This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

PET: Prefixing, Encapsulation and Translation for IPv4-IPv6 Coexistence

Peng Wu*, Yong Cui*, Mingwei Xu*, Jianping Wu*, Xing Li*, Chris Metz† and Shengling Wang*
*Tsinghua University, Beijing, P.R.China
†Cisco Systems, Inc. San Jose, CA, USA

{weapon, cy, xmw, slwang}@csnet1.cs.tsinghua.edu.cn, {jianping, xing}@cernet.edu.cn, chmetz@cisco.com

***Abstract*—IPv6 transition problem has become one of the key factors which are holding up the development of the next generation Internet. Aiming to solve IPv6 transition problem, several translation and tunneling techniques have been proposed, satisfying the demand of IPv4-IPv6 interconnection and traversing respectively. However, translation techniques can't convert the semantic between IPv4 and IPv6 protocol perfectly, and they have serious limitations in operation complexity and scalability. Researchers tried to decompose, simplify these problems and improve translation techniques accordingly, but they've come to little achievement since these problems result from the very nature of translation. We propose a novel approach of choosing appropriate translation spot to solve these problems in a different angle, and hence make effective use of translation technique. Then we propose a framework for IPv4-IPv6 coexistence called PET, which integrates tunneling and translation to support both traversing and IPv4-IPv6 interconnection, and uses them properly to constitute communication models in different scenarios. Moreover, we put forward PET signaling method to achieve automatic translation spot election and translation context advertisement, as a complement to the framework.**

I. Introduction

IPv4 has achieved a great success in the past twenty years. However, IPv4 is facing serious problems including address space exhaustion and routing scalability problem. IPv6 is the network layer protocol for the next generation Internet, with much larger address space, hierarchical routing architecture and some other new features.

Currently many ISPs have already deployed IPv6 transit networks or updated their networks to dual-stack. However, due to the wide deployment of IPv4 services, resources and devices, end users still depend heavily on IPv4, they won't switch to IPv6 immediately. So IPv6 can't replace IPv4 quickly, they have to coexist for a long time. It's a critical subject to maintain the network availability during the transition period.

Various transition techniques have been proposed to support IPv4/IPv6 inter-operation and coexistence, which can be divided into two categories: tunneling and translation. Tunneling is used for External-IP(E-IP) over Internal-IP(I-IP) traversing. For example, we proposed softwire mesh technique, which builds dynamic tunnels with BGP to connect E-IP networks over I-IP transit[1, 2]. Tunneling is light-weighted and stateless, but it can't achieve IPv4-IPv6 interconnection: IPv4 and IPv6 are still separated. On the contrary, translation converts the semantic between IPv4 and IPv6. That is, IPv6 packet will turn into IPv4 packet after translation, and vice versa. Hence, translation enables IPv4 and IPv6 network to communicate. SIIT and IVI are stateless translation techniques[3, 4]. They use algorithms for IP/ICMP translation between IPv4 and IPv6. They assume that a global IPv4 address is reserved for each IPv6 host, so that address mapping can be done by adding and removing well-known IPv6 prefix. NAT-PT and NAT64 are stateful translation techniques[5, 6]. They maintain dynamic mappings of (address, port) tuples to translate the addresses of IPv6 hosts, and adopt the way of adding/removing IPv6 prefix to translate the addresses of IPv4 hosts.

However, translation has several technical issues[7]. The issues such as disruption of application protocols embedding IP addresses, information loss and misunderstanding of DNS-ALG are applicable to all translation techniques; the issues like address-mapping state persistence, inability to redirect traffic by address are applicable to stateful translation. Because of these issues NAT-PT was deprecated from RFC standards. Researchers have spent years trying to solve them. The DNS-ALG problem can be lightened by separating DNS-ALG from translator[6]. But the other issues are related to the very nature of translation, and we have to stand them.

Unfortunately we have to point out some more limitations. The first one is scalability problem. Stateless translation has to consume IPv4 addresses for IPv6 hosts. It is not scalable since IPv6 address space is much larger than IPv4. Meanwhile, stateful translation requires translator to maintain dynamic (address, port) mappings. The number of the mapping entries is up to the quantity of traffic flow. So both stateless and stateful translation have scalability problem. The second one is operation complexity. Address and port substitution, IP/ICMP protocol translation, TCP/UDP checksum recomputing, application layer translation and fragmentation are all required to accomplish correct translation. It's unrealistic for a router to execute these per-packet operations while guarantee high speed forwarding, especially when application layer translation is included, since it demands network devices to execute per-application detecting-and-modifying procedures[8].

These limitations are also inherent disadvantages due to the principle of translation; but we can't stand them since they strongly restrict the application scope of translation. This paper proposes a novel approach to deal with them: locat-

Supported by National Major Basic Research Program of China (2009CB320501, 2009CB320503), NSFC (no. 60911130511, 60873252), CNGI Project (no. 2008-102), the National High Technology Development Program of China (No. 2008AA01A324).

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

ing appropriate spot to perform translation, where operation complexity is tolerable and scalability isn't a concern. In this way we can avoid the limitations while still achieving IPv4-IPv6 interconnection. Then we proposes a generic transition framework called PET to solve transition problems in various scenarios. PET uses tunneling and translation properly to fit different cases, and combines them when appropriate to utilize their respective advantages. Furthermore, we put forward the signaling method to achieve automatic translation spot election and translation context advertisement.

The rest of this paper is organized as follows. In Section II we bring forward the method of choosing appropriate translation spot. Section III provides the design of PET framework. We propose PET signaling for automatic translation in Section IV. Section V is a case study and evaluation of applying PET for IPv6-to-IPv4 communication. We give our conclusion in section VI.

II. Choosing Appropriate Translation Spot

Translation has high operation complexity and scalability issue as its inherent disadvantages. Luckily, by choosing appropriate spot to perform translation, we can diminish these disadvantages. We'll describe the method in this section.

A. Locating Translation Spot in Dual-Stack Transit Situation

The basic idea under this direction is to put translation devices in edge networks rather than in backbone. The reason is as follows. Due to their high traffic volume, routers in backbone networks only support routing and forwarding as basic functions, and their forwarding is implemented by hardware. It's not realistic to perform a real-time per-flow mapping and per-packet modification deep to application layer on them. On the contrary, traffic volume in edge networks isn't that high, and line-speed forwarding isn't necessary there. The cost of performing translation is acceptable in edge networks.

Intuitively translation should be done on IPv4-IPv6 border. So let's consider the situation with multiple IPv4-IPv6 borders first. A typical topology is shown in Figure 1(a). A dual-stack backbone is connected to IPv4 Internet through AFBR2(Address Family Border Router). One or more IPv6 edge networks are attached to the dual-stack backbone through other AFBRs, such as AFBR1. Hosts in IPv6 edge network want to communicate with IPv4, so translation is required.

Apparently, there are two spots which can perform translation: AFBR1 and AFBR2, both on IPv4-IPv6 border. When translation is performed on AFBR1, packets from IPv6 edge to IPv4 Internet go through the backbone as IPv4 packets after translation; Otherwise translation is performed on AFBR2, packets from IPv6 edge to IPv4 Internet go through the backbone as IPv6 packets before translation. If the backbone is small in size and attached with only several edge networks, then translation can be done on AFBR2 for unified administration and simplicity. If the backbone is large and attached with many client networks, then it's not scalable and efficient to perform translation on AFBR2. In this case we'd better choose AFBR1 instead. In general, we can locate translation spot based on network condition in dual-stack transit situation.

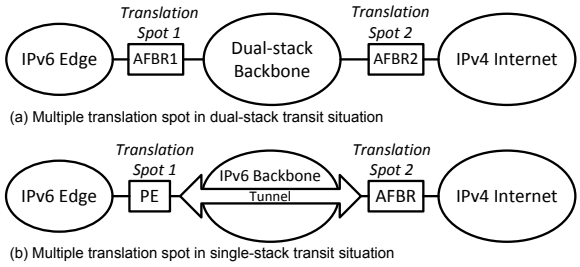

Fig. 1. locating translation spot in different situations

B. Locating Translation Spot in Single-Stack Transit Situation

We can go a step further here. What happens when the transit becomes single-stack? Then there is only one IPv4-IPv6 border spot. It seems that we have no choice: whether the border spot is in backbone or edge network, we have to perform translation there. However, actually we can perform translation in locations other than IP protocol border, as long as we hold enough knowledge to execute it. The only difficulty is forwarding: if translation is executed in advance(before packets reach the border), then translated packets can't be forwarded to the IP border spot because of the diversity in IP protocol; else translation is executed postponed(after packets pass the IP border), then to-be-translated packets can't be forwarded from the IP border spot to the translation spot.

Actually, if we regard the protocol of the network between the border spot and the translation spot as I-IP, this is an E-IP over I-IP traversing problem: in the former case, I-IP packets are translated to E-IP in I-IP network, so an E-IP over I-IP traversing path is required to forward translated packets to reach E-IP network; in the latter case E-IP packets are translated to I-IP in I-IP network, so when to-be-translated packets enter I-IP network, an E-IP over I-IP traversing path is required to forward them to the translation spot. In both cases we can use a tunnel connecting the border spot and the translation spot to build the forwarding path. Our design brings an extra tunnel here, nevertheless the cost of a tunnel is much lower than translation process itself, and hence acceptable.

Figure 1(b) provides a typical topology of performing translation in single-stack transit situation. Unlike the topology above, the backbone network here is IPv6 only, and the spot between an IPv6 edge network and the backbone is not an IPv4-IPv6 border. We call the router on that spot a PE(Provider Edge router). There're two ways to perform translation in this situation, too. The first one is translating on the AFBR directly, and the second one is translating on PE and using a tunnel to traverse the IPv6 backbone. When applying the second way, Every PE builds a tunnel with the AFBR to forward the translated IPv4 packets to the AFBR, and to receive the to-be-translated IPv6 packets from the AFBR. In general, we can locate the translation spot based on network condition in single-stack transit situation, too.

III. PET Framework

A. An Overview of PET Framework

Based on the improvement of performing translation on appropriate spot, we propose a framework for IPv4-IPv6 coexis-

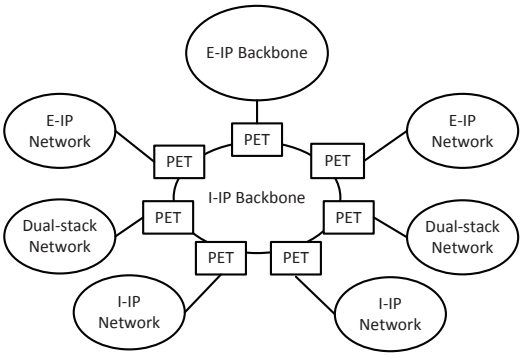

Fig. 2. PET Framework Topology

tence called PET(short for Prefixing, Encapsulation and Translation) in this section. PET is a generic solution for IPv4/IPv6 transition, which can achieve both E-IP over I-IP traversing and IPv4-IPv6 interconnection(for notation unification we use E-IP and I-IP to represent IPv4-IPv6 interconnection as well below). Both tunneling and translation are used properly in PET to fit different scenarios, and they can collaborate to utilize their respective advantages when necessary.

Figure 2 presents the design topology of PET framework. There is an I-IP backbone in the center working as a transit, and a number of E-IP/I-IP client networks attached to the backbone. There is also a connection between the I-IP backbone and an E-IP backbone which also have E-IP client networks. PET can achieve end-to-end communications between any two client networks of I-IP backbone, and between client network of I-IP backbone and network behind E-IP backbone. We believe this topology covers most network transition scenarios.

B. Basic Components of PET

1) Encapsulation: the Encapsulation component includes all the tunneling operations on data plane, such as encapsulation, decapsulation, fragmentation and so on. Based on these operations, packets from E-IP network are encapsulated, sent across I-IP transit network, decapsulated and reach another E-IP network. Prefix advertisement on control plane is required to support correct forwarding along the path.

2) Translation: the Translation component includes all the translation operations on data plane, such as address mapping and protocol translation, fragmentation and so on. Here protocol translation includes network layer translation and application layer translation. Based on these operations, packets are translated between IPv4 and IPv6.

3) Prefixing: the Prefixing component stands for all prefix-related control plan operations for tunneling and translation. In tunneling, prefixing includes routing prefix advertisement, tunnel endpoint discovery, tunnel signaling, etc. For example, in softwire mesh mechanism, MP-BGP is extended to advertise E-IP routes and to distribute tunnel parameters before data forwarding. In translation, prefixing includes address mapping algorithm, routing prefix advertisement and prefix configuration. For example, in IVI, address mapping is done by adding/removing ISP specific IPv6 prefix, this prefix and the more specific prefixes of client networks are advertised to guide the traffic between client networks and translation spot.

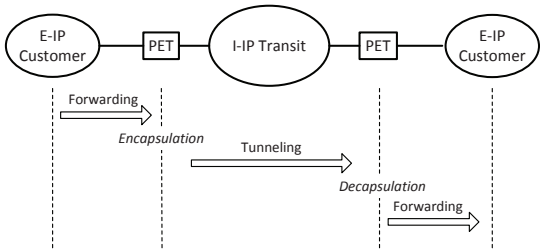

Fig. 3. PET communication model design for E-IP=>I-IP=>E-IP

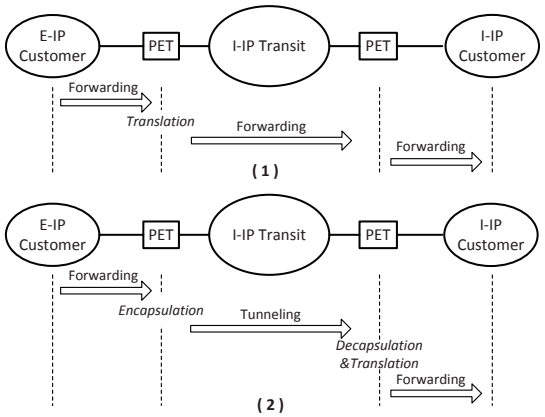

Fig. 4. PET communication model design for E-IP=>I-IP=>I-IP

C. Communication Mode Design for Different Scenarios

We can extract three distinct scenarios from figure 2, which cover all the communication demands in the topology of PET framework. We offer the design of communication models for these three scenarios as follows.

1) E-IP=>I-IP=>E-IP: This is the traversing scenario: an E-IP network want to communicate with another E-IP network across the I-IP transit. PET adopts tunneling to handle this scenario. The operations and procedure are shown in figure 3. The PET device on the border of source client network encapsulates the E-IP packets and forward them across the I-IP transit to the PET device on the border of the destination client network. There the packets are decapsulated and forwarded to the destination in E-IP customer network.

2) E-IP=>I-IP=>I-IP: In this scenario, an E-IP customer network wants to communicate with an I-IP customer network across the I-IP transit. There are two ways to deal with this scenario, both shown in figure 4. The first is translation plus forwarding and the second is tunneling plus translation.

Taking the first way, when an E-IP packet arrives at the PET device on the border of the source client network, it will be translated into an I-IP packet and forwarded directly through the I-IP transit to the destination in I-IP client network. Taking the second way, when an E-IP packet arrives at the first PET device, it will be encapsulated as an I-IP packet and forwarded through the I-IP transit to the second PET device. Translation from E-IP to I-IP is performed on the second PET device after the I-IP packet gets decapsulated to E-IP. Then the packet will be forwarded to the destination host in I-IP client network.

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

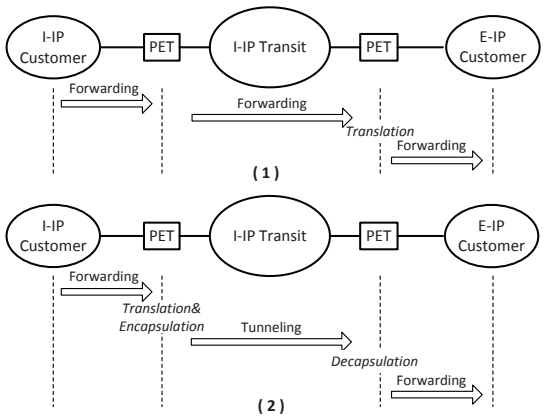

Fig. 5. PET communication model design for I-IP=>I-IP=>E-IP

The two communication models both work. The choice is made once the translation spot is decided. There's no conclusion on which way is better, we should choose the model based on concrete situation. Factors like the size of I-IP client network and I-IP backbone, the capability of the PET devices performing translation will all affect the choice.

3) I-IP=>I-IP=>E-IP: This is the reverse direction of last scenario: an I-IP customer network wants to communicate with an E-IP customer network across the I-IP transit. There are also two communication models for this scenario, both shown in figure 5. The first is forwarding plus translation, and the second is translation plus tunneling.

Taking the first way, an I-IP packet is forwarded directly from I-IP client network to I-IP transit and reach the second PET device which is on IP protocol border. There the packet will be translated into an E-IP packet and forwarded to the E-IP destination. Taking the second way, when an I-IP packet arrives at the first PET device, it will be translated in advance into E-IP, and then get encapsulated as an I-IP packet to pass through the I-IP backbone. When the encapsulated packet reach the second PET device, it will be decapsulated to E-IP and forwarded to the destination. Here both models work, too.

IV. PET Signaling for Automatic Translation

A. Translation Spot Election

Now we are clear that it's important to locate appropriate translation spot. Since PET framework support translation on multiple spots, we should guarantee it's performed on the right spot. This can be done by a signaling process between PET devices, carried by protocols like ICMP or BGP.

The remaining question is, what's the criterion to decide which spot is an appropriate translation spot? According to our analysis in section II, PET device whose performance is better and where there's no scalability issue to execute translation is preferred to be translation spot.

We propose the concept of Translation Preference (TP) to represent the appropriateness to perform translation. The value of TP can be inferred by bandwidth, traffic volume, traffic load, etc. The PET device whose bandwidth/traffic volume is high will get a low TP value, because this device is probably locating in a backbone where it's not effective to perform translation. More intuitively, if the network behind a PET device is large, it'll get a low TP value. If the traffic load is relatively low in a period, then this PET device has extra capability to perform translation, it should raise its TP value. The value may be affected by administrators' policy, too. For example, some network administrators are not willing to provide translation service for certain address blocks. As a qualitative description, we won't give an exact formula to calculate TP value.

After determining their own TP values, PET devices can exchange TP values with each other. Then each PET device can decide independently which one along the path should perform translation. The communication model is decided simultaneously: whether we need an extra tunnel, and if we do, which devices are the tunnel endpoints.

B. Translation Context Advertisement

There is some more information that PET devices need to learn from others through signaling process. It's the necessary context to execute translation. For stateless translation it's the mapping prefix, and for stateful translation it's the address pool used for dynamic address mapping.

For example, in an IPv6=>PET1=>IPv6=>PET2=>IPv4 scenario, if we use stateless translation, then PET2 should tell PET1 the prefix for IPv4-IPv6 address mapping when PET1 performs the translation; if we use stateful translation, then PET2 should tell PET1 the IPv4 addresses PET1 can use for address mapping when PET1 performs the translation. PET1 itself can't decide this information since it's not in the IPv4-IPv6 border and hence doesn't own the whole IPv6 address block or the IPv4 address block.

The translation context advertisement can also be achieved by BGP, ICMP or other protocols. Whether to advertise the context is decided by the result of translation spot election.

V. PET64: A Case Study and Evaluation

A. PET64 with IVI and softwire

In this section we'll present an instance to apply PET framework. We restrict the scenario into the situation where IPv6-only host communicates with IPv4-only host/server in IPv4 Internet through an IPv6 backbone. We call the PET solution under this scenario PET64. Figure 6 shows the scenario of PET64. It is a specialization of *I-IP=>I-IP=>E-IP* scenario in figure 5, so the communication can be achieved by either forwarding plus translation or translation plus tunneling.

We adopt softwire mesh as tunneling technique and IVI as translation technique. PET devices are deployed between IPv6 backbone and client networks(PET1 in figure 6), and on border of IPv6 and IPv4(PET2 in figure 6). Both PET1 and PET2 are capable of performing softwire and IVI, i.e., they're routers with IVI translating function as well as softwire endpoint.

In control plan PET signaling is used to decided the communication model of PET64. As is shown in the figure 6, PET1 and PET2 exchange their TP to negotiate the translation spot for IPv6 client network behind PET1. If TP2 has higher value, PET2 will perform IVI translation. Else PET1 will perform IVI

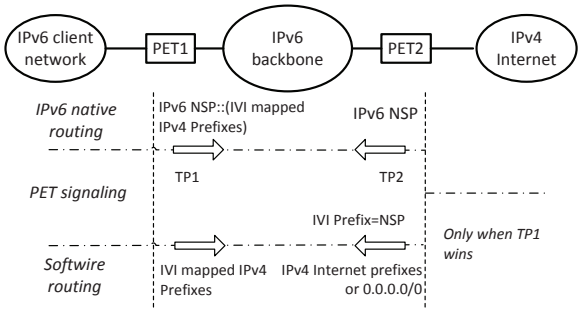

Fig. 6. PET64: control plan process

translation, in this case PET1 should learn from PET2 the IVI mapping prefix which is an IPv6 network specific prefix(NSP).

Besides, Prefixing is required in control plan. PET1 and PET2 should execute IVI-related IPv6 prefix routing in IPv6 backbone to guide the IPv6-IPv4 traffic. PET2 advertises the IVI mapping prefix, i.e.,the IPv6 NSP, while other PET devices like PET1 advertise IVI-mapped IPv6 prefixes which are composed of NSP followed by the IVI-mapped IPv4 prefixes. Moreover, when PET1 performs translation, a softwire between PET1 and PET2 is involved. Softwire routing and signaling will be executed through BGP. PET1 will advertises the IVI-mapped IPv4 prefixes of its client network to PET2, and PET2 will advertise the prefixes from IPv4 Internet or just 0.0.0.0/0 to PET1. Then the tunneled forwarding is guaranteed.

Based on these efforts on control plan, the data plan can work successfully. If PET signaling decides that PET2 performs translation, then the data plan procedure is a standard IVI procedure. Else PET1 is the translation spot, then a softwire tunnel is set up. The traffic will be translated in PET1 and reach PET2 through the tunnel.

B. A Preliminary Evaluation of PET64

To our knowledge, there're no routers that have implemented high-performance IPv4-IPv6 tunneling and translation, so it's difficult to carry out experiments and provide a systematic evaluation on the method of choosing translation spot and on PET. Nevertheless we'll provide a preliminary evaluation.

In routers, The whole data plan of softwire, and the network-layer translation of IVI can be implemented by hardware. However, the application layer translation has to be implemented by software. According to our prototype of translation on linux, software translation speed using mainstream PC can only reach 400-500Mbps. We check that the CPU and memory of mainstream routers can't even match mainstream PC, so their software translation speed will be even lower, say it's r.

Assume there are n IPv6 client networks in PET64 scenario, and the hardware processing speed of PET1 is p, while for PET2 it's $q(q > p >> r)$. If PET2 performs translation like traditional IVI, then the volume of translated traffic in this scenario can be $min\{q, r\} = r$. If PET64 is adopted and PETs on client network borders(such as PET1) are selected as translation spots, then the total volume of translated traffic can be up to $min\{q, n*r\}$, which is $min\{q/r, n\}$ times the number in traditional IVI. Say r=0.5Gbps, q=20Gbps and n=50, we'll have the result 40. Obviously using PET we can improve the translation throughput of networks while avoid the limitations of translation. So far this is just a straightforward analysis, we'll carry out better evaluation in the future with coming tunneling and translation devices by venders.

VI. Conclusion

In this paper, we point out the limitations of translation as IPv4-IPv6 coexistence technique in operation complexity and scalability, and we propose the method to deal with these problems by locating appropriate spot to perform translation. If the translation spot isn't located on IPv4-IPv6 border, we bring in an extra tunnel to build the forwarding path. By performing translation on appropriate spot, we can make effective use of translation technique, achieve high-speed IPv4-IPv6 interconnection and avoid the operation complexity and scalability issues.

Based on the method of locating appropriate translation spot, we propose PET as a framework for IPv4-IPv6 coexistence, which solves both traversing and IPv4-IPv6 interconnection by integrate tunneling, translation, and the prefix-related control plan operations. PET can provide proper communication models for different scenarios to achieve IPv4-IPv6 coexistence and inter-operation. In some scenarios, tunnel and translation are used together to achieve the highest benefit. PET is a generic framework for network-side IPv4/IPv6 transition.

Moreover, when performing translation, we propose a signaling method for automatic translation spot negotiation and translation context advertisement. Through this signaling process, PET devices along the path can elect the translation device automatically and offer the device enough knowledge to perform translation. We present a case study and preliminary evaluation to show the availability and benefits of choosing appropriate translation spot and PET framework.

References

[1] J.Wu, Y.Cui, C.Metz, and E.Rosen, "Softwire Mesh Framework," 2009, IETF RFC 5565.

[2] Y. Cui, J. Wu, X. Li, M. Xu, and C. Metz, "The Transition to IPv6, Part II: The Softwire Mesh Framework Solution," *IEEE Internet Computing*, vol. 10, no. 5, pp. 76–80, Sep/Oct 2006.

[3] E.Nordmark, "Stateless IP/ICMP Translation Algorithm," 2000, IETF RFC 2765.

[4] Y. Zhu, M. Chen, H. Zhang, and X. Li, "Stateless mapping and multiplexing of ipv4 addresses in migration to ipv6 internet," in *GLOBECOM*, 2008, pp. 2248–2252.

[5] G.Tsirtsis and P.Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)," 2000, IETF RFC 2766.

[6] M.Bagnulo, P.Matthews, and I.van Beijnum, "NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers," 2009, IETF draft.

[7] C.Aoun and E.Davies, "Reasons to Move the Network Address Translator - Protocol Translator(NAT-PT) to Historic Status," 2007, IETF RFC 4966.

[8] F.Baker, X.Li, C.Bao, and K.Yin, "Framework for IPv4/IPv6 Translation," 2009, IETF draft.