Paket IPV6 - Erweiterung um IPv6-Funktionalität Version 3.10.4

Christoph Schulz
E-Mail: fli4l@kristov.de

Das fli4l-Team
E-Mail: team@fli4l.de

25. Oktober 2015

Inhaltsverzeichnis

1. Dokumentation des Paketes IPV6

1.1. IPv6 - Internet Protokoll Version 6

1.1.1. Einleitung

Dieses Paket ermöglicht es, den fli4l-Router in vielerlei Hinsicht IPv6-tauglich zu machen. Dazu gehören Angaben über die IPv6-Adresse des Routers, die verwalteten IPv6-(Sub-)Netze, vordefinierte IPv6-Routen und Firewall-Regeln bzgl. IPv6-Paketen. Des Weiteren lassen sich IPv6-Dienste wie DHCPv6 konfigurieren. Schließlich ist es möglich, Tunnel zu IPv6-Anbietern automatisch aufbauen zu lassen. Dies funktioniert momentan aber leider nur mit so genannten 6in4-Tunneln, wie sie etwa der Anbieter "SixXS" unterstützt. Andere Technologien (AYIYA, 6to4, Teredo) werden zur Zeit nicht unterstützt.

IPv6 ist der Nachfolger des Internet-Protokolls IPv4. Hauptsächlich wurde es entwickelt, um die relativ kleine Menge von eindeutigen Internet-Adressen zu vergrößern: Während IPv4 ungefähr 2^{32} Adressen unterstützt,[1] sind es bei IPv6 bereits 2^{128} Adressen. Dadurch kann mit IPv6 jedem kommunizierenden Host eine eindeutige Adresse zugeordnet werden, und man ist nicht mehr auf Techniken wie NAT, PAT, Masquerading etc. angewiesen.

Neben diesem Aspekt spielten bei der Entwicklung des IPv6-Protokolls auch Themen wie Selbstkonfiguration und Sicherheit eine Rolle. Dies wird in späteren Abschnitten aufgegriffen.

Das größte Problem bei IPv6 ist dessen Verbreitung: Momentan wird IPv6 – verglichen mit IPv4 – nur sehr spärlich verwendet. Das liegt daran, dass IPv6 und IPv4 technisch nicht miteinander kompatibel sind und somit alle Software- und Hardware-Komponenten, die an der Paket-Weiterleitung im Internet beteiligt sind, für IPv6 nachgerüstet werden müssen. Auch bestimmte Dienste wie DNS (Domain Name System) müssen für IPv6 entsprechend aufgebohrt werden.

Hier tut sich also ein Teufelskreis auf: Die geringe Verbreitung von IPv6 bei Dienstanbietern im Internet führt zu Desinteresse seitens der Router-Hersteller, ihre Geräte mit IPv6-Funktionalität auszustatten, was wiederum dazu führt, dass Dienstanbieter die Umstellung auf IPv6 scheuen, weil sie fürchten, dass sich der Aufwand nicht lohnt. Erst langsam wendet sich das Blatt zugunsten von IPv6, nicht zuletzt unter dem immer stärkeren Druck der knappen Adressvorräte.[2]

1.1.2. Adressformat

Eine IPv6-Adresse besteht aus acht Zwei-Byte-Werten, die hexadezimal notiert werden:

Beispiel 1: `2001:db8:900:551:0:0:0:2`

Beispiel 2: `0:0:0:0:0:0:0:1` (IPv6-Loopback-Adresse)

Um die Adressen etwas übersichtlicher zu gestalten, werden aufeinander folgende Nullen zusammengelegt, indem sie entfernt werden und lediglich zwei unmittelbar aufeinander folgende

[1] nur ungefähr, weil einige Adressen speziellen Zwecken dienen, etwa für Broad- und Multicasting

[2] Inzwischen sind die letzten IPv4-Adressblöcke von der IANA vergeben worden.

Doppelpunkte verbleiben. Die obigen Adressen können also auch so geschrieben werden:

Beispiel 1 (kompakt): `2001:db8:900:551::2`

Beispiel 2 (kompakt): `::1`

Eine solche Kürzung ist aber nur höchstens einmal erlaubt, um Mehrdeutigkeiten zu vermeiden. Die Adresse `2001:0:0:1:2:0:0:3` kann also entweder zu `2001::1:2:0:0:3` oder zu `2001:0:0:1:2::3` verkürzt werden, nicht aber zu `2001::1:2::3`, da jetzt unklar wäre, wie die vier Nullen jeweils auf die zusammengezogenen Bereiche verteilt werden sollen.

Eine weitere Mehrdeutigkeit existiert, wenn eine IPv6-Adresse mit einem Port (TCP oder UDP) kombiniert werden soll: In diesem Fall darf man den Port nicht unmittelbar mit Doppelpunkt und Wert anschließen, weil der Doppelpunkt bereits innerhalb der Adresse verwendet wird und somit in manchen Fällen unklar wäre, ob die Port-Angabe nicht vielleicht doch eine Adress-Komponente darstellt. Deshalb muss in solchen Fällen die IPv6-Adresse in eckigen Klammern angegeben werden. Dies ist auch die Syntax, wie sie in URLs gefordert wird (etwa wenn im Web-Browser eine numerische IPv6-Adresse verwendet werden soll).

Beispiel 3: `[2001:db8:900:551::2]:1234`

Ohne die Verwendung von Klammern entsteht die Adresse `2001:db8:900:551::2:1234`, die unverkürzt der Adresse `2001:db8:900:551:0:0:2:1234` entspricht und somit keine Port-Angabe besitzt.

1.1.3. Konfiguration

Allgemeine Einstellungen

Die allgemeinen Einstellungen beinhalten zum einen die Aktivierung der IPv6-Unterstützung und zum anderen die optionale Vergabe einer IPv6-Adresse an den Router.

OPT_IPV6 Aktiviert die IPv6 Unterstützung.

Standard-Einstellung: `OPT_IPV6='no'`

HOSTNAME_IP6 (optional) Diese Variable stellt explizit die IPv6-Adresse des Routers ein. Falls die Variable nicht gesetzt wird, wird die IPv6-Adresse auf die Adresse des ersten konfigurierten IPv6-Subnetzes gesetzt (`IPV6_NET_x`, siehe unten).

Beispiel: `HOSTNAME_IP6='IPV6_NET_1_IPADDR'`

Subnetz-Konfiguration

In diesem Abschnitt wird die Konfiguration von einem oder mehreren IPv6-Subnetzen vorgestellt. Ein IPv6-Subnetz ist ein IPv6-Adressraum, der über ein so genanntes Präfix spezifiziert wird und an eine bestimmte Netzwerk-Schnittstelle gebunden ist. Weitere Einstellungen betreffen das Veröffentlichen des Präfixes und des DNS-Dienstes innerhalb des Subnetzes sowie einen optionalen Router-Namen innerhalb dieses Subnetzes.

IPV6_NET_N Diese Variable enthält die Anzahl der verwendeten IPv6-Subnetze. Mindestens ein IPv6-Subnetz sollte definiert werden, um IPv6 im lokalen Netz nutzen zu können.

Standard-Einstellung: `IPV6_NET_N='0'`

IPV6_NET_x Diese Variable enthält für ein bestimmtes IPv6-Subnetz die IPv6-Adresse des Routers sowie die Größe der Netzmaske in CIDR-Notation. Soll das Subnetz öffentlich geroutet werden, so stammt es in der Regel vom Internet- bzw. Tunnel-Anbieter.

Wichtig: *Soll in dem Subnetz die zustandslose Selbstkonfiguration (siehe den Abschnitt zu `IPV6_NET_x_ADVERTISE` weiter unten) aktiviert werden, dann muss die Länge des Subnetz-Präfixes 64 Bit betragen!*

Wichtig: *Ist das Subnetz an einen Tunnel angeschlossen (siehe `IPV6_NET_x_TUNNEL` weiter unten), dann darf hier nur der Teil der Router-Adresse angegeben werden, der* nicht *zum dem Tunnel zugeordneten Subnetz-Präfix (zu finden in `IPV6_TUNNEL_x_PREFIX`) gehört, da jenes Präfix und diese Adresse miteinander kombiniert werden! In früheren Versionen des IPv6-Pakets gab es die Variable `IPV6_TUNNEL_x_PREFIX` noch nicht, und Subnetz-Präfix sowie Router-Adresse wurden zusammen in `IPV6_NET_x` festgelegt. Das funktioniert aber nicht, wenn das vom Tunnelanbieter zugeordnete Subnetz-Präfix erst dynamisch beim Tunnelaufbau mitgeteilt wird. Außerdem bleibt dabei die Länge des Subnetz-Präfixes (hier: /48) im Dunkeln, so dass bestimmte vordefinierte Routen nicht ordentlich gesetzt werden können, was beim Routen zu bestimmten Zieladressen dann zu seltsamen Effekten führt. Deswegen wurde die Konfiguration entsprechend aufgetrennt.*

Beispiele:

```
IPV6_NET_1='2001:db8:1743:42::1/64'       # ohne Tunnel: komplette Adresse
IPV6_NET_1_TUNNEL=''

IPV6_NET_2='0:0:0:42::1/64'               # mit Tunnel: partielle Adresse
IPV6_NET_2_TUNNEL='1'
IPV6_TUNNEL_1_PREFIX='2001:db8:1743::/48' # s. Abschnitt "Tunnel-Konfiguration"
```

IPV6_NET_x_DEV Diese Variable enthält für ein bestimmtes IPv6-Subnetz den Namen der Netzwerk-Schnittstelle, an welche das IPv6-Netz gebunden wird. Dies kollidiert *nicht* mit den in der Basis-Konfiguration (`base.txt`) vergebenen Netzwerk-Schnittstellen, da einer Netzwerk-Schnittstelle sowohl IPv4- als auch IPv6-Adressen zugeordnet werden dürfen.

Beispiel: `IPV6_NET_1_DEV='eth0'`

IPV6_NET_x_TUNNEL Diese Variable enthält für ein bestimmtes IPv6-Subnetz den Index des zugehörigen Tunnels. Das Präfix des angegebenen Tunnels wird mit der Router-Adresse kombiniert, um die vollständige IPv6-Adresse des Routers zu erhalten. Wenn die Variable leer oder nicht definiert ist, dann gehört zu dem Subnetz kein Tunnel, und in `IPV6_NET_x` muss die vollständige IPv6-Adresse des Routers inklusive Netzmaske angegeben werden (siehe oben).

Ein Tunnel kann mehreren Subnetzen zugeordnet werden, da ein Tunnel-Subnetzpräfix normalerweise groß genug ist, dass er in mehrere Subnetze aufgeteilt werden kann (/56 oder größer). Andersherum ist es natürlich nicht möglich, einem Subnetz mehrere Tunnel-Subnetzpräfixe zuzuordnen, da die Adresse des Subnetzes in diesem Fall mehrdeutig wäre.

Beispiel: `IPV6_NET_1_TUNNEL='1'`

IPV6_NET_x_ADVERTISE Diese Variable legt fest, ob das eingestellte Subnetz-Präfix im LAN mittels "Router Advertisements" verbreitet wird. Dies wird für die so genannte "stateless autoconfiguration" (zustandslose Selbstkonfiguration) verwendet und ist nicht mit DHCPv6 zu verwechseln. Mögliche Werte sind "yes" und "no".

Es ist empfehlenswert, diese Einstellung zu aktivieren, es sei denn, alle Adressen im Netz werden statisch vergeben oder ein anderer Router ist bereits dafür zuständig, das Subnetz-Präfix anzukündigen.

Wichtig: *Die automatische Verteilung des Subnetzes funktioniert nur, wenn das Subnetz ein /64-Netz ist, d.h. wenn die Länge des Subnetz-Präfixes 64 Bit beträgt! Der Grund hierfür ist, dass die anderen Hosts im Netzwerk ihre IPv6-Adresse aus dem Präfix und ihrer Host-MAC-Adresse berechnen und dies nicht funktioniert, wenn der Host-Anteil nicht 64 Bit beträgt. Wenn die Selbstkonfiguration fehlschlägt, sollte also geprüft werden, ob das Subnetz-Präfix nicht vielleicht falsch angegeben worden ist (z.B. als /48).*

Standard-Einstellung: `IPV6_NET_1_ADVERTISE='yes'`

IPV6_NET_x_ADVERTISE_DNS Diese Variable legt fest, ob auch der lokale DNS-Dienst im IPv6-Subnetz mittels "Router Advertisements" angekündigt werden soll. Dies funktioniert aber nur, wenn die IPv6-Funktionalität des DNS-Dienstes mit Hilfe der Variable `DNS_SUPPORT_IPV6`='yes' aktiviert ist. Mögliche Werte sind "yes" und "no".

Standard-Einstellung: `IPV6_NET_1_ADVERTISE_DNS='no'`

IPV6_NET_x_DHCP Diese Variable aktiviert den DHCPv6-Dienst für dieses IPv6-Subnetz. Mögliche Werte sind "yes" und "no". DHCPv6 wird dabei nur verwendet, um Hosts in diesem Subnetz Informationen zum Domänen-Namen und zur Adresse des zu verwendenden DNS-Servers zukommen zu lassen. Die Zuordnung von IPv6-Adressen ist über DHCPv6 mit fli4l momentan nicht möglich.

Die Adresse des DNS-Servers wird nur dann via DHCPv6 veröffentlicht, wenn die IPv6-Unterstützung des DNS-Dienstes via `DNS_SUPPORT_IPV6` im dns_dhcp-Paket eingeschaltet ist.

Wichtig: *Die Variablen `IPV6_NET_x_ADVERTISE_DNS` und `IPV6_NET_x_DHCP` schließen sich nicht gegenseitig aus, sondern können beide aktiviert sein. In diesem Fall kann die Adresse des DNS-Servers auf zweierlei Weise von Hosts im lokalen Netzwerk in Erfahrung gebracht werden.*

Pro Netzwerkschnittstelle darf höchstens ein gebundenes IPv6-Subnetz für DHCPv6 konfiguriert werden!

Standard-Einstellung: `IPV6_NET_1_DHCP='no'`

IPV6_NET_x_NAME (optional) Diese Variable legt einen Interface-spezifischen Hostnamen für den Router in diesem IPv6-Subnetz fest.

Beispiel: `IPV6_NET_1_NAME='fli4l-subnet1'`

Tunnel-Konfiguration

Dieser Abschnitt stellt die Konfiguration von 6in4-IPv6-Tunneln vor. Ein solcher Tunnel bietet sich an, wenn der eigene Internet-Anbieter kein IPv6 von Haus aus unterstützt. In diesem Fall wird mit einem bestimmten Internet-Host eines Tunnel-Brokers, dem so genannten PoP (Point of Presence), via IPv4 eine bidirektionale Verbindung aufgebaut, über die dann alle IPv6-Pakete verpackt geroutet werden (deswegen 6 "in" 4, weil die IPv6-Pakete innerhalb von IPv4-Paketen gekapselt werden).[3] Damit das funktioniert, muss zum einen der Tunnel aufgebaut und zum anderen der Router so konfiguriert werden, dass die IPv6-Pakete, die ins Internet sollen, auch über den Tunnel geroutet werden. Der erste Teil wird in diesem Abschnitt konfiguriert, der zweite Teil wird im nächsten Abschnitt beschrieben.

IPV6_TUNNEL_N Diese Variable enthält die Anzahl der aufzubauenden 6in4-Tunnel.

Beispiel: `IPV6_TUNNEL_N='1'`

IPV6_TUNNEL_x_TYPE Diese Variable bestimmt den Typ des Tunnels. Momentan werden die Werte "raw" für "rohe" Tunnel, "static" für statische Tunnel, "sixxs" für dynamische Heartbeat-Tunnel des Anbieters SixXS und "he" für Tunnel des Anbieters Hurricane Electric unterstützt. Mehr zu Heartbeat-Tunneln steht im nächsten Absatz.

Beispiel: `IPV6_TUNNEL_1_TYPE='sixxs'`

IPV6_TUNNEL_x_DEFAULT Diese Variable legt fest, ob IPv6-Pakete, die nicht an das lokale bzw. die lokalen Netze adressiert sind, über diesen Tunnel geroutet werden sollen. Es kann nur einen solchen Tunnel geben (weil nur eine Default-Route existieren kann). Mögliche Werte sind "yes" und "no".

Wichtig: *Genau ein Tunnel sollte ein Default-Gateway für nausgehende IPv6-Daten sein, da andernfalls eine Kommunikation mit IPv6-Hosts im Internet nicht möglich ist! Die ausschließliche Verwendung von Nicht-Default-Tunneln ist nur sinnvoll, wenn ausgehende IPv6-Daten über eine separat konfigurierte Default-Route geschickt werden, die nicht mit einem Tunnel zusammenhängt. Siehe hierzu auch die Einleitung zum Unterabschnitt "Routen-Konfiguration" sowie die Beschreibung der Variable `IPV6_ROUTE_x` weiter unten.*

Standard-Konfiguration: `IPV6_TUNNEL_1_DEFAULT='no'`

IPV6_TUNNEL_x_PREFIX Diese Variable enthält den IPv6-Subnetzpräfix des Tunnels in CIDR-Notation, d.h. es wird sowohl eine IPv6-Adresse als auch die Länge des Präfixes angegeben. Diese Angabe wird in der Regel vom Tunnelanbieter vorgegeben. Bei Tunnelanbietern, die den Präfix beim Tunnelaufbau jedes Mal neu vergeben, ist diese Angabe unnötig. (Momentan werden solche Anbieter aber noch nicht unterstützt.) Diese Variable muss auch bei rohen ("raw") Tunneln leer bleiben.

Wichtig: *Diese Variable* darf *leer bleiben, wenn dem Tunnel noch kein Subnetz-Präfix zugewiesen worden ist. Allerdings kann dieser Tunnel dann nicht einem IPv6-Subnetz (`IPV6_NET_x`) zugeordnet werden, weil die IPv6-Adressen im Subnetz nicht berechnet werden können. Sinnvoll ist eine solche Konfiguration also nur übergangsweise, etwa wenn*

[3] Es handelt sich um das IPv4-Protokoll 41, "IPv6 encapsulation".

der Tunnel einige Zeit aktiv sein muss, bevor der Tunnelanbieter einem ein Subnetz-Präfix zuweist (diese Vorgehensweise ist z.B. beim Tunnelanbieter SixXS üblich).

Beispiele:

```
IPV6_TUNNEL_1_PREFIX='2001:db8:1743::/48'      # /48-Subnetz
IPV6_TUNNEL_2_PREFIX='2001:db8:1743:5e00::/56' # /56-Subnetz
```

IPV6_TUNNEL_x_LOCALV4 Diese Variable enthält die lokale IPv4-Adresse des Tunnels oder den Wert 'dynamic', wenn die dynamisch zugewiesene IPv4-Adresse des aktiven WAN-Circuits verwendet werden soll. Letzteres ist nur sinnvoll, wenn es sich um einen Heartbeat-Tunnel handelt (siehe `IPV6_TUNNEL_x_TYPE` weiter unten).

Beispiele:

```
IPV6_TUNNEL_1_LOCALV4='172.16.0.2'
IPV6_TUNNEL_2_LOCALV4='dynamic'
```

IPV6_TUNNEL_x_REMOTEV4 Diese Variable enthält die entfernte IPv4-Adresse des Tunnels. Diese Angabe wird in der Regel vom Tunnel-Anbieter vorgegeben.

Beispiel (entspricht dem PoP deham01 von Easynet):

```
IPV6_TUNNEL_1_REMOTEV4='212.224.0.188'
```

Wichtig: *Wenn `PF_INPUT_ACCEPT_DEF` auf "no" steht, d.h. wenn die IPv4-Firewall manuell konfiguriert wird, dann wird eine Regel benötigt, die alle IPv6-in-IPv4-Pakete (IP-Protokoll 41) vom Tunnelendpunkt akzeptiert. Für den o.g. Tunnelendpunkt sähe die entsprechende Regel wie folgt aus:*

```
PF_INPUT_x='prot:41 212.224.0.188 ACCEPT'
```

IPV6_TUNNEL_x_LOCALV6 Diese Variable legt die lokale IPv6-Adresse des Tunnels inklusive verwendeter Netzmaske in CIDR-Notation fest. Diese Angabe wird vom Tunnelanbieter vorgegeben. Bei Tunnelanbietern, welche die Tunnelendpunkte beim Tunnelaufbau jedes Mal neu vergeben, ist diese Angabe unnötig. (Momentan werden solche Anbieter aber noch nicht unterstützt.)

Beispiel: `IPV6_TUNNEL_1_LOCALV6='2001:db8:1743::2/112'`

IPV6_TUNNEL_x_REMOTEV6 Diese Variable legt die entfernte IPv6-Adresse des Tunnels fest. Diese Angabe wird vom Tunnelanbieter vorgegeben. Eine Netzmaske wird nicht benötigt, da sie der Variable `IPV6_TUNNEL_x_LOCALV6` entnommen wird. Bei Tunnelanbietern, welche die Tunnelendpunkte beim Tunnelaufbau jedes Mal neu vergeben, ist diese Angabe unnötig. (Momentan werden solche Anbieter aber noch nicht unterstützt.)

Beispiel: `IPV6_TUNNEL_1_REMOTEV6='2001:db8:1743::1'`

IPV6_TUNNEL_x_DEV (optional) Diese Variable enthält den Namen der zu erstellenden Tunnel-Netzwerkschnittstelle. Verschiedene Tunnel müssen unterschiedlich benannt werden, damit alles funktioniert. Falls die Variable nicht definiert ist, wird ein Tunnelname automatisch generiert ("v6tun" + Tunnelindex).

Beispiel: `IPV6_TUNNEL_1_DEV='6in4'`

IPV6_TUNNEL_x_MTU (optional) Diese Variable enthält die Größe der MTU (Maximum Transfer Unit) in Bytes, d.h. des größten Pakets, das noch getunnelt werden kann. Diese Angabe wird in der Regel vom Tunnelanbieter vorgegeben. Die Standard-Einstellung, falls nichts angegeben wird, lautet "1280" und sollte mit allen Tunneln funktionieren.

Standard-Konfiguration: `IPV6_TUNNEL_1_MTU='1280'`

Einige Tunnelanbieter verlangen, dass über den Tunnel permanent ein Lebenszeichen vom Router an den Anbieter gesandt wird, um zu verhindern, dass ein Host einen Tunnel in Anspruch nimmt, obwohl er ihn nicht nutzt. Dazu wird ein so genanntes Heartbeat-Protokoll (dt. "Herzschlag") verwendet. Zusätzlich verlangen Anbieter in der Regel eine erfolgreiche Anmeldung mit Benutzernamen und Passwort, um Missbrauch zu vermeiden. Soll ein solcher Heartbeat-Tunnel genutzt werden (wie ihn etwa SixXS anbietet), dann müssen entsprechende Angaben gemacht werden, die im Folgenden beschrieben werden.

IPV6_TUNNEL_x_USERID Diese Variable enthält den Namen des Benutzers, der beim Tunnel-Login erforderlich ist.

Beispiel: `IPV6_TUNNEL_1_USERID='ABCDE-SIXXS'`

IPV6_TUNNEL_x_PASSWORD Diese Variable enthält das Passwort für den oben angegebenen Benutzernamen. Es darf keine Leerzeichen enthalten.

Beispiel: `IPV6_TUNNEL_1_PASSWORD='passwort'`

IPV6_TUNNEL_x_TUNNELID Diese Variable enthält den Identifikator des Tunnels. Der Name eines SixXS-Tunnels beginnt immer mit einem großen 'T'.

Beispiel: `IPV6_TUNNEL_1_TUNNELID='T1234'`

IPV6_TUNNEL_x_TIMEOUT (optional) Diese Variable enthält die Zeitspanne in Sekunden, die beim Tunnelaufbau maximal gewartet wird. Der Standard-Wert ist abhängig vom eingestellten Tunnelanbieter.

Beispiel: `IPV6_TUNNEL_1_TIMEOUT='30'`

Routen-Konfiguration

Routen sind Wege für IPv6-Pakete. Damit der Router weiß, welches eingehende Paket er wohin schicken soll, greift er auf eine Routing-Tabelle zurück, in der genau diese Informationen zu finden sind. Im Falle von IPv6 ist es wichtig zu wissen, wohin IPv6-Pakete geschickt werden, die nicht ins lokale Netz sollen. Eine Default-Route, die alle Pakete an das andere Ende eines IPv6-Tunnels schickt, wird automatisch konfiguriert, wenn `IPV6_TUNNEL_x_DEFAULT` für den entsprechenden Tunnel gesetzt ist. Andere Routen, die etwa benachbarte IPv6-Subnetze miteinander verbinden, können hier konfiguriert werden.

IPV6_ROUTE_N Diese Variable legt die Anzahl der zu spezifizierenden IPv6-Routen fest. In der Regel werden keine zusätzlichen IPv6-Routen benötigt.

Standard-Einstellung: `IPV6_ROUTE_N='0'`

IPV6_ROUTE_x Diese Variable enthält die Route in der Form 'Zielnetz Gateway', wobei das Zielnetz in der CIDR-Notation erwartet wird. Für die Default-Route muss als Zielnetz `::/0` verwendet werden. Es ist aber nicht nötig, Default-Routen, die über einen Tunnel gehen, hier zu konfigurieren (siehe Einleitung zu diesem Abschnitt).

Beispiel: `IPV6_ROUTE_1='2001:db8:1743:44::/64␣2001:db8:1743:44::1'`

IPv6-Firewall

Wie für IPv4 wird auch für IPv6-Netzwerke eine Firewall benötigt, damit nicht jeder von außen jeden Rechner im lokalen Netz erreichen kann. Dies ist um so wichtiger, als dass jeder Rechner im Normalfall eine weltweit eindeutige IPv6-Adresse erhält, die dem Rechner permanent zugeordnet werden kann, da sie auf der MAC-Adresse der verwendeten Netzwerkkarte aufbaut.[4] Deshalb verbietet die Firewall erst einmal jegliche Zugriffe von außen und kann dann durch entsprechende Einträge in diesem Abschnitt Stück für Stück – je nach Bedarf – geöffnet werden.

Die Konfiguration der IPv6-Firewall entspricht im Großen und Ganzen der Konfiguration der IPv4-Firewall. Auf Besonderheiten und Unterschiede wird gesondert eingegangen.

PF6_LOG_LEVEL Für alle folgenden Ketten gilt die in `PF6_LOG_LEVEL` vorgenommene Einstellung der Protokoll-Stufe, deren Inhalt auf einen der folgenden Werte gesetzt werden kann: debug, info, notice, warning, err, crit, alert, emerg.

PF6_INPUT_POLICY Diese Variable legt die Standard-Strategie für auf dem Router eingehende Pakete fest (INPUT-Kette). Mögliche Werte sind "REJECT" (Standard, weist alle Pakete ab), "DROP" (verwirft klammheimlich alle Pakete) und "ACCEPT" (akzeptiert alle Pakete). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_POLICY`.

Standard-Einstellung: `PF6_INPUT_POLICY='REJECT'`

PF6_INPUT_ACCEPT_DEF Diese Variable aktiviert die voreingestellten Regeln für die INPUT-Kette der IPv6-Firewall. Mögliche Werte sind "yes" und "no".

Die voreingestellten Regeln öffnen die Firewall für eingehende ICMPv6-Pings (ein Ping pro Sekunde als Limit) sowie für NDP-Pakete (Neighbour Discovery Procotol), das zur zustandslosen Selbstkonfiguration von IPv6-Netzen benötigt wird. Verbindungen von localhost sowie Antwortpakete zu lokal initiierten Verbindungen werden ebenfalls erlaubt. Schließlich wird die IPv4-Firewall dahingehend angepasst, dass für jeden Tunnel gekapselte IPv6-in-IPv4-Pakete vom Tunnelendpunkt akzeptiert werden.

Standard-Einstellung: `PF6_INPUT_ACCEPT_DEF='yes'`

[4] Eine Ausnahme existiert, wenn auf den LAN-Hosts die so genannten "Privacy Extensions" aktiviert werden, weil dann ein Teil der IPv6-Adresse zufällig generiert wird. Diese Adressen sind jedoch per Definition nicht nach außen hin bekannt und somit für die Firewall-Konfiguration nur bedingt bis gar nicht relevant.

PF6_INPUT_LOG Diese Variable aktiviert das Logging aller zurückgewiesenen eingehenden Pakete. Mögliche Werte sind “yes” und “no”. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_LOG`.

Standard-Einstellung: `PF6_INPUT_LOG='no'`

PF6_INPUT_LOG_LIMIT Diese Variable konfiguriert das Log-Limit der INPUT-Kette der IPv6-Firewall, um die Log-Datei lesbar zu halten. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_LOG_LIMIT`.

Standard-Einstellung: `PF6_INPUT_LOG_LIMIT='3/minute:5'`

PF6_INPUT_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von eingehenden TCP-Paketen ein. Überschreitet ein solches Paket dieses Limit, wird das Paket klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_REJ_LIMIT`.

Standard-Einstellung: `PF6_INPUT_REJ_LIMIT='1/second:5'`

PF6_INPUT_UDP_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von eingehenden UDP-Paketen ein. Überschreitet ein solches UDP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_UDP_REJ_LIMIT`.

Standard-Einstellung: `PF6_INPUT_UDP_REJ_LIMIT='1/second:5'`

PF6_INPUT_ICMP_ECHO_REQ_LIMIT Definiert, wie häufig auf eine ICMPv6-Echo-Anfrage reagiert werden soll. Die Häufigkeit wird analog zur Limit-Einschränkung als ‘n/Zeiteinheit’ mit Bursts beschrieben, also z.B. ’3/minute:5’. Ist das Limit überschritten, wird das Paket einfach ignoriert (DROP). Ist dieser Eintrag leer, wird der Standardwert ’1/second:5’ verwendet; enhält er ’none’, wird keine Limitierung durchgeführt.

Standard-Einstellung: `PF6_INPUT_ICMP_ECHO_REQ_LIMIT='1/second:5'`

PF6_INPUT_ICMP_ECHO_REQ_SIZE Definiert, wie groß eine empfangene ICMPv6-Echo-Anfrage sein darf (in Bytes). In dieser Angabe sind neben den “Nutzdaten” auch die Paket-Header mit zu berücksichtigen. Der Standard-Wert liegt bei 150 Bytes.

Standard-Einstellung: `PF6_INPUT_ICMP_ECHO_REQ_SIZE='150'`

PF6_INPUT_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln für eingehende Pakete (INPUT-Kette). Standardmäßig werden zwei Regeln aktiviert: Die erste erlaubt allen lokalen Hosts Zugriff auf den Router über so genannte Link-Level-Adressen, und die zweite erlaubt die Kommunikation von Hosts aus dem ersten definierten IPv6-Subnetz mit dem Router.

Falls mehrere lokale IPv6-Subnetze definiert werden, muss die zweite Regel entsprechend oft vervielfältigt werden. Siehe hierzu die Konfigurationsdatei.

Beispiel: `PF6_INPUT_N='2'`

PF6_INPUT_x Diese Variable spezifiziert eine Regel für die INPUT-Kette der IPv6-Firewall. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

Beispiele:

```
PF6_INPUT_1='[fe80::0/10] ACCEPT'
PF6_INPUT_2='IPV6_NET_1 ACCEPT'
PF6_INPUT_3='tmpl:samba DROP NOLOG'
```

PF6_INPUT_x_COMMENT Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen INPUT-Regel.

Beispiel: `PF6_INPUT_3_COMMENT='no␣samba␣traffic␣allowed'`

PF6_FORWARD_POLICY Diese Variable legt die Standard-Strategie für von dem Router weiterzuleitenden Pakete fest (FORWARD-Kette). Mögliche Werte sind “REJECT” (Standard, weist alle Pakete ab), “DROP” (verwirft klammheimlich alle Pakete) und “ACCEPT” (akzeptiert alle Pakete). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_POLICY`.

Standard-Einstellung: `PF6_FORWARD_POLICY='REJECT'`

PF6_FORWARD_ACCEPT_DEF Diese Variable aktiviert die voreingestellten Regeln für die FORWARD-Kette der IPv6-Firewall. Mögliche Werte sind “yes” und “no”.

Die voreingestellten Regeln öffnen die Firewall für ausgehende ICMPv6-Pings (ein Ping pro Sekunde als Limit). Antwortpakete zu bereits erlaubten Verbindungen werden ebenfalls erlaubt.

Standard-Einstellung: `PF6_FORWARD_ACCEPT_DEF='yes'`

PF6_FORWARD_LOG Diese Variable aktiviert das Logging aller zurückgewiesenen weterzuleitenden Pakete. Mögliche Werte sind “yes” und “no”. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_LOG`.

Standard-Einstellung: `PF6_FORWARD_LOG='no'`

PF6_FORWARD_LOG_LIMIT Diese Variable konfiguriert das Log-Limit der FORWARD-Kette der IPv6-Firewall, um die Log-Datei lesbar zu halten. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_LOG_LIMIT`.

Standard-Einstellung: `PF6_FORWARD_LOG_LIMIT='3/minute:5'`

PF6_FORWARD_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von weiterzuleitenden TCP-Paketen ein. Überschreitet ein solches TCP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_REJ_LIMIT`.

Standard-Einstellung: `PF6_FORWARD_REJ_LIMIT='1/second:5'`

PF6_FORWARD_UDP_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von weiterzuleitenden UDP-Paketen ein. Überschreitet ein solches UDP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_UDP_REJ_LIMIT`.

Standard-Einstellung: `PF6_FORWARD_UDP_REJ_LIMIT='1/second:5'`

PF6_FORWARD_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln für weiterzuleitende Pakete (FORWARD-Kette). Standardmäßig werden zwei Regeln aktiviert: Die erste verhindert die Weiterleitung aller lokalen Samba-Pakete in nicht-lokale Netze, und die zweite erlaubt letzteres für alle anderen lokalen Pakete aus dem ersten definierten IPv6-Subnetz.

Falls mehrere lokale IPv6-Subnetze definiert werden, muss die letzte Regel entsprechend oft vervielfältigt werden. Siehe hierzu die Konfigurationsdatei.

Beispiel: `PF6_FORWARD_N='2'`

PF6_FORWARD_x Diese Variable spezifiziert eine Regel für die FORWARD-Kette der IPv6-Firewall. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

Beispiele:

```
PF6_FORWARD_1='tmpl:samba DROP'
PF6_FORWARD_2='IPV6_NET_1 ACCEPT'
```

PF6_FORWARD_x_COMMENT Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen FORWARD-Regel.

Beispiel: `PF6_FORWARD_1_COMMENT='no␣samba␣traffic␣allowed'`

PF6_OUTPUT_POLICY Diese Variable legt die Standard-Strategie für vom Router ausgehende Pakete fest (OUTPUT-Kette). Mögliche Werte sind “REJECT” (Standard, weist alle Pakete ab), “DROP” (verwirft klammheimlich alle Pakete) und “ACCEPT” (akzeptiert alle Pakete). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_POLICY`.

Standard-Einstellung: `PF6_OUTPUT_POLICY='REJECT'`

PF6_OUTPUT_ACCEPT_DEF Diese Variable aktiviert die voreingestellten Regeln für die OUTPUT-Kette der IPv6-Firewall. Mögliche Werte sind “yes” und “no”. Momentan existieren keine voreingestellten Regeln.

Standard-Einstellung: `PF6_OUTPUT_ACCEPT_DEF='yes'`

PF6_OUTPUT_LOG Diese Variable aktiviert das Logging aller zurückgewiesenen ausgehenden Pakete. Mögliche Werte sind “yes” und “no”. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_LOG`.

Standard-Einstellung: `PF6_OUTPUT_LOG='no'`

PF6_OUTPUT_LOG_LIMIT Diese Variable konfiguriert das Log-Limit der OUTPUT-Kette der IPv6-Firewall, um die Log-Datei lesbar zu halten. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_LOG_LIMIT`.

Standard-Einstellung: `PF6_OUTPUT_LOG_LIMIT='3/minute:5'`

PF6_OUTPUT_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von ausgehenden TCP-Paketen ein. Überschreitet ein solches Paket dieses Limit, wird das Paket klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_REJ_LIMIT`.

Standard-Einstellung: `PF6_OUTPUT_REJ_LIMIT='1/second:5'`

PF6_OUTPUT_UDP_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von ausgehenden UDP-Paketen ein. Überschreitet ein solches UDP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_UDP_REJ_LIMIT`.

Standard-Einstellung: `PF6_OUTPUT_UDP_REJ_LIMIT='1/second:5'`

PF6_OUTPUT_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln für eingehende Pakete (OUTPUT-Kette). Standardmäßig werden zwei Regeln aktiviert: Die erste erlaubt allen lokalen Hosts Zugriff auf den Router über so genannte Link-Level-Adressen, und die zweite erlaubt die Kommunikation von Hosts aus dem ersten definierten IPv6-Subnetz mit dem Router.

Falls mehrere lokale IPv6-Subnetze definiert werden, muss die zweite Regel entsprechend oft vervielfältigt werden. Siehe hierzu die Konfigurationsdatei.

Beispiel: `PF6_OUTPUT_N='1'`

PF6_OUTPUT_x Diese Variable spezifiziert eine Regel für die OUTPUT-Kette der IPv6-Firewall. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

Beispiele:

```
PF6_OUTPUT_1='tmpl:ftp IPV6_NET_1 ACCEPT HELPER:ftp'
```

PF6_OUTPUT_x_COMMENT Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen OUTPUT-Regel.

Beispiel: `PF6_OUTPUT_3_COMMENT='no␣samba␣traffic␣allowed'`

PF6_USR_CHAIN_N Diese Variable enthält die Anzahl der vom Benutzer definierten IPv6-Firewall-Tabellen. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_N`.

Standard-Einstellung: `PF6_USR_CHAIN_N='0'`

PF6_USR_CHAIN_x_NAME Diese Variable enthält den Namen der entsprechenden benutzerdefinierten IPv6-Firewall-Tabelle. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_x_NAME`.

Beispiel: `PF6_USR_CHAIN_1_NAME='usr-myvpn'`

PF6_USR_CHAIN_x_RULE_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln in der zugehörigen benutzerdefinierten IPv6-Firewall-Tabelle. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_x_RULE_N`.

Beispiel: `PF6_USR_CHAIN_1_RULE_N='0'`

PF6_USR_CHAIN_x_RULE_x Diese Variable spezifiziert eine Regel für die benutzerdefinierte IPv6-Firewall-Tabelle. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_x_RULE_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

PF6_USR_CHAIN_x_RULE_x_COMMENT Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen Regel.

Beispiel: `PF6_USR_CHAIN_1_RULE_1_COMMENT='some␣user-defined␣rule'`

PF6_POSTROUTING_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln fürs Maskieren (POSTROUTING-Kette). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_POSTROUTING_N`.

Beispiel: `PF6_POSTROUTING_N='2'`

PF6_POSTROUTING_x PF6_POSTROUTING_x_COMMENT
Eine Liste der Regeln, die beschreiben, welche IPv6-Pakete vom Router maskiert werden (bzw. unmaskiert weitergeleitet werden). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_POSTROUTING_x`.

PF6_PREROUTING_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln fürs Weiterleiten an ein anderes Ziel (PREROUTING-Kette). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_PREROUTING_N`.

Beispiel: `PF6_PREROUTING_N='2'`

PF6_PREROUTING_x PF6_PREROUTING_x_COMMENT
Eine Liste der Regeln, die beschreiben, welche IPv6-Pakete vom Router an ein anderes Ziel weitergeleitet werden sollen. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_PREROUTING_x`.

1.1.4. Web-GUI

Das Paket installiert einen Menüeintrag “Paketfilter (IPv6)” im Mini-HTTPD, unter dem die Einträge des für IPv6 konfigurierten Paketfilters eingesehen werden können.

A. Anhang zum Paket IPV6

A.1. IPV6 - Anbindung ans IPv6-Internet mit Hilfe eines SixXS-Tunnels

In diesem Abschnitt wird beschrieben, wie das IPv6-Paket genutzt werden kann, um mit Hilfe eines Tunnels des Anbieters SixXS (https://www.sixxs.net/) das eigene Heimnetzwerk mit dem IPv6-Internet zu verbinden.

A.1.1. Account erstellen

Zuerst muss ein SixXS-Account unter “Signup for new users” beantragt werden. Hat man diese Hürde genommen, besitzt man einen Benutzernamen der Form YYYYY-SIXXS sowie ein zugehöriges Passwort. Diese Daten benötigt man später für die Tunnelkonfiguration.

A.1.2. Tunnel konfigurieren

Vorbereitungen

Doch vorher muss man den Tunnel beantragen. Dies geschieht nach der Anmeldung über den Menüpunkt “Request tunnel”. Hier ist es wichtig, beim Tunneltyp den zweiten Eintrag, “Dynamic IPv4 Endpoint using Heartbeat protocol” auszuwählen, weil diese Konfiguration vom fli4l direkt unterstützt wird. Die dritte Variante, “Static IPv4 Endpoint”, ist ebenfalls möglich, wenn man eine fest zugeordnete IPv4-Adresse sein Eigen nennt, die sich nie ändert. Die Tunnelvariante “Dynamic NAT-traversing IPv4 Endpoint using AYIYA” wird derzeit vom IPv6-Paket nicht unterstützt.

Sobald man in den anderen Feldern Angaben zum Ort des Routers gemacht und via “Next step” zur zweiten Seite gewechselt hat, stehen hier ein oder mehrere PoPs (Points of Presence) zur Auswahl, die später für den Tunnelaufbau wichtig sind. Man sollte denjenigen nehmen, der am dichtesten ist, um das Tunneln von IPv6-Paketen möglichst effizient zu gestalten.

Sind alle Angaben gemacht und via “Place request for new Tunnel” abgeschickt worden, kommt irgendwann darauf eine E-Mail mit den nötigen Tunneldaten an. Dazu gehören:

1. die Identifikationsnummer des Tunnels (T...)
2. der Name des zugeordneten PoPs
3. die IPv4-Adresse des zugeordneten PoPs (“SixXS IPv4”)
4. die lokale IPv6-Adresse des Tunnels inklusive Subnetzmaske (bei SixXS typischerweise /64), also die Adresse des Routers (“Your IPv6”)
5. die entfernte IPv6-Adresse des Tunnels inklusive Subnetzmaske (die mit der Subnetzmaske der lokalen IPv6-Adresse identisch ist), d.h. die Adresse des PoPs (“SixXS IPv6”)

Konfiguration

Jetzt kann der Tunnel konfiguriert werden! Als erstes wird die Variable `IPV6_TUNNEL_N` auf "1" gesetzt, weil genau ein Tunnel aufgebaut werden soll:

```
IPV6_TUNNEL_N='1'
```

Die SixXS-Angaben werden wie folgt in der IPv6-Konfiguration vermerkt:

1. Die Identifikationsnummer des Tunnels wird in `IPV6_TUNNEL_1_TUNNELID` eingetragen.
2. entfällt (der Name des PoPs ist uninteressant)
3. Die IPv4-Adresse des PoPs wird in der Variable `IPV6_TUNNEL_1_REMOTEV4` vermerkt.
4. Die lokale IPv6-Adresse des Tunnels landet *inklusive* Subnetzmaske in der Variable `IPV6_TUNNEL_1_LOCALV6`.
5. Die entfernte IPv6-Adresse des Tunnels landet *ohne* Subnetzmaske in der Variable `IPV6_TUNNEL_1_REMOTEV6`.

Zusätzlich müssen Benutzername und Passwort in der Tunnelkonfiguration in den Variablen `IPV6_TUNNEL_1_USERID` und `IPV6_TUNNEL_1_PASSWORD` angegeben werden. Schließlich muss in der Variable `IPV6_TUNNEL_1_TYPE` vermerkt werden, dass der konfigurierte Tunnel ein SixXS-Tunnel ist:

```
IPV6_TUNNEL_1_TYPE='sixxs'
```

Hat man von SixXS den PoP "deham01" mit der IPv4-Adresse `212.224.0.188` sowie die Tunnelendpunkte `2001:db8:900:551::1/64` (entfernt) und `2001:db8:900:551::2/64` (lokal) erhalten, und lauten die Tunnel-ID "T1234", der Benutzername "USER1-SIXXS" und das Passwort "sixxs" (bitte dieses Passwort *nicht* verwenden!), dann sieht die resultierende Konfiguration so aus:

```
IPV6_TUNNEL_N='1'
IPV6_TUNNEL_1_LOCALV4='dynamic' # oder feste lokale IPv4-Adresse
IPV6_TUNNEL_1_REMOTEV4='212.224.0.188'
IPV6_TUNNEL_1_LOCALV6='2001:db8:900:551::2/64'
IPV6_TUNNEL_1_REMOTEV6='2001:db8:900:551::1'
IPV6_TUNNEL_1_TYPE='sixxs'
IPV6_TUNNEL_1_USERID='USER1-SIXXS'
IPV6_TUNNEL_1_PASSWORD='sixxs'
IPV6_TUNNEL_1_TUNNELID='T1234'
```

Test

Hat man dies geschaffft, dann kann man nach Aktualisierung der Konfiguration den fli4l-Router neu starten. Nach dem Einloggen auf dem Router (direkt oder z.B. per SSH) sollte man den Tunnelendpunkt bereits anpingen können. Das Ganze sieht dann mit den oben genannten Beispiel-Daten folgendermaßen aus:

```
garm 3.6.0-revXXXXX # ping -c 4 2001:db8:900:551:0:0:0:1
PING 2001:db8:900:551::1 (2001:db8:900:551::1): 56 data bytes
64 bytes from 2001:db8:900:551::1: seq=0 ttl=64 time=67.646 ms
64 bytes from 2001:db8:900:551::1: seq=1 ttl=64 time=72.001 ms
64 bytes from 2001:db8:900:551::1: seq=2 ttl=64 time=70.082 ms
64 bytes from 2001:db8:900:551::1: seq=3 ttl=64 time=67.996 ms

--- 2001:db8:900:551::1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 67.646/69.431/72.001 ms
```

Wichtig ist in der vorletzten Zeile der Teil “0% packet loss”, d.h. dass für alle PING-Pakete Antwortpakete empfangen wurden. Kommt keine Antwort vom anderen Ende des Tunnels, sieht das Ergebnis anders aus:

```
garm 3.6.0-revXXXXX # ping -c 4 2001:db8:900:551:0:0:0:1
PING 2001:db8:900:551::1 (2001:db8:900:551::1): 56 data bytes

--- 2001:db8:900:551::1 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

Hier fällt auf, dass für kein einziges PING-Paket eine Antwort empfangen wurde (“100% packet loss”). Das bedeutet, dass entweder die Konfiguration nicht korrekt ist, oder dass der Tunnel seitens SixXS noch nicht vollständig eingerichtet worden ist. Im zweiten Fall sollte man erst einige Zeit abwarten, weil die Konfiguration auf Seiten des PoPs durchaus ein paar Stunden dauern kann. Hat man die Konfiguration doppelt und dreifach geprüft und keinen Fehler entdeckt, und ist einige Zeit verstrichen, ohne dass der Tunnel funktioniert, sollte man sich per E-Mail an SixXS wenden und das Problem möglichst detailliert beschreiben.

A.1.3. Subnetz konfigurieren

Vorbereitungen

Funktioniert der Tunnel, hat man den ersten großen Punkt geschafft. Fertig ist man aber noch nicht. Denn nun hat der Router zwar die Möglichkeit, Pakete ins IPv6-Internet zu schicken und von dort zu empfangen, aber die Hosts im lokalen Netz noch nicht. Dafür muss erst ein IPv6-Subnetz konfiguriert werden, innerhalb dessen die Hosts eingebunden werden.

Hier fällt ein kleiner, aber wesentlicher Unterschied zur Konfiguration eines IPv4-Netzwerks auf: Wegen der Adressknappheit ist in der Regel nur ein Host direkt mit dem Internet verbunden. Die anderen Hosts im lokalen Netz erhalten nur netzinterne, d.h. nicht nach außen geroutete Adressen aus den Bereichen `192.168.*.*`, `172.16.*.*` bis `172.31.*.*` sowie `10.*.*.*`, je nach Größe des Subnetzes.[1] Bei IPv6 gibt es Adressen in Hülle und Fülle, somit entfällt die Notwendigkeit, netzinterne Adressen zu verwenden. Wegen des globalen Charakters lokaler Subnetze muss jedoch sichergestellt werden, dass die Adressen lokaler Hosts nicht mit anderen Adressen im Internet kollidieren. Deshalb muss ein Subnetz vom IPv6-Anbieter zugeteilt werden, um solche Kollisionen zu vermeiden.

Bei SixXS geschieht dies mit dem Menüpunkt “Request subnet”. Hier muss man hauptsächlich den zu verwendenden Tunnel angeben, was leicht ist, weil bisher nur einer konfiguriert

[1] siehe RFC 1918 (http://tools.ietf.org/html/rfc1918) für Details

worden ist. Nach dem Abschicken des Formulars via “Place request for new subnet” erhält man nach einiger Zeit eine E-Mail mit den folgenden Informationen:

1. Die IPv6-Adresse des Subnetzes inklusive Subnetzmaske (“Subnet IPv6”)
2. Die IPv6-Adresse des Routers im Tunnel, wohin das Subnetz seitens SixXS geroutet wird (“Routed to”)
3. Die IPv4-Adresse des Routers (“Your IPv4”)[2]

Diese Daten reichen aus, beim fli4l jetzt ein eigenes IPv6-Subnetz zu konfigurieren. Allerdings muss man eines noch wissen: Das zugewiesene Subnetz ist in der Regel sehr groß. SixXS teilt für gewöhnlich /48er-Subnetze zu, d.h. innerhalb der 128 Bit langen IPv6-Adresse belegt der Anteil, der auf das Netzpräfix fällt, 48 Bit, und der Anteil, der für die Adressierung der Hosts zur Verfügung steht, beträgt 128 - 48 = 80 Bit. Ein solch großes Subnetz hat jedoch zwei große Nachteile. Der erste Nachteil ist die schiere Größe: Man kann in dem Netz $2^{80} \approx 1209$ Trilliarden Hosts unterkriegen. Das zu verwalten, ohne eine weitere Struktur auf dem Hosts-Anteil der Adresse zu nutzen, erscheint nicht ratsam. Der zweite Nachteil wiegt schwerer: Innerhalb eines solch großen Subnetzes funktioniert die so genannte *IPv6-Autokonfiguration* nicht mehr. Das ist ein Vorgang, bei dem ein IPv6-Host über bestimmte Protokolle das Subnetz-Präfix erhält und sich seine IPv6-Adresse mit Hilfe der MAC-Adresse seines Netzwerk-Adapters zurechtbastelt. Die MAC-Adresse besteht aus sechs Bytes, und mit Hilfe des Standards EUI-64 kann man sie auf acht Bytes strecken. Das entspricht 64 Bit, und dann ist Schluss. Für 80 Bit stehen einfach nicht genügend Informationen seitens des Hosts zur Verfügung.

Lange Rede, kurzer Sinn: Das Subnetz muss kleiner gemacht werden, und zwar muss, damit später die Autokonfiguration auch funktionieren kann, daraus ein /64er-Subnetz werden. Das ist ganz einfach: Die Subnetzmaske wird einfach zu /64 abgeändert. Wurde also von SixXS z.B. das Subnetz `2001:db8:123::/48` zugewiesen, dann ist das Subnetz, für das der fli4l konfiguriert werden soll, einfach `2001:db8:123::/64`. Das bedeutet im Detail, dass das /48-Subnetz in $2^{(64-48)} = 2^{16} = 65536$ Sub-Subnetze eingeteilt wird, von denen das erste mit der Nummer Null vom fli4l verwendet werden soll. Dazu muss man sich erinnern, dass die Kurzform `2001:db8:123::` eigentlich für die Adresse `2001:db8:123:0:0:0:0:0` steht, wobei die ersten drei Zahlen die vom IPv6-Anbieter global eindeutig vergebenen Teile des Subnetzes sind, die vierte Zahl das eigens ausgesuchte Sub-Subnetz “Null” darstellt,[3] und die letzten vier Zahlen für den Host-Anteil reserviert sind. Das ergibt dann zwar immer noch ein riesiges (Sub-)Subnetz, in dem bis zu $2^{64} \approx 18,4$ Trillionen Hosts untergebracht werden können. Dank der IPv6-Autokonfiguration kommt man aber mit den tatsächlichen Adressen nicht in Berührung. Und das ist gut so...

Konfiguration

Zurück zur Konfiguration! Als erstes wird die Variable `IPV6_NET_N` auf “1” gesetzt, weil genau ein lokales IPv6-Subnetz eingerichtet werden soll. Die IPv6-Adresse des /64-Subnetzes landet inklusive Subnetzmaske in der Variable `IPV6_NET_1`. Doch das ist nicht ganz richtig: Vielmehr steht hier die IPv6-Adresse *des Routers innerhalb dieses Subnetzes*, aber *ohne* das Subnetz-Präfix, das dem Tunnel zugeordnet ist. Das wird nämlich an anderer Stelle konfiguriert, und

[2] falls der Router die IPv4-Adresse dynamisch erhält, steht hier “heartbeat”
[3] Man kann natürlich sich für ein anderes Sub-Subnetz entscheiden!

zwar in der Tunnelkonfiguration. Dort muss nun die Variable `IPV6_TUNNEL_1_PREFIX` auf das angoforderte Subnetz-Präfix gesetzt werden.

Hat man nun das /48er-IPv6-Subnetz `2001:db8:123::/48` von SixXS erhalten, daraus das Subnetz mit der Nummer '456' als zu verwendendes /64er-Sub-Subnetz ausgewählt und schließlich bestimmt, dass der Router innerhalb dieses Subnetzes die Adresse "1" erhalten soll, dann erhalten wir die folgende Konfiguration:

```
IPV6_NET_N='1'
IPV6_NET_1='0:0:0:456::1/64'             # IPv6-Adresse des Routers (ohne
                                         # Subnetz-Präfix) + Subnetzmaske
IPV6_TUNNEL_1_PREFIX='2001:db8:123::/48' # /48-Subnetz-Präfix
```

Zu beachten ist, dass die ersten drei Nullen in `IPV6_NET_1` quasi den Platz für das dem Tunnel zugeordnete /48-Subnetz-Präfix freihalten. Zusammen mit dem /48-Subnetz-Präfix, der vom Tunnelanbieter zugewiesen wird, ergeben sich das /64-Subnetz `2001:db8:123:456::/64` und die IPv6-Routeradresse `2001:db8:123:456::1`.

Jetzt fehlt noch der Name der Netzwerkschnittstelle, an die dieses Subnetz gebunden werden soll. Jedes Subnetz wird genau einer Netzwerkschnittstelle zugeordnet. Falls sich nur eine konfigurierte Netzwerkkarte im Router befindet, so lautet der Name der Netzwerkschnittstelle typischerweise "eth0" für kabelgebundene oder "wlan0" für drahtloses Adapter. Schauen Sie im Zweifelsfall einfach die Einstellung für `IP_NET_1_DEV` ("IP" ohne "6") und kopieren Sie den Inhalt einfach herüber:

```
IPV6_NET_1_DEV='eth0' # Netzwerkschnittstelle für dieses IPv6-Subnetz
```

Schließlich müssen wir nur noch alle Hebel der IPv6-Autokonfiguration ziehen:

```
IPV6_NET_1_ADVERTISE='yes'     # /64-Subnetz-Präfix und Default-Route per RA
IPV6_NET_1_ADVERTISE_DNS='yes' # DNS-Server per RA (erfordert
                               # DNS_SUPPORT_IPV6='yes'!)
IPV6_NET_1_DHCP='yes'          # Domänen-Name und DNS-Server per DHCPv6
                               # (letzteres erfordert DNS_SUPPORT_IPV6='yes')
```

Die beiden letzten Einstellungen sind nicht zwingend notwendig für ein funktionierendes IPv6-Subnetz, sind aber ganz hilfreich. Sie dienen der Verbreitung zusätzlicher Informationen im IPv6-Subnetz, nämlich der IPv6-Adresse des DNS-Servers sowie der verwendeten Domäne. Den DNS-Server kann man sogar auf zweierlei Weise veröffentlichen. Weil verschiedene Systeme hier unterschiedliche Vorlieben an den Tag legen, ist es von Vorteil, beide Verfahren (RDNSS via Router Advertisements sowie DHCPv6) zu aktivieren.

Test

Die gesamte IPv6-Konfiguration dieses Beispiels (`DNS_SUPPORT_IPV6='yes'` wird dabei angenommen!) lautet:

```
IPV6_NET_N='1'
IPV6_NET_1='0:0:0:456::1/64'   # IPv6-Adresse des Routers (ohne
                               # Subnetz-Präfix) + Subnetzmaske
IPV6_NET_1_DEV='eth0'          # Netzwerkschnittstelle für dieses IPv6-Subnetz
IPV6_NET_1_ADVERTISE='yes'     # /64-Subnetz-Präfix und Default-Route per RA
```

```
IPV6_NET_1_ADVERTISE_DNS='yes' # DNS-Server per RA
IPV6_NET_1_DHCP='yes'          # Domänen-Name und DNS-Server per DHCPv6

IPV6_TUNNEL_N='1'
IPV6_TUNNEL_1_PREFIX='2001:db8:123::/48' # /48-Subnetzmaske
IPV6_TUNNEL_1_LOCALV4='dynamic' # oder feste lokale IPv4-Adresse
IPV6_TUNNEL_1_REMOTEV4='212.224.0.188'
IPV6_TUNNEL_1_LOCALV6='2001:db8:900:551::2/64'
IPV6_TUNNEL_1_REMOTEV6='2001:db8:900:551::1'
IPV6_TUNNEL_1_TYPE='sixxs'
IPV6_TUNNEL_1_USERID='USER1-SIXXS'
IPV6_TUNNEL_1_PASSWORD='sixxs'
IPV6_TUNNEL_1_TUNNELID='T1234'
```

Ein normal konfigurierter Windows 7-Host sollte mit einem solch konfigurierten fli4l-Router automatisch seine IPv6-Adresse sowie Default-Route, DNS-Server und Domäne konfigurieren und somit den Rechner IPv6-tauglich machen. Das kann man z.B. mit einem enfachen PING vom Windows-Rechner ins IPv6-Internet realisieren. Im folgenden Beispiel versuchen wir, vom Windows-Host aus den fli4l.de-Webserver zu erreichen (wir benutzen dabei direkt die IPv6-Adresse, um nicht von der Korrektheit der DNS-Funktionalität auszugehen):

```
C:\>ping 2001:bf0:c000:a::2:132

Ping wird ausgeführt für 2001:bf0:c000:a::2:132 mit 32 Bytes Daten:

Antwort von 2001:bf0:c000:a::2:132: Zeit=104ms
Antwort von 2001:bf0:c000:a::2:132: Zeit=102ms
Antwort von 2001:bf0:c000:a::2:132: Zeit=106ms
Antwort von 2001:bf0:c000:a::2:132: Zeit=106ms

Ping-Statistik für 2001:bf0:c000:a::2:132:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0 (0% Verlust),
Ca. Zeitangaben in Millisek.:
    Minimum = 102ms, Maximum = 106ms, Mittelwert = 104ms
```

Schließlich kann man das Werkzeug “traceroute” (unter Windows: “tracert”) verwenden, um zu untersuchen, ob ein Paket korrekt geroutet wird. Ein Beispiel aus dem lokalen Netz des Autors ist untenstehend abgebildet. Daran kann man gut erkennen, dass ein Paket erst zum fli4l-Router (erste Zeile), dann zum anderen Ende des Tunnels (zweite Zeile) und schließlich in das weltweite IPv6-Internet (ab der dritten Zeile) gelangt:

```
C:\>tracert 2001:bf0:c000:a::2:132

Routenverfolgung zu virtualhost.in-berlin.de [2001:bf0:c000:a::2:132]
  über maximal 30 Abschnitte:

  1    <1 ms    <1 ms    <1 ms  garm.example.org [2001:db8:13da:1::1]
  2    70 ms    79 ms    71 ms  gw-1362.ham-01.de.sixxs.net [2001:db8:900:551::1]
  3    67 ms    71 ms    76 ms  2001:db8:800:1003::209:55
  4    68 ms     *       70 ms  2001:db8:1:0:87:86:71:240
  5    69 ms     *       71 ms  2001:db8:1:0:87:86:77:67
  6    72 ms     *       71 ms  2001:db8:1:0:86:87:77:81
```

```
 7    71 ms     *       71 ms  2001:db8:1:0:87:86:77:83
 8    90 ms     *       81 ms  2001:db8:1:0:87:86:77:62
 9    84 ms     *       88 ms  2001:db8:1:0:87:86:77:71
10    99 ms    83 ms    83 ms  2001:db8:1:0:87:86:77:249
11    94 ms    87 ms    87 ms  20gigabitethernet4-3.core1.fra1.he.net
 [2001:7f8::1b1b:0:1]
12    96 ms    99 ms    99 ms  10gigabitethernet1-4.core1.ams1.he.net
 [2001:470:0:47::1]
13   105 ms   108 ms   107 ms  2001:7f8:8:5:0:73e6:0:1
14   106 ms   107 ms   104 ms  virtualhost.in-berlin.de [2001:bf0:c000:a::2:132]

Ablaufverfolgung beendet.
```

Abbildungsverzeichnis

Tabellenverzeichnis

Index