Migration and Co-existence of IPv4 and IPv6 in Residential Networks

Pekka Savola
CSC/FUNET
`Pekka.Savola@funet.fi`

Abstract

There has been an increased, and increasing, interest in IPv6; it has also already been deployed outside of academic and test networks. Here, the transition to IPv6 and co-existance of it with IPv4 in the residential networks will be examined. Technologies such as 6to4 and possibly Teredo are seen as the primary connectivity mechanisms, and NAT-PT and TCP/UDP relay as the main translator mechanisms; depending on usage scenarios, protocol translation may not be necessary, though. It will be seen that mechanisms are ready or almost ready, and that new applications – mainly peer-to-peer ones – will drive the deployment of IPv6 especially in the residential networks.

1 Introduction

IPv6 [1] is the new version of Internet Protocol which offers quite a few enhancements and possibilities. The transition [13] to it has already begun even in operational networks, and residential networks such as home networks are soon to follow.

This paper discusses reasons to start using IPv6, different connectivity and translation mechanisms, and migration and co-existence approaches; introduction to IPv6 as such is out of scope.

The primary focus of this paper is to briefly introduce the mechanisms ("toolbox") that could be used, but more importantly, discuss why or why not certain mechanisms or approaches apply to the residential network requirements. Some possible ways to build IPv4/IPv6 and to place different network elements are shown in figure 1.

Residential networks have a few things that are often common; these will set some requirements to the mechanisms and approaches:

– residential customers usually don't want to pay any more than they absolutely have to: they choose the cheapest equipment and connectivity mechanisms, don't want to pay extra for IPv6, etc.

– access routers and such are usually low-end (in the price range of 100 EUR rather than 500 EUR), no native IPv6 in sight soon; it is probable that some form of tunneling is to be used

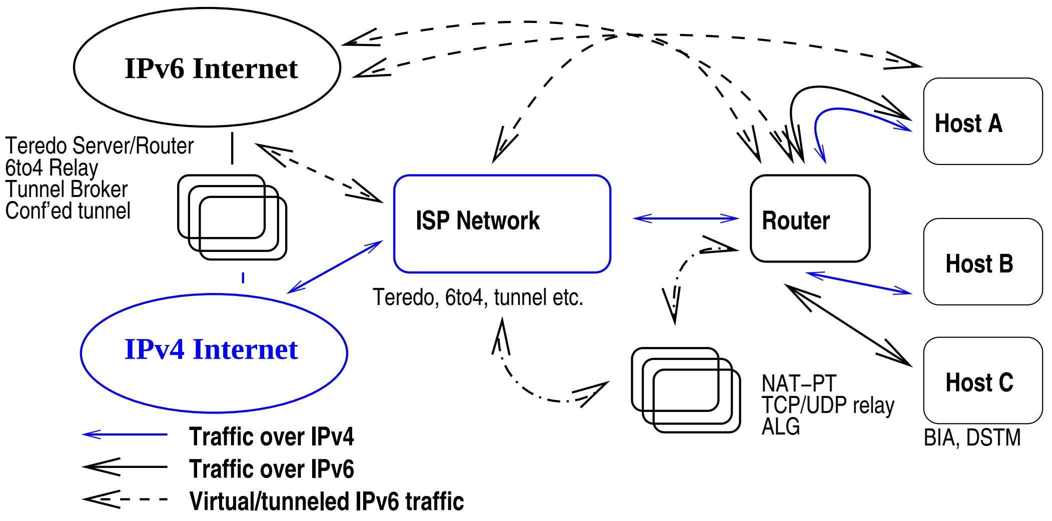

Figure 1: Some possible ways to build IPv4/IPv6 network

- residential networks are beginning to be "always-on"
- residential users have increased concern about privacy and often about security
- an important feature of a residential network is the ability to access it from elsewhere (e.g. work)
- residential environments have many devices that will be networked using Internet Protocol

So, when a user chooses his service provider, IPv6 will often be a deciding factor only when two roughly equal solutions are on the table; but that will be enough of a motivator for operators to start offer at least some form of IPv6 service.

2 Reasons for Starting to Use IPv6

IPv6 is not the solution to every possible problem and should not be over-hyped in that regard. It does offer some significant benefits in some aspects, and cleaner design in others, but is only a so-called "red herring" in some others (like Quality of Service at the moment).

New features of IPv6 enable the creation of new types of applications; another issue entirely is when the "critical mass" of the user base is achieved and IPv6 applications will be extensively developed and deployed. Most of the features are due to the restored end-to-end connectivity (e.g. no NAT's in the middle). Examples of new applications include:

- integrated autoconfiguration, security and end-to-end transparency enable the easy use of low-cost network appliances [19]
- enhanced mobility support enhances the roaming capabilities and virtual home environments

- peer-to-peer VPN's enable secure, billable services
- peer-to-peer applications such as file sharing, distributed computing (e.g. GRID), VoIP, and videoconferencing are much easier

One could argue that all (or almost all) or these are already possible with IPv4: it is true, using e.g. DHCP, IPSEC, MIPv4, and NAT traversal mechanisms. However, the question is what is the the technically better, cleaner, and long term solution and which is only patchwork that has been lasting for close to 10 years now.

One could also argue that IPv4 NAT's being replaced by IPv4/6 protocol translators is not much of an improvement; this is also true, but it should be remembered that translation will only be needed when communicating with legacy systems – IPv6 to IPv6 works flawlessly.

It should be noted that due to end-to-end transparency and no NAT's, installing packet filters or firewalls at the edge of the residential network becomes even more crucial. This is no news though, and Free software which can do this has existed for years.

One of the primary motivations for being able to use IPv6 is remote connectivity to some home environment. In many cases, this was not possible with NAT's; IPv6 appliances can be used from afar using IPv6 with IPsec without problems, as there are no address or protocol translations (which would break IPsec); here, we make the assumption that for remote connectivity, IPv6 could be required.

However, quite often we must also assume that the residential network may have hosts which support only IPv4: for example, legacy computers. Being able to use IPv6-only appliances from those might be valuable. Luckily enough, this is not necessarily a big problem: protocol translation will be required, but the question is where it must be done. Often it can be assumed that the residential network is rather secure (in contrast to insecure Internet), so if the translation is done at the edge of the network (e.g. DSL router, physically located at the residence), the network would still internally reasonably safe without IPsec.

So, the only really problematic scenario would be IPv4-only nodes in the Internet trying to securely access an IPv6-only host in the residential network. In many cases, this is a scenario that can be ignored as it really can't be done even now.

For example, consider an IPv6-only appliance like a video recorder (VCR): it can auto-configure itself and use IPsec for security. With proper firewall policy and set-up IPsec Security Associations, the appliance can be remotely accessed securely using IPv6. Also, the appliance could be accessed from the IPv4 nodes of the residential network (e.g. an old PC) reasonably securely if some internal device (e.g. the DSL router or a modern computer) performs the translation.

3 Obtaining IPv6 Connectivity

The first step in the migration to IPv6 is naturally obtaining connectivity from the residential network to the IPv6 Internet. Some basic mechanisms are considered and compared.

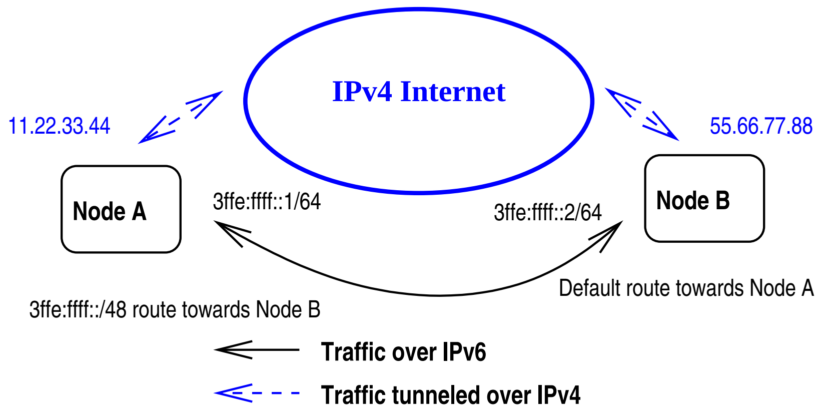

Figure 2: Example of a tunnel configuration

3.1 Configured Tunnels and Tunnel Broker

When no native IPv6 infrastructure exists between two points but there is IPv4 connectivity, tunneling IPv6 in IPv4 [2] can be used. This is a very common scenario in the early stages of the transition process.

When sending an IPv6 packet, it is encapsulated as payload in simple IPv4 packets with “41” as the protocol number. When receiving an IPv4 packet with protocol number “41”, the recipient can decapsulate it, discard the outer IPv4 header and process the inner IPv6 packet as normal; a tunnel is modeled as a one-hop link-layer from the perspective of IPv6.

Automatic tunneling mechanisms are discussed in other sections below; here, only configured or semi-configured (Tunnel Broker) mechanisms are explored.

When a tunnel is configured (see fig. 2 for an example), usually the following has to be known:

– link-layer (“IPv4”) addresses of the both parties
– global IPv6 addresses of the both parties used on the link (not absolutely necessary)
– which traffic should be routed on the link and how (e.g. by setting static routes)

In addition, from the ISP point-of-view, connectivity requests have to be processed and mails exchanged, documentation and continuous support provided for many customers, address ranges delegated and routed, and such: it can be seen that this could easily become a management nightmare for an ISP (especially if no additional fees are collected of IPv6 tunnels); the provider-customer interface is too broad.

In contrast, so-called Tunnel Brokers [10] can somewhat help in the management. The idea is that setting up the tunnels can be automated and requires, after setting up the “brokering” system, no management: for example, the users may be asked the required data in a WWW form, ready configuration provided for them, the tunnel configured in a router automatically.

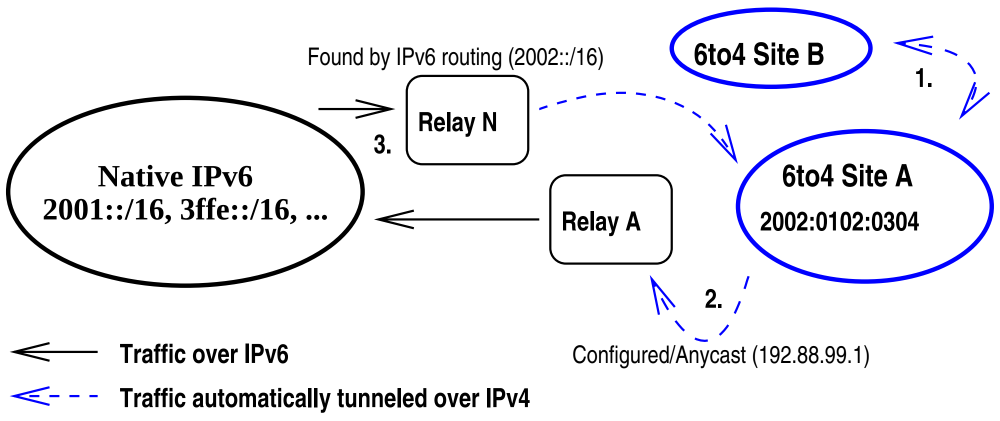
Found by IPv6 routing (2002::/16)
6to4 Site B
1.
Relay N
3.
Native IPv6
2001::/16, 3ffe::/16, ...
Relay A
6to4 Site A
2002:0102:0304
2.
Configured/Anycast (192.88.99.1)
Traffic over IPv6
Traffic automatically tunneled over IPv4

Figure 3: 6to4 modes of operation

There are drawbacks, though; if getting connectivity is easy but the application support is still lacking, it's common that after an initial interest, users tend to forget having been signed up altogether: there is a need for a "garbage collection" process in the system. In the same tune, if a user signing up is using a dynamic IPv4 address, and his address changes at some point, it's quite probable that the old data will remain in the database for long: and the address may be given to some other user who has never even heard of IPv6. Now, if some IPv6 user tries to reach the original user, the packets are tunneled to the new IPv6-unaware user. This could even be seen as a Denial of Service -attack, even if the IPv6 packets were completely legitimate. All in all, rather heavy infrastructure is required from ISP's part to make the Tunnel Broker service robust.

3.2 6to4

6to4 [4] is a relatively new mechanism to completely automate getting IPv6 connectivity. It was defined after problems with "automatic tunneling using compatible addresses", discussed below, became apparent.

For each globally unique IPv4-address there exists a mapping to a 6to4 IPv6 prefix. For example, "11.12.13.14" becomes "2002:0b0c:0d0e::/48". The prefix can be subnetted to up to 65536 networks.

There are three ways traffic can flow when using the 6to4 mechanism (fig. 3):

1. traffic from 6to4 to 6to4
2. traffic from 6to4 to non-6to4
3. traffic from non-6to4 to 6to4

The first will work trivially if the nodes are within the same 6to4 site. Otherwise, direct tunneling between the 6to4 sites is used. As the IPv6 address embeds the IPv4 destination address, packets can easily be tunneled to the destination site's 6to4 border router, which will decapsulate them and process or forward them as appropriate.

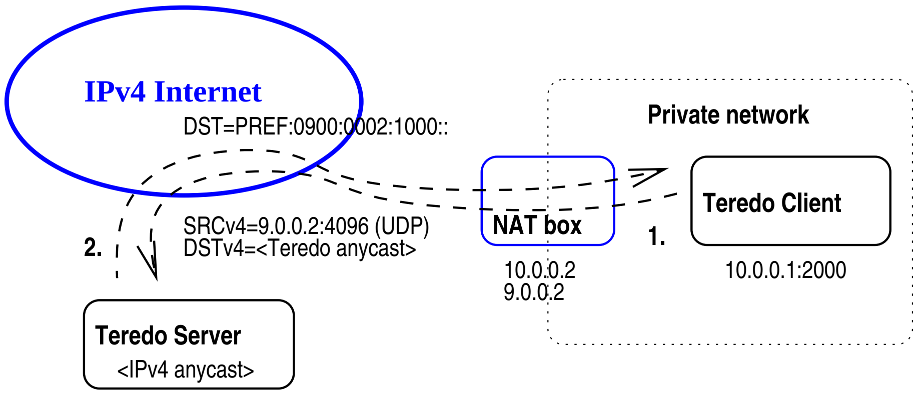

Figure 4: Teredo IPv6 address determination (note: 4096 = 0x1000)

The second and the third rely on the existence of so-called 6to4 relay routers: routers which have both non-6to4 and 6to4 IPv6 addresses, and which are willing to forward packets, bridging these two conceptual clouds.

In the second case, the 6to4 border router has to configure an IPv4 address of a relay router, hopefully rather close to it. This can be a small management problem: the user may not know which is currently the closest one or where to find the information. Luckily enough, IPv4 anycast address “192.88.99.1” [5] always finds the closest one supporting anycast. The finding is done by advertising the same route to “192.88.99.1” from every 6to4 relay supporting anycast; the source only sees and uses the topologically closest route.

In the third case, non-6to4 sites try to reach 6to4 sites also via a relay; this is referred to as the “return path”. Usually the relay is different from the second case unless the source and destination are topologically close to each other. The issue is that these sites can’t configure any 6to4 relays and finding the closest happens via the normal routing process: all relays advertise “2002::/16” route to everywhere they want to serve. As every relay advertises the same route, every relay will transport packets to all 6to4 destinations.

This is problematic because it can take, in the worst case, quite a long (e.g. 100-200ms) to even find a relay router willing to serve third parties. This is one of the greatest drawbacks of 6to4, but can be remedied by adding more relays. Others include no real support for inter-domain multicast (not that it is that common anyway) and problems with reverse DNS delegations under 2002::/16 prefix.

3.3 Teredo aka Shipworm

Teredo, previously known as Shipworm, [14], is a mechanism similar to 6to4. Its focus is to be able to provide connectivity easily in a network where NAT is being used, so that there is no globally unique IPv4 address to generate a 6to4 prefix from. Teredo is intended to be even a more short-term solution than 6to4; just enough to get the ball rolling until native (or even IPv4 NAT-free) connectivity becomes available and viable in a few years.

In Teredo, UDP is used for tunneling instead of protocol 41, and the generated IPv6 prefix will include the IP address of the first (from the direction of Internet) router performing

NAT and UDP port number of the port that will be used for tunneled to the internal nodes. This is done via sending a small packet outside, have it reflected back from a Teredo Server and see which address and port is the way back in (fig. 4). Very similar mechanisms are being defined for e.g. IPSEC and Mobile IPv4 NAT Traversal.

As Teredo is still work-in-progress, it isn't useful to cover the design in detail. Recently, concerns have been raised, including ones from IESG and IAB [21], about address stability and reliability of the mechanism, security of routers acting as relays and interactions with routing and ingress filtering.

3.4 Other mechanisms

Native connection, with all the equipment (e.g. ISP's edge router and customer's router or computer) supporting IPv6 is at least a theoretical possibility. However, this is not applicable in practice yet: remembering the requirements, residential networks usually cannot (or don't want to) afford expensive high-end equipment (and IPv6 support is not usually available to the low-end systems yet, due to low demand), and the providers are naturally hesitant on upgrading their production, revenue-generating infrastructure to IPv6.

6over4 [3] was one of the first defined tunneling solutions, but requires that IPv4 infrastructure is multicast-enabled, as Neigbor Discovery, required for establishing automatic tunnels, is performed using multicast. Site-local or global multicast is a difficult requirement. Therefore 6over4 hasn't really been implemented. In certain restricted scenarios, the mechanism might have been more practical.

Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) [18] is a mechanism being defined that can be used for automatic tunneling (IPv4 address is embedded in the last 32 bits of the IPv6 address) inside a site; thus it is not all that usable in this scope.

So called “BGP tunneling” [15] can be used to set up tunnels between the edge routers over an non-upgraded core similar to how ATM or Multi-protocol Label Switching (MPLS) works. BGP tunneling is also only mentioned for completeness. A similar mechanism is being defined to be used with interior routing protocols OSPF and IS-IS.

Automatic tunneling with compatible addresses [6] (using addresses of the form “::1.2.3.4”) was a predecessor of 6to4; however, it is being deprecated because every node would have to have a global IPv4 address (in 6to4, only site border routers do), and because there are a few security problems with the approach.

Dual-stack Transition Mechanism (DSTM) [17] is a mechanism for enabling casual, temporary IPv4 connectivity in the IPv6-only network. Especially in the context of residential networks, this will not be applicable for some time yet except for some IPv6-only scenarios.

3.5 Applicability of Connectivity Mechanisms

It is assumed that native connectivity is not an option with the usual equipment and budget at this point.

Applicability of the connectivity mechanisms is usually set by whether NAT is being used, whether by the operator or in CPE (Customer Premises Equipment) router. It usually does not matter if the residential user himself performs NAT (e.g. to connect multiple computers with one public IP address), as long as the said system can be upgraded to IPv6.

If there is no NAT in the network, everything is possible. Moreover, if the IPv4 address(es) are static, manual tunneling and tunnel brokering are possible in a practical sense; if the address(es) are dynamic, manual mechanisms should be avoided at all costs. In any case, even though there are some shortcomings in 6to4, it is clearly the winner due to its ease of use in the start-up phase.

However, if there is NAT in the network, there are basically two options depending on whether the operator wants to co-operate or not. If it does, mechanisms like configured tunnels, automatic tunneling, tunnel brokers or ISATAP can be used in the private network to gain global IPv6 connectivity through an IPv4/IPv6 router the operator has set up; actually this can be done by anyone in the internal network who has a public IP address. Of the previous mechanisms, ISATAP would be preferred due to its automation. This may come up in e.g. xDSL networks where routed ATM encapsulation is used. If the operator does not want to co-operate to provide IPv6 service in the internal network, mechanisms like Teredo can be used (assuming they would be finalized by then): punching a hole through NAT requires nothing at all of the operator.

4 IPv4 to IPv6 Translation Mechanisms

There are two approaches to the transition, which may be applicable in different phases of the process: Dual-stack (all nodes implement both IPv4 and IPv6) and IPv6-only combined with translation. The former requires no special IPv6 architecture besides connectivity. The latter, in practice, requires some mechanisms to access IPv4-only services which are bound to be used for quite a while yet.

Here, the major translation mechanisms are very briefly introduced. These could be divided in two: those that work in the network or in the node itself.

4.1 TCP/UDP Relay

TCP/UDP relay [12] works as a transport-level relay between IPv6 and IPv4: IPv6 node connects to a published address (e.g. in DNS) consisting of the network prefix and IPv4 address embedded in the last bits, e.g. “3ffe:ffff::0b0c:0d0e” for “11.12.13.14”. This way, the TCP connection or a UDP stream is established first to a port of an IPv6 address, and all the payload of the protocol is relayed to the IPv4 address in a separate connection from the relay.

TCP/UDP relay can also be used as a full-fledged protocol translator like NAT-PT (see below) with a DNS proxy placed between the hosts and the DNS server. The advantage compared to NAT-PT is that nothing needs to be done on the direct path toward the destination (i.e. the router).

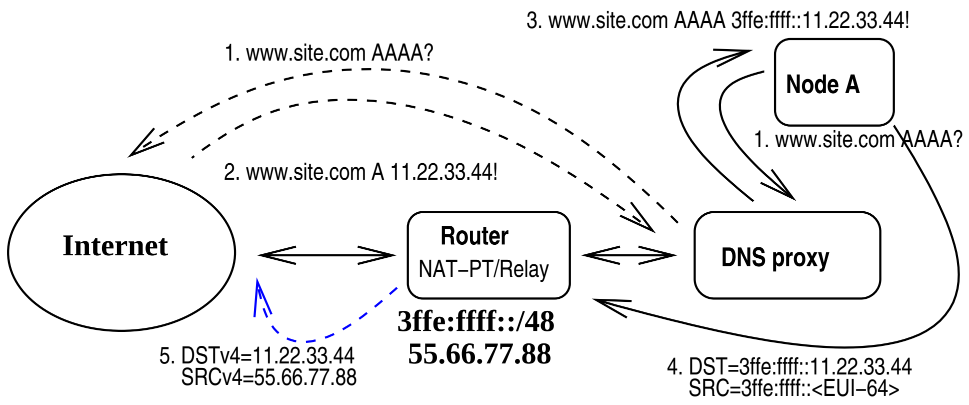

Figure 5: Outbound DNS translation algorithm and translation (DNS proxy case)

A disadvantage, in some scenarios, is that all the IPv6 traffic seems to originate from one IPv4 address of the from the relay; this can be bad from e.g. statistics point-of-view; this is common to NAT-PT and other mechanisms, of course.

Another disadvantage is that usually only the headers are translated; if addresses are transported in the payload (as e.g. some videoconferencing or VoIP applications do), they cannot be translated because the translator cannot know how to perform the operation for every possible protocol. Such a translation would require mechanisms similar to ALG; but if an algorithm is available for IPv4 NAT's, it could be reasonably well adopted to IPv6 too.

4.2 NAT-PT

Network Address Translation - Protocol Translation (NAT-PT) [8] uses Stateless IP/ICMP Translation [7] algorithm in a stateful manner. It uses the same address mapping as TCP/UDP relay, but DNS proxying is done transparently on the wire by inspecting DNS queries with IPv4 'A' records and rewriting the replies to be IPv6 'AAAA' records to point at the NAT-PT translator (see fig. 5 for an example of DNS proxying and translator; with NAT-PT DNS proxying is done on the router). The disadvantage is that all the packets need to go through the same device; in practice this is not a problem if e.g. an xDSL router performs this translation.

There are numerous problems with NAT-PT's DNS proxying mechanism: for example, with DNS queries that return both A and AAAA, or those that are done over native IPv6 transport [20].

4.3 ALG and SOCKS

Application-level Gateways and SOCKS [11] are rather high-level translation mechanisms. SOCKS provides automatic translation for all applications but they must support SOCKS interfaces. ALG is an application-specific translation mechanism, typically used for e.g. HTTP or FTP.

Both require special configuration in all the clients, so the scalability is questionable.

4.4 Bump-in-the-API

Bump-in-the API (BIA) [16] is an in-the-node technique which intercepts incoming and outgoing packets' system calls in the socket API level and provides translation there using a temporary address pool with addressed from the prefix 0.0.0.0/24. As previously seen, DNS proxying at the API level in a key role in the mechanism.

This is an improvement over similar Bump-in-the-Stack (BITS) [9] approach as there is no need for any header translation in the packets.

The mechanism is especially useful for being able to use old IPv4-only, often binary, applications with IPv6.

4.5 Applicability of Translation Mechanisms

All the mechanisms described above are only capable of bidirectional unicast traffic; separate work is being done on multicast translation, for example [22]. However, multicast is only sparsely deployed at the moment, so this is not too huge a practical problem.

NAT-PT has some problems, but in small controlled environments like residential networks, it may very well be just enough.

TCP/UDP relay with DNS proxying has a share of the NAT-PT problems, and it scales better to a bit larger environments too.

SOCKS is usually only an option if there is already experience with SOCKS in the organization; thus it does not fit well into residential networks. ALG, in the form of FTP and HTTP proxies, could help move lots of traffic over to IPv6, but that may not be a goal in itself, as noted below.

BIA is useful for old binary applications; it could be used for others too, those that haven't been ported yet, but that is frowned upon.

5 Migration and Co-existence Scenarios

There are different paths to adopting IPv6; which makes the most sense often depends on which phase of the transition is in progress and how one expects the future progress to go. Three different approaches are discussed.

5.1 IPv6 for requiring applications only

This is the most conservative approach (excluding "we don't need IPv6 at all", of course), but in many scenarios also the most sensible approach: use IPv6 only when there are some benefits in doing so.

This is well-suited to network architectures where there are lots of legacy equipment, but potentially also new IPv6-aware equipment and applications; for completely new infrastructure "IPv6-only" may be a better course.

In practice this would mean obtaining IPv6 connectivity via some mechanism, preferably automated one, and using IPv6 when possible for applications such as peer-to-peer VPN's, mobility, VoIP, videoconferencing, and distributed file sharing and computing. The requirement is that the other end is also able to communicate using IPv6; first, this may be a problem, but if applications gain popularity and obtaining connectivity is easy, this will happen fairly quickly.

In this scenario, converting generic applications like WWW browsers, especially if the IPv6 support cannot be disabled, may be counter-productive, as noted below.

This is the most likely approach for most residential users in Western countries: added benefits without degrading service for the casual use.

5.2 IPv6 on all enabled applications to all IPv6 destinations

A bit more radical approach is to get as many applications ported to support IPv6 as soon as possible, and try to use IPv6 as much as possible – if the destination site supports IPv6, always use it. This is likely to be adopted at least by some technically-oriented people.

This approach may not be the most suitable for every user because of a potential drawback: in this phase, IPv6 Internet architecture is still evolving and the connectivity is often worse than with IPv4. This can be seen in e.g. latency, jitter, bandwidth, and overall reliability. In the long term, especially when dedicated infrastructure is used, IPv6 will improve these, though.

One powerful element here would be enabling IPv6 on e.g. operators' WWW-proxies: that way the amount of IPv6 traffic would increase dramatically as IPv6 gets enabled in WWW and FTP servers. However, as stated above, this should not be a goal as such.

This is likely to happen for technical users, and will get more popular at the expense of the limited approach as more benefits of IPv6 are realized and the IPv6 infrastructure has stabilized.

5.3 IPv6-only

IPv6-only is the most radical approach: do not use IPv4 in the network at all (temporary needs can be met with DSTM or with translation), or even, don't even have IPv4 capability in the nodes (in practice may require at least BIA).

This approach may be, in this phase, applicable to:

- test networks and environments
- whole new infrastructures which have to live long and cannot be depreciated in a few years' time, for example 3G.

- in locations where there is a sore lack of IPv4 addresses, namely the Far East: among others China, South Korea, and Japan.
- in locations where the society is rather self-contained: almost all the content people are interested in can be produced nationally (and there is no such a big need for protocol translation if one assumes services will also be IPv6-enabled); this is the case in e.g. Japan and in Europe to a lesser extent perhaps in e.g. France and Germany/Austria.

This approach requires a rather heavy protocol translation infrastructure; WWW-proxies are essential, but smaller-scale translation will also be necessary for the last 5-10 (very rough estimate) percent of the traffic.

This is the most likely approach at least in the environments of the last two cases above; indeed, trials of IPv6-only home infrastructure have already been running for a long time now (e.g. by NTT in the Far East), and soon they will move to production.

6 Future Work

Teredo [14], despite being still work-in-progress, needs more study. In particular, especially how different kinds of NAT's interact with NAT piercing techniques should be properly analyzed. To avoid having to use mechanisms such as Teredo, papers exploring the options for operators and organizations (similar to this) should be published. The benefits and the deployment of VoIP, videoconferencing, file sharing and such over IPv6 should be measured.

7 Conclusions

In this paper, several reasons for introducing IPv6 in residential networks, applicable connectivity and translation mechanisms, and a few transition and co-existence scenarios were discussed.

It is clear that IPv6 does provide new services to the users, as well as powerful capabilities of introducing new ones: provided by end-to-end transparency, autoconfiguration, global addressing and more.

There a few different approaches IPv6 could be taken to use in residential networks: using it for new services only, using it for everything that supports IPv6 and using IPv6 exclusively. The best approach depends heavily on the circumstances, but for most Western countries the first would be the most cautious and possibly the most effective approach.

References

[1] Deering, S., Hinden, R. Internet Protocol, Version 6 (IPv6) Specification. IETF Draft Standard RFC2460, December 1998.

[2] Conta, A., Deering, S. Generic Packet Tunneling in IPv6 Specification. IETF Proposed Standard RFC2473, December 1998.

[3] Carpenter, B., Jung, R. Transmission of IPv6 Packets over IPv4 Domains without Explicit Tunnels. IETF Proposed Standard RFC2529, March 1999.

[4] Carpenter, B., Moore, K. Connection of IPv6 Domains via IPv4 Clouds without Explicit Tunnels. IETF Proposed Standard RFC3056, February 2001.

[5] Huitema, C. An anycast prefix for 6to4 relay routers. IETF Proposed Standard RFC3068, June 2001.

[6] Gilligan, R., Nordmark, E. Transition Mechanisms for IPv6 Hosts and Routers. IETF Proposed Standard RFC2893, August 2000.

[7] Nordmark, E., Stateless IP/ICMP Translator (SIIT). IETF Proposed Standard RFC2765, February 2000.

[8] Tsirtsis, G., Srisuresh, P. Network Address Translation - Protocol Translation. IETF Proposed Standard RFC2766, February 2000.

[9] Tsuchiya, K. et al., Dual Stack Hosts using the "Bump-In-the-Stack" Technique. IETF Informational RFC2767, February 2000.

[10] Durand, A. et al. IPv6 Tunnel Broker. IETF Informational RFC3053, January 2001.

[11] Kitamura, H. A SOCKS-based IPv6/IPv4 Gateway Mechanism. IETF Informational RFC3089, April 2001.

[12] Hagino, J., Yamamoto, K. An IPv6-to-IPv4 Transport Relay Translator. IETF Informational RFC3142, April 2001.

[13] Biemont, W. et al. An overview of the introduction of IPv6 in the Internet. http://www.ietf.org/internet-drafts/draft-ietf-ngtrans-introduction-to-ipv6-transition-07.txt, work-in-progress, July 2001.

[14] Huitema, C. Teredo: Tunneling IPv6 over UDP through NATs. http://www.ietf.org/internet-drafts/draft-ietf-ngtrans-shipworm-05.txt, work-in-progress, February 2002.

[15] De Clerq, D. et al. Connecting IPv6 Islands across IPv4 Clouds with BGP. http://www.ietf.org/internet-drafts/draft-ietf-ngtrans-bgp-tunnel-04.txt, work-in-progress, January 2002.

[16] Lee, S. et al. Dual Stack Hosts using "Bump-in-the-API". http://www.ietf.org/internet-drafts/draft-ietf-ngtrans-bia-03.txt, work-in-progress, February 2002.

[17] Bound, J., at al. Dual Stack Transition Mechanism. http://www.ietf.org/internet-drafts/draft-ietf-ngtrans-dstm-05.txt, work-in-progress, November 2001.

[18] Templin, F., et al. Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). http://www.ietf.org/internet-drafts/draft-ietf-ngtrans-isatap-03.txt, work-in-progress, January 2002.

[19] Okabe N., et al. Minimum Requirement of IPv6 for Low Cost Network Appliance. http://www.ietf.org/internet-drafts/draft-okabe-ipv6-lcna-minreq-01.txt, work-in-progress, February 2002.

[20] Durand, A., Issues with NAT-PT DNS ALG in RFC2766. http://www.ietf.org/internet-drafts/draft-durand-natpt-dns-alg-issues-00.txt, work-in-progress, February 2002.

[21] Internet Architecture Board,, Daigle, L. (ed.). IAB Considerations for UNilateral Self-Address Fixing http://www.ietf.org/internet-drafts/draft-iab-unsaf-considerations-01.txt, work-in-progress, February 2002.

[22] Tsuchiya, K. et al., An IPv6/IPv4 Multicast Translator based on IGMP/MLD Proxying. http://www.ietf.org/internet-drafts/draft-ietf-ngtrans-mtp-01.txt, work-in-progress, February 2002.