Deployment of Services on IPv6

Best Practice Document

Produced by the RENATER-led working group on IPv6 Services

Authors: J. Benoit (Université de Strasbourg/GIP RENATER), S. Muyal (GIP RENATER), C. Palanché (Université de Strasbourg/GIP RENATER), P. Wender (INSA Rouen/GIP RENATER)

November 2013

Document No: GN3plus-NA3-T2-R3.2
Version / date: V1.0, 27 November 2013
Original language : French
Original title: “Déployer des services en IPv6”
Original version / date: V1.0, 27 November 2013
Contact: cbp@listes.renater.fr

RENATER bears responsibility for the content of this document. The work has been carried out by a RENATER-led working group on IPv6 as part of a joint-venture project within the HE sector in France.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 605243, relating to the project 'Multi-Gigabit European Research and Education Network and Associated Services (GN3plus)'.

Table of Contents

Table of Figures

Executive Summary

There are already numerous best practice documents about the deployment of IPv6 on both client workstations and a network infrastructure, however, the deployment of IPv6 on servers and services has rarely been covered. This document is aimed at overcoming this deficiency. The scope of best practice discussed here includes network services (web, mail, DNS) and the servers on which these services run.

This document is aimed at system and network administrators. Topics such as workstation configuration strategy (with SLAAC, DHCPv6, etc.) or configuration of interconnections between routers on IPv6 is not covered. IPv6 connectivity must already be available on the core network.

More precisely, these recommendations apply to the deployment of services on IPv6 on Unix servers for a service that already exists on IPv4. The dual-stack protocol approach (IPv4 – IPv6) on the server is preferred.

Initially, some general advice is given about the deployment of services on IPv6: preparation, training, organisation, tests and implementation. Different cases (messaging, web and load sharing) are then studied in detail:

Finally, advice on security is given.

1 General recommendations

The approach to the implementation of services on IPv6 can be divided into several stages:

- Definition of services to be deployed on IPv6, with impact analysis.
- Training of system administrators.
- Setting up IPv6 connectivity on the server network.
- Pre-deployment monitoring of the service on IPv6.
- Tests to ensure the service is functioning correctly.
- Activation of IPv6.
- Publication of the service address on DNS.

1.1 Stage 1: Organisation of deployment

This stage allows the deployment to be prepared by precisely defining its scope. It requires answers to the following questions:

- Which service will be deployed on IPv6?
- Which servers is it running on?
- Which other services does it depend on? (example of dependence: a web service depends on a database or a directory to operate)
- Which other services depend on this service? (e.g.: a web service used by other applications).

1.1.1 Approaches

Deployment of IPv6 is implemented for a service that already exists on IPv4. There are three possible approaches:

- Service on IPv6 running on a dedicated server,
- A proxy application server playing the role of translator: the client connects to it on IPv6 and the proxy will relay the traffic on IPv4 to the service which remains unchanged,
- The service running on the same server, which has a dual-stack network, IPv4 and IPv6.

The first approach is not recommended as it complicates management of the service: by doubling the servers, maintenance is increased.

The second approach is possible if the service cannot run natively on IPv6. It allows access to the service on IPv6 without modifying it, but transfers some of the complexity to the proxy (session management).

The third approach is the most common in the context of a campus network. This is the approach mainly studied in this document.

1.1.2 Deployment order

When IPv6 is activated on a server, this impacts all services hosted on this server.

It is best to organise the deployment progressively, server by server, taking dependencies into account. The servers with the fewest dependencies and services should be picked first. The dependencies are not blocking: the activation of IPv6 on a server for access to the services it hosts does not prevent the use of IPv4 to its dependencies. For example, if a database server doesn't work on IPv6, a web server deployed on IPv6 can connect to the database on IPv4.

1.1.3 Priority for web applications

It is strongly recommended to start with web services:

- The majority of applications rely on web servers that support IPv6 (Apache [1], Lighttpd [2], Nginx [3], etc.),
- The majority of web applications do not manipulate IP addresses: the front-end web server is most often responsible for IPv6 management.

1.1.4 Applications (web or non-web) manipulating IP addresses

For an application using IP addresses, its compatibility with IPv6 must be checked. There are test suites for validating its correct operation, for example IPv6CARE [4]. In addition, many languages and their environments (such as C, PHP, Java, Perl, Ruby, Python, etc.) offer a good level of IPv6 support.

1.2 Stage 2: training of system administrators on IPv6

This first stage is indispensable for those responsible for server administration and for services to run services on IPv6.

It is recommended to start training several months before deploying IPv6. The training must be essentially practical and the training plan must include at least the following points:

- Addressing and routing (local and global address scope, etc.), with, in particular, an example of an addressing plan for server networks.
- Differences between IPv4 and IPv6.
- Description of the protocols associated with IPv6 (NDP, ICMP6, etc.).
- Security.
- Configuration of servers on a test network (devoting sufficient time to hands-on training in a lab with real or virtual equipment).
- Deployment strategy (in different stages).

1.3 Stage 3: activating IPv6 on the server network

It is assumed that IPv6 is already active on the core network and that external connectivity is already established and that IPv6 is not delivered on a separate VLAN. IPv6 connectivity must be set up on the server production network.

Important! Activation of IPv6 on the server network must be monitored:

- It is not necessary to activate router-advertisements on the server network: there is no need for either prefix or router advertisements.
- De-activation of auto-configuration of IPv6 addresses on the servers is recommended.
- The gateway is also statically configured by default.
- It is recommended that the IPv6 filtering policy on the network periphery be checked for consistency with the IPv4 policy. For example, if SSH access is blocked for all servers on IPv4, it must also be blocked on IPv6.

Reminder: all servers on which IPv6 activation is desired must have an IPv4 and IPv6 dual-stack.

1.3.1 Server network addressing plan

In setting up a server addressing plan, attention must be paid to the following points:

- Servers must only have static IPv6 addresses.
- The numbering method must remain simple. For example, make the last byte of the IPv6 address the same as the IPv4 address: if the server IPv4 address is 192.0.2.54 on the 192.0.2.0/24 network, then the IPv6 address will be: 2001:db8::54 on the 2001:db8::/64 network. When the last byte is insufficiently specific (for example for a prefix on /23 or smaller), the last two or three bytes of the IPv4 addresses can be transposed to the IPv6 address. For example, 192.0.2.1/23 becomes 2001 :db8 ::2 :1/64.

1.4 Stage 4: monitoring the IPv6 address

The operation of the service on IPv6 must be visible before the declaration in the DNS. It is necessary to both test the IPv6 connectivity and supervise the service on IPv6. It is appropriate to bring all the tests related to IPv6 services together into a dashboard dedicated to IPv6 monitoring.

For example, in Nagios [5], it is recommended to add an additional “host” for IPv6 (mywebserver-v6) to make the state of the server explicit.

Important: a check must be made for compatibility with IPv6 (for example, Nagios plugins do not all support IPv6).

If the test supports IPv4 and IPv6, it is necessary to verify that the proper functioning status reported by the test explicitly gives the state of service on IPv6.

For example, for some plugins, it is possible to force the test on IPv6:

```
check_http -H www.example.com -6
```

1.5 Stage 5: tests and validation

As far as possible, IPv6 activation must be validated on a pre-production platform:

This platform must be a copy of the target platform in conditions as close as possible to those of the production network. If the platform consists of several servers (frontal load-balancing, back-end servers and database servers), the entire chain must be tested.

1.6 Stage 6: activation

Activate the static IPv6 address on the production server. Verify via monitoring that the service is operational on IPv6.

1.7 Stage 7: publication of the service on DNS

Note: it is not necessary to declare the IPv6 address of a server before ascertaining that the services run on IPV6.

Once monitoring indicates that the service is operational, the IPv6 address of the server can now be declared (AAAA record) and its reverse record (PTR) on DNS.

The operation of the service must now be tested by name. The most thorough test (i.e. guaranteeing that IPv6 is used) is done with a tool like telnet or netcat [6] by specifying the option “-6”. For example:

```
nc -v -6 monserveur.exemple.com 80
```

Once the address is published on DNS, it is possible to modify the server declaration via monitoring to perform an IPv4 and IPv6 test, if the plugin allows this.

2 Implementation of the IPv4/IPv6 dual-stack

Taking the web service as an example, two methods are possible:

- The web server listens on IPv4 and IPv6 sockets at the same time.
- The web server only listens on an IPv6 socket and IPv6 addresses of the IPv4-mapped (::FFFF:ADDR_IPv4) type are used.

First case – IPv4 and IPv6 sockets: The web server listens on an IPv4 socket as well as on an IPv6 socket on the configured ports (by default, ports 80 and 443).

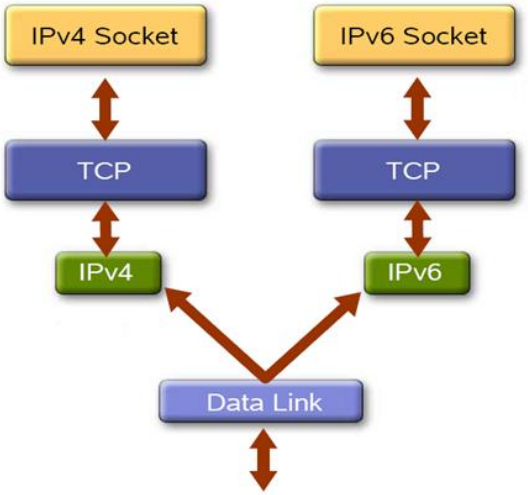
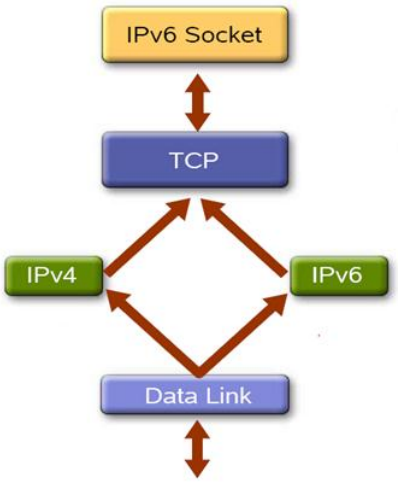


Figure 2.1: IPv4 and IPv6 sockets

Second case – Socket IPv6: the other possible architecture is that a single TCP socket is open with use of IPv6 addresses of the "IPv4-mapped"-type. The web service listens to V4 and v6 requests only via the IPv6 socket (see RFC 3493, "Basic Socket Interface Extensions for IPv6").

For example: [::ffff:192.0.2.128] for the address IPV4 [192.0.2.128].



Source: Rino Nucara, GARR, EuChinaGRID IPv6 Tutorial [7]

Figure 2.2: Single IPv6 socket (IPv4 “mapped” to IPv6)

In the majority of Linux distributions, the second case is used. A single socket functions on IPv4 and IPv6 at the same time. Note that this will not work is if the address associated with the socket is not specified. This socket is represented as an IPv6 socket in the result of the netstat command:

```
tcp6       0      0 :::80           :::*            LISTEN      24035/nginx
```

3 Mail

3.1 Configuration of main MTAs on IPv6

3.1.1 Sendmail

In the mc configuration system, define the address family using the directive DAEMON_OPTIONS (corresponding to the option DaemonPortOptions in sendmail.cf [8]):

```
DAEMON_OPTIONS(`Family=inet6, Name=MTA, Modifier=O')dnl
```

The “O” modifier allow sendmail to run even if IPv6 is still not active.

Note: under Linux, by default, a single socket is created for both IPv4 and IPv6.

This option is sufficient and cannot co-exist with the option “Family=inet” for IPv4.

Under *BSD, both options must be set.

The following command must be used to verify that sendmail is indeed listening on IPv6:

```
lsof -i 6TCP:25 -n
```

The connection on the SMTP port must then be tested, using the following command:

```
openssl s_client -starttls smtp -connect @IPv6_server:587
```

3.1.2 Postfix

IPv6 is activated by default in Postfix, starting from version 2.9. Note that for earlier versions, the following must be specified:

```
inet_protocols = all
```

in the file main.conf **[9]**.

3.2 Impact of IPv6 activation

Messaging relay mechanisms are the same on IPv4 and IPv6. There is therefore no specific architecture to be implemented. However, the activation of IPv6 has an impact on the routing of messages to other sites. Be aware that many messaging servers (MTA) are currently badly configured:

- They lack external IPv6 connectivity.
- They only have one local IPv6 address.
- Even so, they perform a DNS resolution by requesting the server IPv6 address from the client messaging server.

The external server is consequently unable to send mail.

The messaging administrators of the sites concerned must be warned, indicating that they should activate IPv6 on their site or de-activate IPv6 on the server or the messaging service.

This problem is often linked to the default configuration of the variable “inet_protocols” of the postfix MTA. The default configuration contains:

```
inet_protocols = all
```

The administrator must be told to force this value to “ipv4” if they are unable to bring about correct functioning of IPv6 immediately.

4 Web

4.1 Architecture

4.1.1 Architecture without redundancy

In a classic web server architecture, a few configuration changes are generally sufficient to “v6fy” the server.

4.1.2 Proxy and load-balancer

In order to make the web server architecture more robust, proxies and load-balancing mechanisms are often used. In the majority of cases, it is sufficient that these mechanisms support IPv6 in order to offer a web service that is accessible on IPv6. The ‘backend’ part does not necessarily need to be “v6fied”.

There are several ways of implementing load-balancing:

- **at the network level:** VRRP or CARP mechanisms can send packets to a set of web servers. These mechanisms do not take the state of the web service into account. If the network interface is active, the mechanism will continue to send packets to the server in question. A probe to verify the state of the service must be set up.
- **at the DNS level:** Round-robin DNS mechanisms can be used. However, these mechanisms do not take the status of the web service into account.
- **at the application level:** The application load-sharing mechanisms have the advantage of taking the status of the web service into account.

4.2 Examples

4.2.1 Architecture without redundancy

4.2.1.1 *Apache*

Apache configuration for host:

If the IPv6 stack is already configured on the server, the apache server can respond to IPv6 requests by default, by listening to all of the server's IP addresses. The configuration is in the ports.conf file for a Linux Debian distribution:

```
Listen 80
<IfModule mod_ssl.c>
Listen 443
</IfModule>
```

If you wish to listen to specific IPv4 and IPv6 addresses, the syntax is as follows:

```
Listen 192.168.0.1:80
Listen [2001:db8::1]:80
```

To verify that the server is listening on ports 80 on IPv4 and IPv6, under Linux the following commands verify the list of open ports on IPv4 and IPv6 on TCP port 80:

```
netstat -ant
lsof -ni tcp:80
```

Apache configuration for SSL :

```
For the apache server to listen to the default SSL web port (443), the ports.conf
file must contain the following lines:
<IfModule mod_ssl.c>
Listen 443
</IfModule>
```

Then the different `virtual hosts` need to be associated in the configuration.

```
<VirtualHost *:80>
        ServerName web_server_name
        DocumentRoot /directory/
</VirtualHost>
```

A second example with a virtual host listening on the IPv4 and IPv6 addresses 192.168.0.1 and 2001:db8::1:

```
<VirtualHost 192.168.0.1:80 [2001:db8::1]:80>
        ServerName web_server_name
        DocumentRoot /directory/
</VirtualHost>
```

4.2.1.2 *Nginx*

You only need to add the following directive to nginx configuration in the server section. (**Note**: in systems that create only one socket for IPv4 and IPv6, e.g. Linux, the following configuration will also work for IPv4):

```
server {
listen  [::]:80
…
}
server {
    listen [::]:443
…
}
```

4.2.2 Proxy and IPv6/IPv4 translation

4.2.2.1 *Apache*

The Apache module “mod_proxy” [10] allows a reverse-proxy to be set up.

This reverse-proxy listens on an IPv6 address, re-sending the requests to the internal site on IPv4 only.

For example, the site “internal.appliv4.example.com” does not support IPv6. The name “www.example.com” is declared on the DNS associated with the address IPv6 “2001:db8::1”

The Apache configuration is as follows:

```
<VirtualHost [2001:db8::1]:80>
  ServerName www.example.com
  ProxyRequests Off
  ProxyPass / http://internal.appliv4.example.com/
  ProxyPassReverse / http://internal.appliv4.example.com/
  ProxyPreserveHost On
</VirtualHost>
```

4.2.2.2 *Nginx*

Nginx allows a reverse-proxy (http/https) to be set up to listen on an IPv6 address, re-sending the requests to the servers on IPv6 or IPv4. To activate this service, it is necessary to:

Define the list of internal servers using the upstream directive:

```
upstream my_service {
    server internal.appliv4.example.com
}
```

- Activate redirection to the internal servers with the proxy_pass directive:

```
server {
        listen [::]:80 ;
        server_name www.example.com;
        location / {
                proxy_pass http://my_service;
        }
}
```

4.2.2.3 *Translation with OpenBSD*

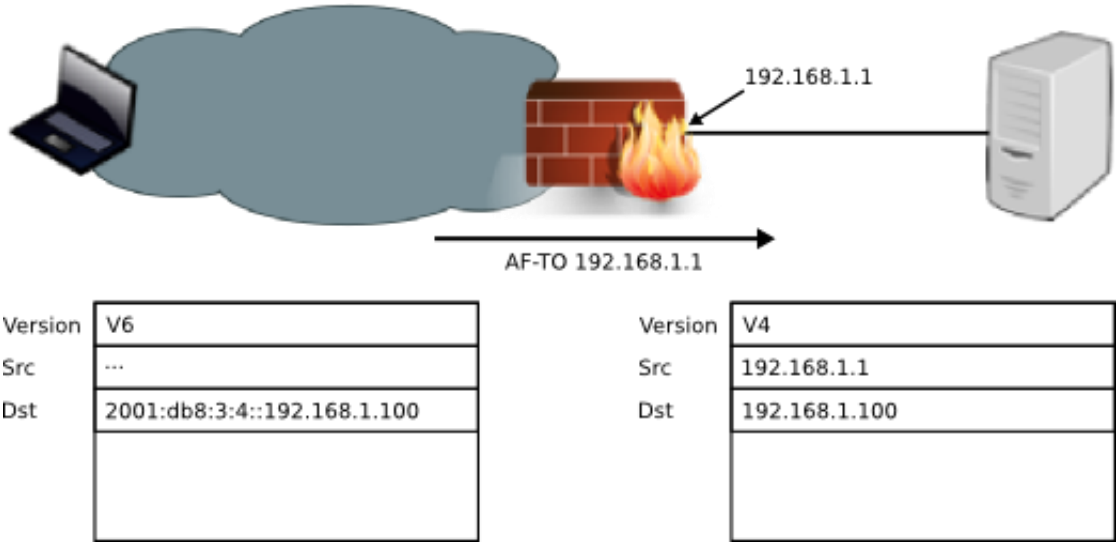


Figure 4.1: IPv6/IPv4 translation with OpenBSD

On OpenBSD **[11]** PF **[12]** allows family address **translation**, in particular from IPv6 to IPv4:

- The server IPv4 address is incorporated at the end of the IPv6 address: 2001:db8:3:4::c0a8:0101 for 192.168.1.1.
- The server IPv6 address must be declared on an external interface.
- Under the translation rule, it is necessary to specify the IPv4 source address to which the IPv6 address will be rewritten (i.e. the address declared on the internal interface of the firewall).

Example of the rule to be added in /etc/pf.conf:

```
pass in on bnx0 inet6 af-to inet from 192.168.1.1
```

4.2.3 Redundancy

On Linux, the Keepalived tool implements the VRRP protocol for IPv6 and is based on IPVS **[13]**, which also supports IPv6.

On OpenBSD, CARP, PF, PFSYNC and Relayd are compatible with IPv6. A redundant load-balancing architecture can be built natively.

5 Security

In principle, the same security policy must be applied to the services on IPv4 and on IPv6. There must be a packet-filtering mechanism on IPv6 before activating IPv6 on the servers. The principle of authorising an IP source address on IPv4 no longer works on IPv6. It is necessary to authorise entire networks and rely on application security (authentication, authorisation).

5.1 Filtering

The filtering policy applied to the server (and not to an external filtering device) is divided into three parts:

- A default policy that prohibits all traffic.
- Rules that allow the functioning of IPv6.
- Specific rules for the functioning of the service.

5.1.1 Default policy

All traffic is prohibited by default

Example for Netfilter/Linux **[14]**:

```
ip6tables -P INPUT DROP
```

Example with PF/*BSD:

```
block inet6 all
```

5.1.2 Rules necessary for IPv6 to work

To guarantee that IPv6 works on the protocol level, when setting up a filtering policy, attention must be paid to the following points in particular:

- ICMPv6.

- Multicast.
- Certain extensions.

5.2 ICMPv6

Internet Control Message Protocol for IPv6 (ICMPv6) is mandatory to the proper operation of IPv6. This protocol must never be blocked completely, otherwise connectivity will not work. For example, the Neighbor Discovery protocol is based on ICMPv6. Similarly, the ICMP message “Packet Too Big” is mandatory to the proper operation of the MTU’s discovery algorithm (Path MTU discovery). A minimum subset of ICMPv6 documented in RFC 4890 should therefore be authorised [15]. ICMPv6 packets of the “router advertisement” type are voluntarily blocked in the context of a server network where it is recommended that RA be deactivated on the routers.

Example with Netfilter/Linux:

```
ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-request -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-reply -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type address-unreachable -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type port-unreachable -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type time-exceeded -j ACCEPT

ip6tables -A INPUT -p icmpv6 --icmpv6-type parameter-problem -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-advertisement -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-solicitation -j ACCEPT

# MLD Query
ip6tables -A INPUT -p icmpv6 --icmpv6-type 130 -j ACCEPT

ip6tables -A INPUT -p icmpv6 -j DROP
```

Example with PF/*BSD:

```
pass in inet6 proto icmp6 all icmp6-type echoreq
pass in inet6 proto icmp6 all icmp6-type echorep
pass in inet6 proto icmp6 all icmp6-type toobig
pass in inet6 proto icmp6 all icmp6-type timex
pass in inet6 proto icmp6 all icmp6-type paramprob

pass in inet6 proto icmp6 all icmp6-type unreach code addr-unr
pass in inet6 proto icmp6 all icmp6-type unreach code port-unr

pass in inet6 proto icmp6 all icmp6-type neighbrsol
pass in inet6 proto icmp6 all icmp6-type neighbradv

# MLD query
pass in inet6 proto icmp6 all icmp6-type listqry
```

5.3 Multicast

The local multicast should not be filtered, as it is required for the Neighbor discovery protocol: it is based on the solicited-node multicast address. The entire ff02::/16 prefix must be authorised as an IP destination address.

Example with Netfilter/Linux:

```
ip6tables -A INPUT --destination ff02::/16 -j ACCEPT
```

Example with PF/*BSD:

```
pass in inet6 from ff02::/16
```

5.4 Extensions

In an IPv6 packet, the “Next-Header” header field allows extensions to be inserted between the IPv6 header and the protocol transporting the payload (TCP, UDP, ICMPv6 etc.).

These extensions are chainable.

It is difficult to set up a general policy for the filtering of extensions.

Some extensions are mandatory, e.g. the Hop by Hop extension, used by MLD. MLD is based on ICMPv6 and is necessary for Multicast to work. However, this extension is only useful to routers.

Other extensions can be attack vectors if they have not been correctly configured on the host. For example, fragmentation and routing header extension [16] have been used in the past to launch attacks on servers following vulnerabilities in the implementation of the IPv6 protocol stack. It is possible to block fragmentation completely (although without blocking ICMP signalling).

5.5 Service filtering

The filtering is identical to that in IPv4.

Linux/Netfilter example:

```
# web
ip6tables -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
# dns
ip6tables -A INPUT -p tcp --dport 53 -j ACCEPT
ip6tables -A INPUT -p udp --dport 53 -j ACCEPT
# ipsec
ip6tables -A INPUT -m ah -j ACCEPT
ip6tables -A INPUT -p esp -j ACCEPT
ip6tables -A INPUT -p udp --dport 500 -j ACCEPT
```

pf/*BSD example:

```
# web
pass inet6 proto tcp from any to any port { 80 443 }
# dns
pass inet6 proto { udp tcp } from any to any port 53
# VPN IPSEC
pass inet6 proto {ah esp} from any to any
pass inet6 proto udp from any to any port 500
```

References

[1]	http://httpd.apache.org/
[2]	http://www.lighttpd.net/
[3]	http://nginx.org
[4]	http://sourceforge.net/projects/ipv6-care/
[5]	http://www.nagios.org/
[6]	http://nc110.sourceforge.net/
[7]	http://www.euchinagrid.org/IPv6/IPv6_presentation/Introduction_to_IPv6_programming.pdf
[8]	http://www.sendmail.org/~ca/email/doc8.12/op.html
[9]	http://www.postfix.org/postconf.5.html
[10]	http://httpd.apache.org/docs/2.2/mod/mod_proxy.html
[11]	http://openbsd.org/
[12]	http://www.openbsd.org/faq/pf/
[13]	http://www.linuxvirtualserver.org/
[14]	http://www.netfilter.org/
[15]	http://www.ietf.org/rfc/rfc4890.txt
[16]	http://www.ietf.org/rfc/rfc5095.txt

Glossary

DNSv6	Domain Name System extensions to support IPv6 (RFC3596)
ICMPv6	Internet Control Message Protocol for IPv6 (RFC4884)
NA	Neighbor Advertisement
ND	Neighbor Discovery for IPv6 (RFC4861)
NS	Neighbor Solicitation
PMDv6	Path MTU Discovery for IPv6 (RFC1981)
RA	Router Advertisement
RS	Router Solicitation
SLAAC	Stateless Address Autoconfiguration (RFC4862)

Complete BPDs available at http://services.geant.net/cbp/Pages/Home.aspx

campus-bp-announcements@terena.org