4over6: Network Layer Virtualization for IPv4-IPv6 Coexistence

Yong Cui, Peng Wu, Mingwei Xu, Jianping Wu, Yiu L. Lee, Alain Durand, and Chris Metz

Abstract

IANA exhausted its IPv4 address space in 2011, and IPv6 is the next generation to replace IPv4. However, the IPv6 transition techniques are still immature and holding back the development of the next-generation Internet. Hence, IPv4-IPv6 coexistence is becoming increasingly imminent. During the coexistence period, the Internet will consist of IPv4-only, IPv6-only, and dual-stack segments. Both the network infrastructure and operations must support IPv4-only, IPv6-only, and dual-stack accordingly. This article develops a 4over6 virtualization architecture that virtualizes IPv4-only networks over IPv6-only networks. This architecture enables two IPv4-only segments to communicate over an IPv6-only network by using an IPv4-in-IPv6 tunnel. The architecture can be examined in different areas such as addressing schema, layer 3 routing, and packet forwarding. The 4over6 virtualization architecture is being standardized in IETF. Various implementations and deployment scenarios are actively discussed in the ISP and vendor communities.

The IPv4 world is facing a series of challenges, including address shortage, routing scalability, broken end-to-end property, and so on. Among them, the shortage of addresses has become extremely urgent, considering that IANA exhausted its remaining global address space, and the regional registries will soon follow. IPv6 is being developed as the fundamental network protocol for the next-generation Internet, with the enhancements of much larger address space, feasibility of hierarchical addressing and routing, improved forwarding efficiency, mobility support, and so forth. The IPv6 transition for the Internet has been discussed for over a decade. On one hand, the pressure from IPv4 address storage seriously threatens Internet growth. This threat encourages the Internet community to speed up IPv6 adoption. On the other hand, the majority of existing network infrastructure, Internet services and applications, and the majority of customers still remain in IPv4. As a result, both IPv4 and IPv6 are going to coexist for some period of time.

IPv4-IPv6 Coexistence Requirements

IPv4 and IPv6 are incompatible in design. The coexistence of two protocols in the same networks will pose many challenges. Most notably, an IPv4 network cannot communicate with an IPv6 network. Therefore, network operators are required to run two independent networks on the same infrastructure, by maintaining dedicated addressing schema, network routing, and data forwarding for each IP protocol. Different Internet service providers (ISPs), Internet content providers (ICPs), and users will adopt IPv6 at different paces. Some networks may choose to stay in IPv4; some may support only IPv6 or both. Despite the network complexity, end-to-end connectivity must be preserved: a user must be able to access Internet services regardless of which IP protocol is used underneath.

There are two types of techniques that may be used for IPv4-IPv6 coexistence: tunneling [3] and translation [4]. Tunneling is used to achieve IPv4-over-IPv6 or IPv6-over-IPv4 connectivity by encapsulation and decapsulation. Translation is used to achieve direct connection between IPv4 and IPv6 by translating IPv4 packets into IPv6 and IPv6 packets into IPv4. Translation requires address binding between IPv4 and IPv6, as well as address translation in application-layer protocols.

From the network perspective, following the end-to-end arguments [5], the network should be simple, and complex functions should be left to end users. Second, the TCP/IP model should not be violated, and cross-layer operations should be kept minimal. Third, massive cross-function between IPv4 and IPv6 should be avoided. Therefore, tunneling will be a better option than translation. From the end-user perspective, ISPs should provide both IPv4 and IPv6 global reachability. Realizing one of them by an IPv4-over-IPv6 or IPv6-over-IPv4 tunnel improves cost efficiency. However, legacy IPv4 users may not ever upgrade to support IPv6 and IPv6-over-IPv4 tunnel; instead, it makes more sense to require new IPv6 users to support IPv4 with IPv4-in-IPv6 tunneling and thereby communicate with legacy users through IPv4. Therefore, we adopt IPv4-over-IPv6 tunneling to solve the more significant side of the problem.

Network Virtualization for IPv4-IPv6 Coexistence

Network virtualization provides a feasible solution to meet the above requirements and achieve IPv4-IPv6 coexistence. The technology is applicable to supporting heterogeneous networks and technology innovations in nature [1]. In this application environment, IPv4 and IPv6 are identical for data forwarding. The forwarding engine of the router and switch vehicles examines the destination field in the IP header and forwards the packet based on the information in the forward information base (FIB). As for addressing and routing, as well as operations, administration, and maintenance (OAM), they must be treated differently and independently. Network-layer virtualization can provide the proper level of isolation to real-

0890-8044/12/$25.00 © 2012 IEEE

ize that. As a result, end-to-end connectivity can be built, as long as two communication ends join the homogeneous virtual networks which are globally interconnected.

In general, the virtualization for IPv4-IPv6 coexistence will depend heavily on the capability of the virtual network of one address family to build part of its entity on top of the virtual network of the other address family [2]. Considering that each ISP has its own schedule for IPv6 transition, the Internet is likely to be mixed with IPv4-only segments, IPv6-only segments, and dual-stack segments. As a result, the global interconnectivities of both the virtual IPv4 and IPv6 networks are only achievable if IPv4 virtual networks can traverse the IPv6-only segments and IPv6 virtual networks can traverse the IPv4-only segments. The solution is to use the heterogeneous segments as virtual "infrastructure" and build forwarding paths over them. This method will significantly improve network cost efficiency, for a great portion of the network can be IPv4-only or IPv6-only and does not require supporting both address families concurrently. This article develops a 4over6 virtualization architecture for IPv4-IPv6 coexistence, in which separate virtual networks are provided for IPv4 and IPv6 services over the same infrastructure, and the IPv4 virtual network can be built over the virtual infrastructure of IPv6 networks. This article analyzes the functional elements including addressing, routing, and forwarding, and also proposes solutions. In addition, we also present a series of ongoing 4over6 efforts, including protocol design, implementation, and deployment.

The rest of the article is organized as follows. We provide the topology and functional entities of the 4over6 virtualization architecture. We propose the addressing, routing, and forwarding mechanisms. Implementation efforts are described. We then conclude the article.

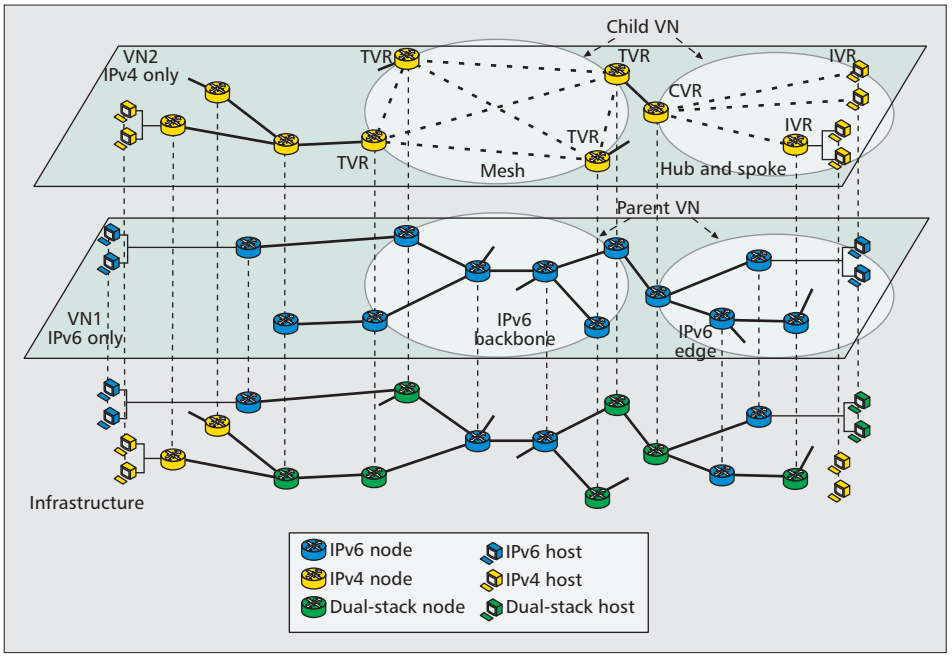

Figure 1. *4over6 virtualization architecture.*

The Topology of 4over6 Virtualization Architecture

4over6 Virtualization Topology

The topology of 4over6 virtualization architecture is shown in Fig. 1. There are three roles in this architecture:

- Infrastructure provider
- IPv6 service provider
- IPv4 service provider

The infrastructure provider manages and operates the infrastructure substrate. It provides transit services for both IPv4 and IPv6. The Internet was designed to communicate over diverse networks. From the data plane perspective, the principle of forwarding IPv6 packets is not different from forwarding IPv4 packets, so the same physical infrastructure could be utilized. In practice, service providers often use a large amount of overlapped equipment and links for dual stack forwarding.

The IPv6 service provider runs the IPv6 network on the infrastructure. The IPv6 network carries all the native IPv6 traffic generated by IPv6 end users, forwarding the data packets from source to destination. Meanwhile, it is also responsible for forwarding the IPv4 traffic passed down from the IPv4 service provider. This is achievable by encapsulating IPv4 packets into IPv6, so the IPv6 infrastructure can forward them while the original IPv4 packets stay unchanged in the payloads. The IPv6 network receives IPv4 traffic from one of the bridging nodes with IPv4 network, forwards the traffic to another bridging node, and hands the traffic back to IPv4. The IPv6 network works as the virtual infrastructure for the IPv4 networks that connect to it.

The IPv4 service provider manages the IPv4 network over the physical infrastructure and the virtual infrastructure. The IPv4 network carries the native IPv4 traffic generated by IPv4 end users and forwards it to the IPv4 destination. When forwarding the traffic, it can leverage either the physical infrastructure or the virtual infrastructure. A forwarding path between two bridging nodes on the virtual infrastructure creates a 4over6 instance. In Fig. 1, there are two typical 4over6 models: Mesh and Hub & Spokes. In the Mesh model, multiple IPv4-IPv6 bridging nodes establish 4over6 forwarding paths with each other to create a mesh-like structure for traffic interchanging. In the Hub & Spokes model, multiple IPv4-IPv6 bridging nodes establish 4over6 forwarding paths with one "hub" bridging node that provides centralized IPv4 access to them. Generally speaking, the Mesh model reflects the network connectivity problem and appears in the transit network, while the Hub & Spokes model reflects the host or LAN connectivity problem and appears in the edge network. So far the combination of these models covers all visible 4over6 demands.

Functional Entities

In 4over6 virtualization architecture, there are two basic functional entities: the bridging node and the IPv4 virtual link.

A bridging node is a dual-stack node that lies on a network device supporting both IPv4 and IPv6, and connects to both networks. The term *bridging* is used because it interconnects IPv4 and IPv6 network segments, and transfers traffic in between. The bridging node can natively route and forward IPv4 and IPv6 packets. It is also capable of transferring IPv4 traffic into IPv6 as well as transferring such traffic back into IPv4, by means of IPv4-in-IPv6 encapsulation and decapsulation. The ingress bridging node is responsible for delivering the encapsulated IPv6 packet to the egress bridging node.

According to the two 4over6 models, there are three types of bridging nodes: the bridging node in the Mesh model is called the transit virtual router (TVR); the hub node in the

Hub & Spokes model is called the concentrator virtual router (CVR); the spoke node is called the initiator virtual router (IVR). An IVR may lie on an end host in some cases.

A virtual link is the "link" between two bridging nodes in a 4over6 instance. Even though the link may traverse one or more IPv6 nodes in the middle, the IPv4 network recognizes it as a direct-connected link. In the Mesh model, the virtual link forms between two TVRs. In the Hub & Spokes model, the virtual link forms between the CVR and IVR (sometimes also between IVRs). The virtual link is realized by an IPv4-in-IPv6 tunnel. It connects two bridging nodes in IPv4 and makes the IPv6 path in the middle transparent to IPv4.

Since a 4over6 virtual topology would always be either Mesh or Hub & Spokes, the diameter of the topology is always one hop. Therefore, dedicated topology maintenance will not be necessary. The virtual network works as long as the bridging nodes are enabled to bridge the IPv4 routing and data packets over the virtual links.

Packet Flow

In the 4over6 architecture, the end-to-end path between two IPv4 users can be a combination of IPv4 virtual links and native links. The packet flow along native IPv4 links follows the regular IPv4 forwarding process. The packet flow in the virtual link follows the two 4over6 models.

In the 4over6 Mesh model, every TVR may receive IPv4 packets from the IPv4 network. The ingress TVR chooses the outgoing virtual link for the packet by using the termination of that link as the IPv4-in-IPv6 encapsulation destination. The encapsulated packet is sent over the chosen virtual link. Every IPv6 node along the virtual link forwards the encapsulated packet as a native IPv6 packet until the packet arrives on the link termination, which is the egress TVR. This TVR decapsulates the packet, extracts the original IPv4 packet from the payload, and hands the packet back to the IPv4 network for further forwarding.

In the 4over6 Hub & Spokes model, the CVR and every IVR may receive IPv4 packets from the IPv4 network. The CVR has a list of virtual links the termination of which is one of the IVRs. When the CVR receives an IPv4 packet, it chooses the outgoing virtual link for the packet by using one of the IVRs as the encapsulation destination. An IVR usually has only one virtual link, which terminates at the CVR. So when the IVR receives an IPv4 packet, it will use the CVR as the encapsulation destination. The IPv6 network connecting the CVR and IVRs will forward the encapsulated IPv6 packet as a native IPv6 packet. When the encapsulated packet arrives on the CVR or an IVR, it will decapsulate the IPv6 header and hand the IPv4 packet back to the IPv4 network for further forwarding.

Addressing, Routing, and Forwarding of 4over6 Virtualization Architecture

In this section, we describe three fundamental elements that are essential to realize the virtualization. They are

- 4over6 addressing
- 4over6 routing
- Packet encapsulation and decapsulation

4over6 Addressing

In the Hub & Spokes model, the IVR acquires a public IPv4 address for its end users to access IPv4 Internet. These IPv4 addresses are administrated by the CVR, which is the first, virtual hop of the IVR to access IPv4 Internet. So, the CVR assigns the IPv4 addresses to the IVRs.

The CVR could share IPv4 addresses to mitigate IPv4 address exhaustion. This is achieved by sharing a single IPv4 address among multiple IPv4 users and assigning different ports to each user. Two approaches can be applied here: the CVR can explicitly pre-assign a set of ports to each IVR, or allocate one port inside the CVR every time a new flow from the IVRs is encountered. The main difference between the two approaches is that port-set pre-assignment must be provisioned to the IVR, while the flow-based port allocation can be processed inside the CVR.

For the pre-assignment approach, the CVR explicitly provisions an address and port-set [6] to the IVR. One implementation is to leverage native IPv6 address provisioning methods by means of encoding the IPv4 address and port into an IPv6 address [7]. Another implementation is leveraging IPv4 address provisioning methods, and provisioning the address and port-set over the virtual link. For the flow-based port allocation, the usual implementation is network address translation (NAT) on the CVR. In this approach, the CVR manages the address resources in a NAT pool. The IVRs and IPv4 users use private IPv4 addresses for IPv4 communication. The CVR "NATs" the private address and port into public when reaching out to the IPv4 Internet. This is the standard NAT procedure.

The two approaches produce different effects. The first approach allocates public address and port-set to IVRs, so the IPv4 users own a public IPv4 address and port-set, and the CVR only tunnels the packets from/to the IPv4 users. This preserves end-to-end transparency. The second strategy translates a private IPv4 address to a public IPv4 address on the CVR. This breaks the end-to-end transparency. There is another consideration. Pre-assigning port-sets to IPv4 users brings fairness between users by giving each user a certain amount of port, but it does not consider actual usage. Contrarily, NAT on the CVR brings better port utilization than port-set pre-assignment, but it does not have a per-user port amount guarantee.

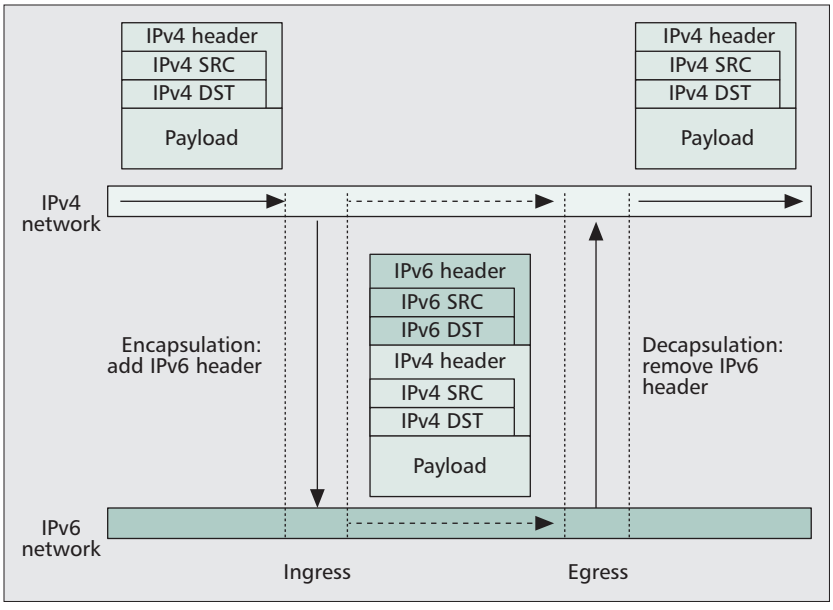

Figure 2. *Packet encapsulation and decapsulation.*

4over6 Routing

In 4over6 virtualization, the general isolation of IPv4 and IPv6 routing processes is required. In spite of that, some IPv6-related IPv4 routing issues should be solved to achieve 4over6 data forwarding, either across or inside IPv6.

The first routing issue is IPv4 routing across an IPv6 network. In the Mesh model, each TVR is connected to some IPv4 networks. The TVR is responsible for advertising the IPv4 prefixes of the networks to its TVR peers. This routing process distributes the reachability and encapsulation information. In the route advertisement of these IPv4 prefix, the next hop is the TVR's IPv6 address. There are two options:

- Embed the IPv4 prefix in a special IPv6 prefix and send it over the IPv6 routing protocol
- Extend the IPv4 routing protocol and enable it to use an IPv6 address in the next hop field

This requires running the IPv4 routing protocol over the IPv6 network between TVRs.

The second routing issue is virtual IPv4 routing in an IPv6 network. In the Hub & Spokes model, which encodes the IVRs' IPv4 address and port-set into IPv6 address, the IVRs actually use the encoded IPv6 addresses for IPv6 communication, particularly for IPv4-in-IPv6 tunnel. Therefore, this type of address should be reachable in the IPv6 network. This is achieved by native IPv6 routing, which should also by default aggregate these addresses. In general, virtual IPv4 addressing and routing are reflected by normal IPv6 addressing and routing.

In both cases, the IPv6 routing of the native IPv6 prefixes remains unchanged. The IPv4 prefixes in the 4over6 virtualization are independent of the IPv6 routing, which makes the IPv4 reachability decoupled from IPv6 routing as long as IPv6 reachability can be guaranteed. When IPv6 reachability is broken, the bridging nodes can detect it (e.g., through keep-alive messages in IPv6), mark it as a virtual link failure, and reflect this in IPv4 routing.

Packet Encapsulation and Decapsulation

In 4over6 virtualization, the packets cross the IPv4 and IPv6 virtual networks by encapsulation and decapsulation, as shown in Fig. 2. The basic principle of encapsulation and decapsulation is simple: when performing encapsulation, the ingress bridging node inserts an IPv6 header to the IPv4 packet. The encapsulated packet will have the destination IPv6 address of the egress bridging node. This packet will be forwarded natively in the IPv6 network. When performing decapsulation, the egress bridging node removes the IPv6 encapsulation header and extracts the original IPv4 packet. The egress virtual node will forward the IPv4 packet natively in the IPv4 network. All these data plane procedures can be implemented in hardware, so performance will not be an issue. There are two challenges, though, that need to be addressed.

The first challenge is encapsulation destination address. The ingress bridging node is responsible for deciding the IPv6 encapsulation destination address. This address should guide the packet toward the correct egress bridging node. One proposal is maintaining an encapsulation table for lookup. In the Mesh model, a mapping entry in this table would consist of an IPv4 prefix and an IPv6 TVR address, installed during 4over6 routing; in the Hub & Spokes model, a mapping entry would consist of an allocated IPv4 address and port-set against an IVR's IPv6 address, or a NAT entry along with an IVR's IPv6 address, depending on the addressing strategy. There is another proposal when the IPv4 address and port-set is encoded in the IPv6 address. In this case the encapsulation destination can be calculated from the IPv4 destination following the address-encoding algorithm. This solution does not require information maintenance, and the communication between two IVRs will not need a "hairpin" on the CVR.

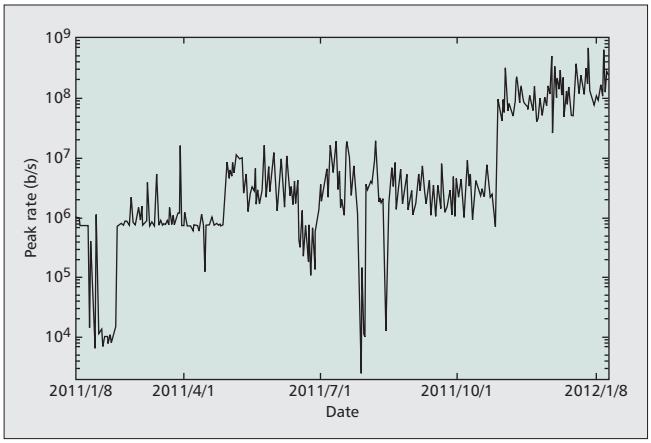

Figure 3. *Traffic volume of 4over6 mesh TVR in CNGI.*

The second challenge is the maximum transmission unit (MTU). Encapsulation decreases the effective MTU size of the IPv4 packet, which could easily result in unexpected fragmentation and reassembly. The simplest solution is to increase the MTU of the links in the IPv6 network for the encapsulation overhead. However, there are cases in which operators cannot do that; then they have no choice but to deal with fragmentation. One solution is to decrease the MTU on the IPv4 upstream link of the ingress bridging node so that all the fragmentation happens in IPv4 before the packet enters the tunnel, and the reassembly is left to the destination end users. The other solution requires the ingress node to handle the fragmentation in IPv6 after encapsulation, and the egress node to handle the reassembly in IPv6 before decapsulation. This way the tunnel becomes purely transparent. However, it brings strict requirements to the egress devices for handling significant amounts of reassembly.

Current Status of 4over6 Implementations

In the past couple of years, the authors have been designing and standardizing 4over6 protocols in the IETF Softwire Working Group. The authors proposed Softwire Mesh [8–10] as the protocol for the Mesh model. In Softwire Mesh the TVRs leverage MP-BGP over IPv6 to advertise IPv4 prefixes and form the encapsulation table. As for the Hub & Spokes case, different protocols are developed for different addressing strategies. For flow-based port allocation, the authors designed Dual-Stack Lite [11], in which the CVR uses the IVR's IPv6 address to create the NAT binding; for address and port-set preprovisioning, Public 4over6 [12] and Lightweight 4over6 [13] are proposed, with DHCPv4 over IPv6 as the signaling method; for embedding IPv4 information in an IPv6 prefix, the community has designed the 4RD protocol [14].

Efficient device implementation is necessary for these mechanisms, especially on TVR and CVR devices that must support vast amounts of flows, users, and traffic, and therefore require high capacity. On the control plane, 4over6 routing and addressing protocols such as Dynamic Host Configuration Protocol (DHCP) should be implemented in software. All these protocols can be developed by extending some existing protocols on current routers or servers. The encapsulation and decapsulation must be implemented by hardware to meet the capacity and performance requirements. In particular, the encapsulation table can be implemented by ternary content addressable memory (TCAM). The destination lookup procedure follows standard TCAM operation, and the TCAM can be regularly updated with the new entries from route advertisement or address provisioning. 4over6 virtualization adds no new requirement to TCAM implementation.

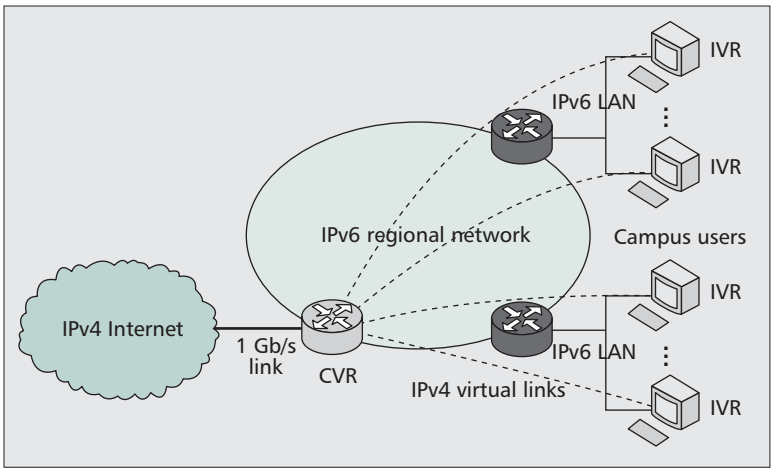

Figure 4. *Public 4over6 deployment in CNGI.*

We have actually deployed two 4over6 virtualization mechanisms, Softwire Mesh and Public 4over6 in China Next Generation Internet (CNGI). The Softwire Mesh deployment set up 100 TVRs in 100 campus networks. These 100 TVRs are connected through the China Education and Research Network II (CERNET2) IPv6 backbone, and 1 TVR amongst them peers to IPv4 Internet. This 4over6 Mesh is used to transport IPv4 traffic between different campus networks, as well as to provide IPv4 Internet connectivity to CERNET2 campus networks. This network has been running for over a year. Figure 3 shows the traffic on the TVR peering to IPv4 Internet. Through the past year, the traffic volume has significantly increased. In the Public 4over6 deployment, we set up a CVR in an IPv6 regional network, and provided the CVR with a 1 Gb/s uplink to IPv4 Internet, as shown in Fig. 4. A software tool was developed to provide IVR functions to Microsoft Windows 7 and Linux PC. Each user who had installed the software tool could obtain an IPv4 access from the CVR even though the PC was connected to an IPv6-only LAN. This system has been deployed in campus networks for more than six months, and we are in the process of adding Lightweight 4over6 function in the system.

Conclusion

This article describes a 4over6 virtualization architecture for IPv4-IPv6 coexistence, in which IPv4 and IPv6 services are implemented in separate virtual networks on the same infrastructure. The IPv4 virtual network can use the IPv6 network as virtual infrastructure to access IPv4 services. Nevertheless, the isolation between the IPv4 and IPv6 networks are sustained as a benefit of the virtualization.

The 4over6 virtualization architecture relies on IPv6 as the virtual infrastructure, and it solves the IPv4 connectivity problem in IPv6. As a result, it will significantly ease the adoption of IPv6. On the other hand, once the majority of Internet applications and ICPs turn to IPv6, this virtualization architecture can be removed from the network without influence on IPv6. The overhead of running 4over6 includes the operation costs of 4over6 addressing and routing and the hardware costs to support tunneling. However, since all the costs happen on a small portion of network nodes, deploying 4over6 virtualization is still much more lightweight than developing a full dual-stack Internet. It also significantly saves the limited IPv4 address resources compared to a dual-stack Internet.

4over6 virtualization can be implemented in various ways. The IPv4 address provisioning can be implemented by explicit allocation of pre-assigned address and port-set, or carrier-grade NAT on CVR. The IPv4 routing over IPv6 can be implemented by embedding IPv4 prefixes into IPv6 prefixes, or running IPv4 routing protocol over 4over6 tunnels. The virtual IPv4 routing leverages native IPv6 routing based on the addressing scheme. The 4over6 forwarding is achieved by encapsulation and decapsulation on the bridging node, in which the encapsulation destination can be either looked up from a maintained encapsulation table or calculated following the address encoding algorithm.

The 4over6 virtualization architecture is an IPv6 transition framework that provides guidelines for transition mechanism design, including Softwire Mesh, Dual-Stack Lite, and Public & Lightweight 4over6 protocols. These protocols have different pros and cons, as well as application scenarios. They have been standardized or are being actively worked on in the Internet Engineering Task Force (IETF). They can be implemented efficiently, and some of them have been deployed in CNGI.

Acknowledgments

This work is supported by the 973 Program of China (nos. 2009CB320501, 2009CB320503) and the NSFC project (nos. 61120106008, 60911130511).

References

[1] N. Niebert *et al.*, "Network Virtualization: A Viable Path towards the Future Internet," *Wireless Pers. Commun.*, 2008, vol. 45, pp. 511–20.
[2] N. M. M. K. Chowdhury and R. Boutaba, "Network Virtualization: State of the Art and Research Challenges," *IEEE Commun. Mag.*, July 2009.
[3] E. Nordmark and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers," IETF RFC 4213, Oct. 2005.
[4] F. Baker *et al.*, "Framework for IPv4/IPv6 Translation," IETF RFC 6144, Apr. 2011.
[5] J. Saltzer, D. Reed, and D. Clark, "End-to-End Arguments in System Design," ICDCS 1981, pp. 509–12.
[6] R. Bush *et al.*, "The Address plus Port (A+P) Approach to the IPv4 Address Shortage," IETF RFC 6346, Aug. 2011.
[7] O. Troan *et al.*, "Mapping of Address and Port," IETF draft, Nov. 2011.
[8] J. Wu *et al.*, "Softwire Mesh Framework," IETF RFC 5565, June 2009.
[9] J. Wu *et al.*, "The Transition to IPv6, Part I: 4over6 for the China Education and Research Network," *IEEE Internet Computing*, May 2006.
[10] Y. Cui *et al.*, "The Transition to IPv6, Part II: The Softwire Mesh Framework Solution," *IEEE Internet Computing*, Sept./Oct. 2006.
[11] A. Durand *et al.*, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion," IETF RFC 6333, August 2011.
[12] Y. Cui *et al.*, "Public IPv4 over Access IPv6 Network," IETF draft, Sept. 2011.
[13] Y. Cui *et al.*, "Lightweight 4over6 in Access Network," IETF draft, Oct. 2011.
[14] T. Murakam *et al.*, "IPv4 Residual Deployment on IPv6 Infrastructure – Protocol Specification," IETF draft, Sept. 2011.

Biographies

YONG CUI (cuiyong@tsinghua.edu.cn) is an associate professor at Tsinghua University, China. He has published two IETF RFCs on IPv6 transition technologies and co-chairs the IETF Softwire Working Group, which focuses on tunneling technology for IPv6 transition.

PENG WU is a Ph.D. candidate at Tsinghua University, China. His research interests include IPv6 transition and next-generation Internet.

MINGWEI XU is a full professor at Tsinghua University. His research interests include computer network architecture, high-speed router architecture, and network security. He is the co-author of one transition protocol RFC.

JIANPING WU is a full professor at Tsinghua University. As the pioneer of IPv6, he built the largest native IPv6 backbone in the world as the China Education and Research Network II, and received the Jonathan B. Postel Award from the Internet Society in2010. He co-authored four IPv6-related IETF RFCs.

YIU L. LEE works in the CTO office at Comcast.

ALAIN DURAND is a director in the IPG/CTO group at Juniper Networks. He is also the co-chair of IETF IPv6 transition WG, Softwire, and has published 14 RFCs.

CHRIS METZ is a technical leader in the Routing Technology Group for Cisco Systems. He is a specialist in IPv6 transition and has co-authored four IETF RFCs.