
<tr>
<td>C4</td>
<td>IPv6 forwarding</td>
<td>![IPv6 Client] - IPv6 - IPv6 Server</td>
</tr>
</tbody>
</table>

Figure 6: RTT measurement result using the ping program

5 Performance Evaluation

The obvious bottleneck of the system is the translator nodes where all the traffic must go through with them. This section shows the evaluation result of the translation software.

5.1 Translation Performance

The performance evaluation is done with the four different configurations shown in figure 5. The configuration 1 and 2 (C1 and C2) are the translation cases using our translator software.

<table>
<thead>
<tr>
<th>Specification of nodes</th>
<th>Client/Server</th>
<th>Translator/Router</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Core2 Duo</td>
<td>Xeon L5630</td>
</tr>
<tr>
<td>Memory</td>
<td>4GB</td>
<td>24GB</td>
</tr>
<tr>
<td>OS</td>
<td>Linux 3.0.0-12-server</td>
<td>Linux 3.0.0-12-server</td>
</tr>
<tr>
<td>NIC</td>
<td>Intel 82573L</td>
<td>Intel 82574L</td>
</tr>
</tbody>
</table>

Figure 6: RTT measurement result using the ping program

Evaluation is done using two methods, one is the ping program to measure RTT, and the other is the iperf program to measure bandwidth. All the results in this experiment show the average value of 5 measurement tries

We didn't record standard deviation of these tries, since the results were stable.
SPECIFICATION OF EQUIPMENTS USED

Table 1: Specification of nodes

<table>
<thead>
<tr>
<th></th>
<th>Client/Server</th>
<th>Translator/Router</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Core2 Duo 3.16GHz</td>
<td>Xeon L5630 2.13GHz × 2</td>
</tr>
<tr>
<td>Memory</td>
<td>4GB</td>
<td>24GB</td>
</tr>
<tr>
<td>OS</td>
<td>Linux 3.0.0-12-server</td>
<td>Linux 3.0.0-12-server</td>
</tr>
<tr>
<td>NIC</td>
<td>Intel 82573L</td>
<td>Intel 82574L</td>
</tr>
</tbody>
</table>
RTT COMPARISON

- RTT degradation is around 0.07ms to 0.09ms worse

![Graph showing RTT comparison between Translation and Normal modes with values for C1, C2, C3, and C4 configurations.]
**TCP PERFORMANCE**

- The forwarding performance of translator is **1.5% to 1.6% worse** than normal forwarding.
UDP PERFORMANCE

- The forwarding performance of translator is **2.0% worse** than normal forwarding.
RELATED TECHNOLOGIES

• Application layer gateway
  • Pros: useful when complicated application protocol conversion is required
  • Cons: per application gateway is required
RELATOD TECHNOLOGIES

- Transport layer gateway
  - Pros: transparent from users
  - Cons: library upgrade (e.g. SOCKS64), DNS service upgrade (TRT and DNS64)
RELATED TECHNOLOGIES

• DNS64/NAT64 (IP layer approach)
  • Pros: transparent from users
  • Cons: DNS service upgrade
RELATED TECHNOLOGIES

• In theory, every translation technologies can be used as a translation component of our proposal

• Most of the current implementation doesn’t support our operation (IPv6 nodes as servers for IPv4 clients)

• And they have too rich function for our solution (such as one to many mapping function)
COMPARISON WITH NAT64

• Almost same or even better performance

![Graph showing RTT and throughput comparison between map646 and linuxnat64]
CONCLUSION

• Proposed a new style of IaaS operation based on IPv6 only network to reduce operation cost, and provide IPv4 compatibility with a 64 translator

• Verified redundant operation of the translation system

• Implemented a simple 64 one to one translator for the proposed IaaS system and evaluated its performance