IPv6 – zum Anfassen
Linux Workshop und CCP

Johannes Hubertz

hubertz-it-consulting GmbH

Universität zu Köln, 15. 6. 2010

IPv6 – zum Anfassen: Inhalt

Vorstellung: Johannes Hubertz

1980 Hardware-Reparatur: Ersatzteile auf Bauteilebene

1984 Entwicklung Sonderprodukte, Assembler, PLM

1987 Erstkontakt mit Unix (SCO-Xenix) und C

1994 Erstkontakt mit IP

1996 Xlink, root@www.bundestag.de, . . .

1997 SSLeay, ipfwadm mit shell-scripts

1998 „Ins Allerheiligste“, iX 1/1998, Heise Verlag

1999 IT-Security Mgr. Bull D-A-CH

2001 Gibraltar, FreeSwan, iptables . . .

2001 Firmenteilung → Entwicklung und Betrieb sspe

2002 Weiterentwicklung und Betrieb von sspe

2005 Gründung der hubertz-it-consulting GmbH

ab 1973 Bundesanstalt Technisches Hilfswerk in Köln-Porz, . . . ZFü, S2

ab 2001 Segeln, am liebsten auf Salzwasser, SKS

Vorstellung: hubertz-it-consulting GmbH

Erkenntnisse aus dem Berufsleben

Bellovin and Cheswick: Firewalls and Internet Security, 1994

Fazit: Keep it simple!

Oder mit Einstein: So einfach wie möglich, aber nicht einfacher!

Etwas Erfahrung war Voraussetzung

Gründung am 8. August 2005, Sitz in Köln

Geschäftsinhalt: Dienstleistungen im Umfeld der IT-Sicherheit

Logo: Johannes Hubertz Certificate Authority als ASCII-7Bitmuster

Diese paar Bits findet sich in einigen 10000 X.509 Anwenderzertifikaten in der Seriennummer wieder

Wir sind käuflich ;-)

IPv6 – nur Marketing?

Haben Sie eine Idee,

wie **Ihr Mobiltelefon**

mit **Ihrem PC**

via bluetooth . . .

???

ewige Wahrheiten . . .

**Ich denke nie an die Zukunft,
sie kommt früh genug.**

Albert Einstein

Bald ist das Internet alle . . .

RIPE Community resolution: (27.October 2007)

„Growth and innovation on the Internet depends on the continued availability of IP address space.

The remaining pool of unallocated IPv4 address space is likely to be fully allocated within two to four years. IPv6 provides the necessary address space for future growth.

We therefore need to facilitate the wider deployment of IPv6 addresses.“

RIPE ⇔ Réseaux IP Européens

Es wird ernst . . .

RIPE Community statement: (7. May 2010)

„The RIPE community supports all efforts to assist in the deployment of IPv6, especially in developing countries.

However, we note concerns being expressed within the ITU by a few members, most recently in the ITU IPv6 Group, that the current address management system is inadequate.

The RIPE community mandates the RIPE NCC to work with the ITU IPv6 Group, individual ITU members, and the community to clearly identify these concerns and to find ways to address them within the current IP address management system.“

ITU ⇔ International Telecommunication Union

IPv6 – Fiktion, Hype, Marketing, oder was?

IPv6 ist schon betagt:

Start in der Mitte der 90er Jahre

IPv6 ist reine Technik:

weit mehr als 200 RFCs spezifizieren IPv6

IPv6 ist notwendig:

IPv4 wird knapp

IPv6 existiert:

IPv6 ist fertig, funktioniert schon bestens und bald omnipräsent

ewige Wahrheiten . . .

**Menschen mit einer neuen Idee gelten solange als Spinner,
bis sich die Sache durchgesetzt hat.**

Mark Twain

Mythen und Märchen

Die schlechte Nachricht

IPv6 ist genauso unsicher wie IPv4

really!

Die gute Nachricht

IPv6 ist genauso sicher wie IPv4

really!

ICMPv6 ist böse

ICMPv6 ist essentieller Bestandteil von IPv6.

ICMPv6 komplett zu filtern (wie es bei IPv4 oft üblich war), ist gleichbedeutend mit IPv6 ausschalten. Es funktioniert nicht.

IPsec ist schon drin!

IPsec wurde zuerst auf IPv6 spezifiziert

Jede IPv6-Implementierung sollte auch IPsec realisieren. Jedoch:
Zur Nutzung braucht man Schüsselverteilung,
z. B. IKEv1 oder IKEv2 oder proprietär, Preshared Keys oder X.509-Zertifikate
KnowHow-trächtige Konfigurationen sind gefragt,
Kompatibilität unter Herstellern,
. . . nix ändert sich, was reg ich mich auf . . . ;-)

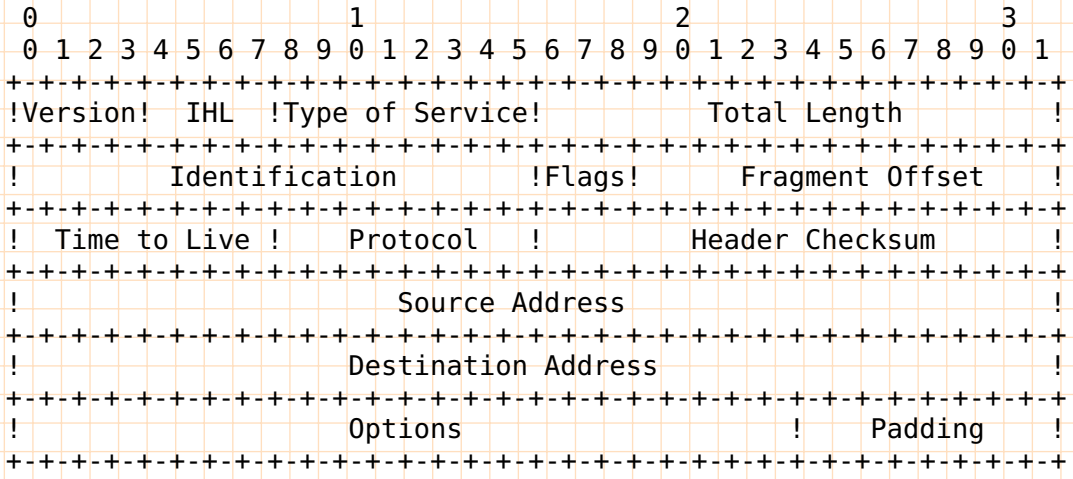
Wir brauchen nur eine
PKI!
Dann ist die Welt ja in Ordnung.

IPv4 – IPv6

unvergleichbar, oder?

IP – header as defined in RFC 791 *as of September* 1981

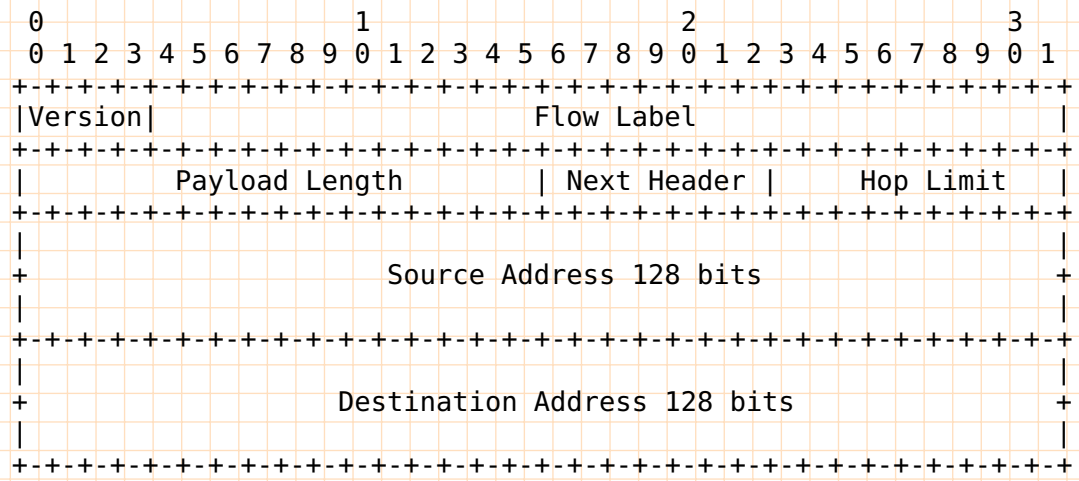
ASCII – ART



IPv6 – header as defined in RFC 2460

as of December 1998
obsoletes RFC 1883 (December 1995)

ASCII – ART



IPv6 – header explanations

Version	4 Bit	IP Version (==6)
Flowlabel	28 Bit	Zusatzinformationen für Router, z.B. für QOS
Payload Length	16 Bit	Länge des Paketes nach dem Header
Next Header	8 Bit	Welcher Header kommt danach?
Hop Limit	8 Bit	vgl. TTL bei IPv4
Source Address	128 Bit	
Destination Address	128 Bit	

IPv6 – address explained . . .

128 Bit are combined from:

bits number	**value** example	**meaning**
3	 001_{bin} 111_{bin}	prefix global allocatable 2000::/3 multicasts et.al.

45	
	`2001:db8::/32`
	`2001:db8:beef::/48`
		global routing prefix
		documentation
		RIPE, ISP, customer friendly user test

16	 0001_{hex} . . . $ffff_{hex}$	subnet ID second usable subnet /64 another $2^{16} - 3$ of /64 last usable /64
64	 `216:d3ff:fea4:5174`	Interface ID a LapTop's ethernet interface ID

Es gibt keine Broadcasts mehr!

Adressen – Kleinigkeiten!

IPv6 – Adresseigenschaften

IPv6 Adressen sind 128 bit lang, d.h. $IP \in \{1..2^{128}\}$

$2^{128} = 340.282.366.920.938.463.463.374.607.431.768.211.456_{dez}$

Das entspricht 665 Milliarden Adressen pro mm^2 Erdoberfläche

Hexadezimale Schreibweise, je 2 Bytes durch einen ':' getrennt.

Beispiel: `2001:db8:beef:4711::1` '::' ⇔ fehlende 0-Bytes

'::' Nur einmal in einer Adresse!

`2001:db8:beef::/48` ⇔ 65.536 /64 Netze (prefix wie bei IPv4)

IPv6 – Scope (Reichweite)

link-local

Jedes Interface hat `1..n` link-local Adresse(n) aus `fe80::/10`

site-local deprecated!

Ein Interface hat `0..n` site-local Adressen aus `fc00::/8` oder `fd00::/8`

global

Ein Interface hat `0..n` globale Adressen aus `2000::/3`

multicast

Ein Interface hat `0..n` Multicast Adressen aus `ff00::/8`

IPv6 – Besondere Adressen

::	nicht spezifizierte Adresse ⇔`0:0:0:0:0:0:0:0`
::1	loopback
`fe80::/10`	link-local
`ff00::/8`	multicast
`ff01::1`	multicast, all hosts
`ff01::2`	multicast, all routers
`fc00::/8`	Unique Local Adressen (zentral verwaltet)
`fd00::/8`	Unique Local Adressen
`2000::/3`	globale Unicast Adressen
`2001:db8::/32`	Prefix für Dokumentation

next header

Beliebig viele Header pro IP-Paket

IPv6 header: next header I

IPv6 header: next header field, 8 bits

- ähnlich dem Protokol-Feld im IPv4-Header, aber universeller
- Werte gleich, siehe auch `/etc/protocols`
- beliebig verkettbar
- RH Typ 0 und 2 bergen einige Sicherheitsrisiken
- RH Typ 0 ist „deprecated“ ⇒, ist aber schon implementiert!(RFC 5095)
- RH Typ 2 ist essentieller Bestandteil von Mobile IPv6

Lösungsansätze gibt es, aber

- entweder kompliziert, z.B. im Firewalling
- oder noch nicht allgemein implementiert (RFC 5095)

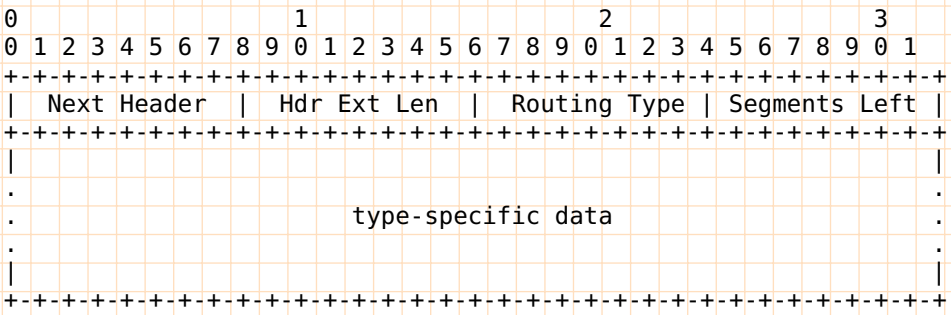
IPv6 header: next header II

Extension Header in der Basis-Spezifikation

value	meaning
0	hop by hop options header
43	routing header
44	fragmentation header
50	encapsulation payload header (RFC 2406)
51	authentication header (RFC 2402)
60	destination options header

IPv6 header: next header III

routing header *müssen* von jedem Host verarbeitet werden



routing header type 0

Enspricht dem (strict/loose) sourcerouting, für debug, ISP-spez.Routing, . . .

routing header type 2

Wird zwingend für Mobile-IPv6 benötigt

IPv6 header: next header IV

Verkettungen sind in beliebiger Länge möglich:

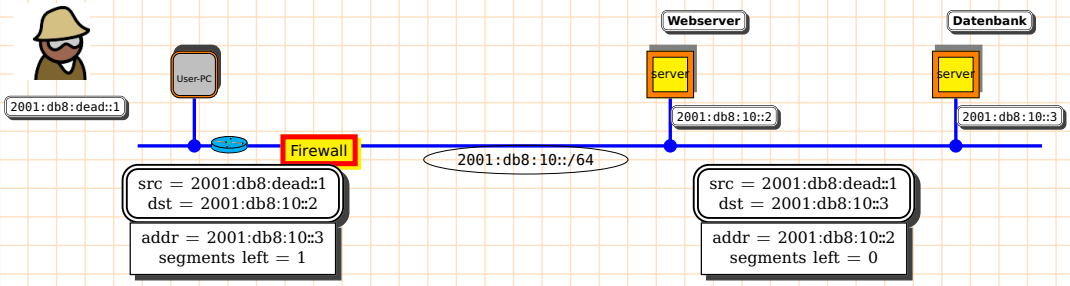
IPv6 header Next header = TCP value 6	TCP header and data		
IPv6 header Next header = Routing Value 43	Routing header Next header = TCP Value = 6	TCP header and data	
IPv6 header Next header = Routing Value 43	Routing header Next header = Fragment Value = 44	Fragment header Next header = TCP Value 6	TCP header and data

IPv6 header: next header V

Reihenfolge der extension header:

1	IPv6 Header
2	Hop-by-Hop Options Header
3	Destination Options Header für Router auf dem Pfad
4	Routing Header
5	Fragment Header
6	Authentication Header
7	Encapsulation Security Payload Header
8	Destination Options Header für den endgültigen Empfänger
9	Upper-Layer Header

IPv6 header: next header – malicious usage



Angreifer schickt ein Paket an die erreichbare Adresse ::2, dieser Host leitet es weiter an ::3, welcher durch Firewall strikt gefiltert wird.

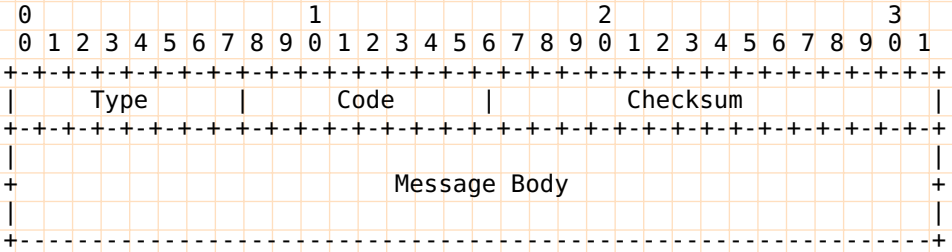
ICMPv6

internet control message protocol version 6

ICMPv6 – as defined in RFC 2463

ICMPv6

1.) ersetzt ARP vollständig: **neighbor discovery (ND)**
2.) kann automatisch Routen: **router discovery (RD)**
3.) erkennt doppelte Adressen: **duplicate address detection (DAD)**
4.) nutzt multicasts: **ff00::0 ip6-mcastprefix**



ICMPv6 ist essentieller Bestandteil der Ende-zu-Ende Kommunikation!
Daher: Filterung von ICMPv6 nur auf spezielle icmp-types möglich

ICMPv6 – as defined in RFC 2463

type	meaning
1	Destination Unreachable
2	Packet Too Big
3	Time Exceeded
4	Parameter Problem
128	Echo Request
129	Echo Reply
130	Group Membership Query
131	Group Membership Report
132	Group Membership Reduction
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect
138	Router Renumbering

ND – neighbor detection

neigbour detection

ND ersetzt arp (address resolution protocol) RFC 2461, December 1998:
„Neighbor Solicitation: Sent by a node to determine the link-layer address of a neighbor, or to verify that a neighbor is still reachable via a cached link-layer address. Neighbor Solicitations are also used for Duplicate Address Detection.
Neighbor Advertisement: A response to a Neighbor Solicitation message. A node may also send unsolicited Neighbor Advertisements to announce a link-layer address change.

1. prefix ff02::1:ff00:0/104 und rechte 3 Oktetts der Ziel-IP → Multicast-Addresse
2. icmpv6 type 135 an diese Adresse
3. Zielhost antwortet mit icmpv6 type 136 oder timeout

DAD – duplicated address detection

duplicate address detection

DAD geschieht vor der Zuweisung der eigenen IPv6-Adresse, RFC 2462, December 1998: „Duplicate Address Detection is performed on unicast addresses prior to assigning them to an interface whose DupAddrDetectTransmits variable is greater than zero. Duplicate Address Detection MUST take place on all unicast addresses, regardless of whether they are obtained through stateful, stateless or manual configuration, with the exception of the following . . . “

1. Unicast, ICMP Typ 135, Absender ‘::‘ an die Zieladresse
2. Falls vorhanden, erfolgt eine Antwort an ff02::1
3. Falls nicht vorhanden, wird die Adresse gewählt

IPv6 – autoconfiguration

automagically configuration of workstations. . .

IPv6 – fast alles kann automatisch geschehen . . .

1. Interface-ID (MAC-Adresse) bestimmt link-local-Adresse → `fe00::/64`
2. DAD = duplicate address detection
3. RD = router detection
4. aktiver Host am LAN, erkennt Routing-Änderungen mit RD

Linux

interfaces – configuration

IPv6 Interface: Linux configuration

static:

```
tut:~# more /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback
#
auto eth2
iface eth2 inet6 static
        pre-up /sbin/modprobe ipv6
        address 2001:db8:beef:fb00::1
        netmask 64
        post-up /sbin/ip route add 2001:db8:beef:fb00::/56 via 2001:db8:beef:fb00::2
        post-up echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
```

autoconfiguration:

```
tut:~# more /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
allow-hotplug eth0
iface eth0 inet6 manual
        pre-up /sbin/ip link set dev eth0 up
        post-down /sbin/ip link set dev eth0 down
```


IPv6 Linux commands

`ip -6 address add nnn dev eth0 scope global`	IP-Adresse hinzufuegen
`ip -6 address del nnn dev eth0`	IP-Adresse wegnehmen
`ip -6 neigh show`	Nachbarschaft anzeigen
`ip -6 route show`	IPv6 Routen auflisten
`ip -6 route add target via nexthop dev eth0`	IPv6 Route hinzufuegen
`ip -6 route del target via nexthop dev eth0`	IPv6 Route wegnehmen

IPv6 Interface: OpenBSD configuration

static:

```
# cat hostname.sis0
inet 192.168.110.177 255.255.255.0 NONE
inet6 2001:db8:beef:2::10/64
#
# grep rtadvd rc.conf
rtadvd_flags=NO         # for normal use: list of interfaces
#
```

autoconfiguration:

```
# grep ip6 sysctl.conf
net.inet6.ip6.forwarding=0      # 1=Permit forwarding (routing) of IPv6 packets
#net.inet6.ip6.mforwarding=1    # 1=Permit forwarding (routing) of IPv6 multicast packets
#net.inet6.ip6.multipath=1      # 1=Enable IPv6 multipath routing
net.inet6.ip6.accept_rtadv=1    # 1=Permit IPv6 autoconf (forwarding must be 0)
#
```


IPv6 OpenBSD commands

`ifconfig`	Alle Interfaces anzeigen [Mix aus L1,L2,L3]
`ndp -an`	Nachbarschaft anzeigen
`route -n show`	Routingtabelle anzeigen

IPv6 Interface: M$ XP configuration

static:

AFAIK: NA

autoconfiguration:

```
cmd.exe
ipv6.exe install
```

reboot (as usual) oder: Zum Beenden drücken Sie **Start!**

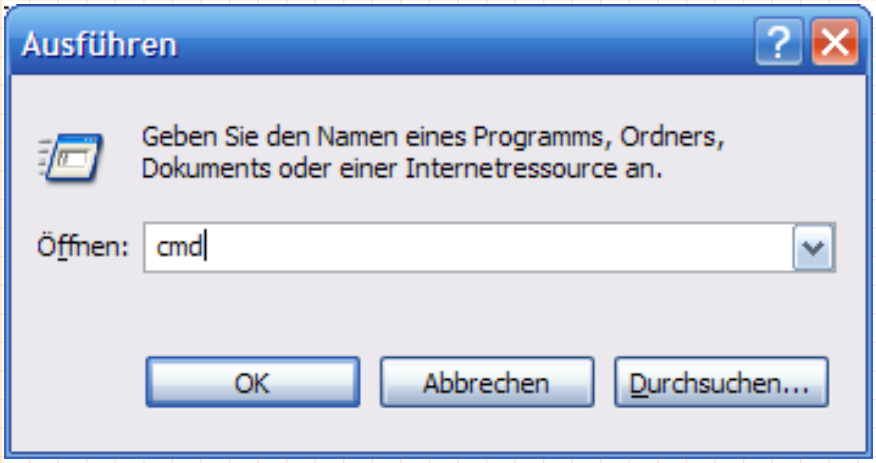
IPv6 M$ XP commands

`ipconfig /all`	**IP Konfiguration anzeigen**
`route print`	**Routingtabelle anzeigen**

Die folgenden Bilder sagen mehr als viele Worte!

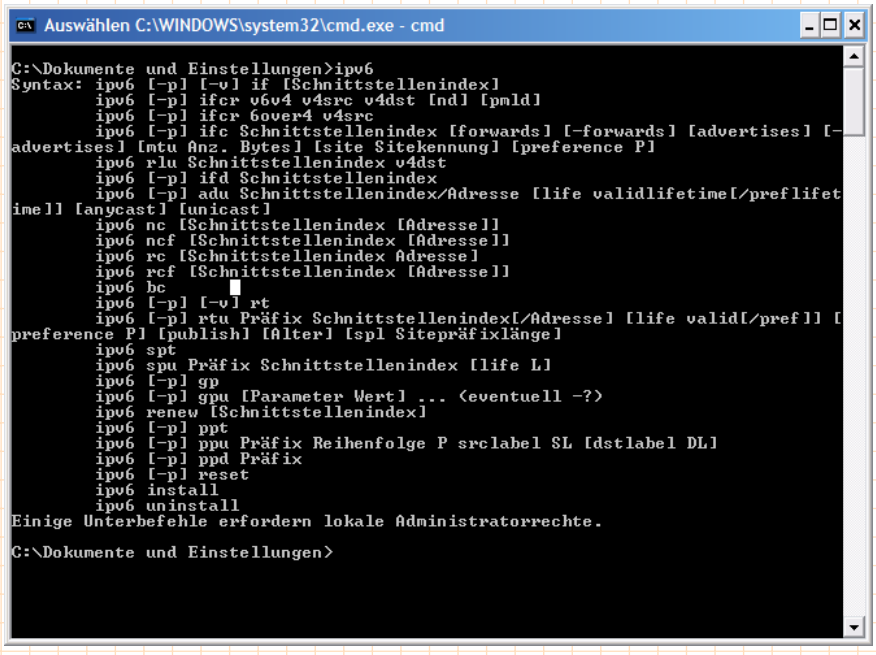
Da geht nix automatisch!

Zum Beenden drücken Sie erst auf Start[1], dann auf ausführen,
„cmd“ eingeben, dann OK klicken!



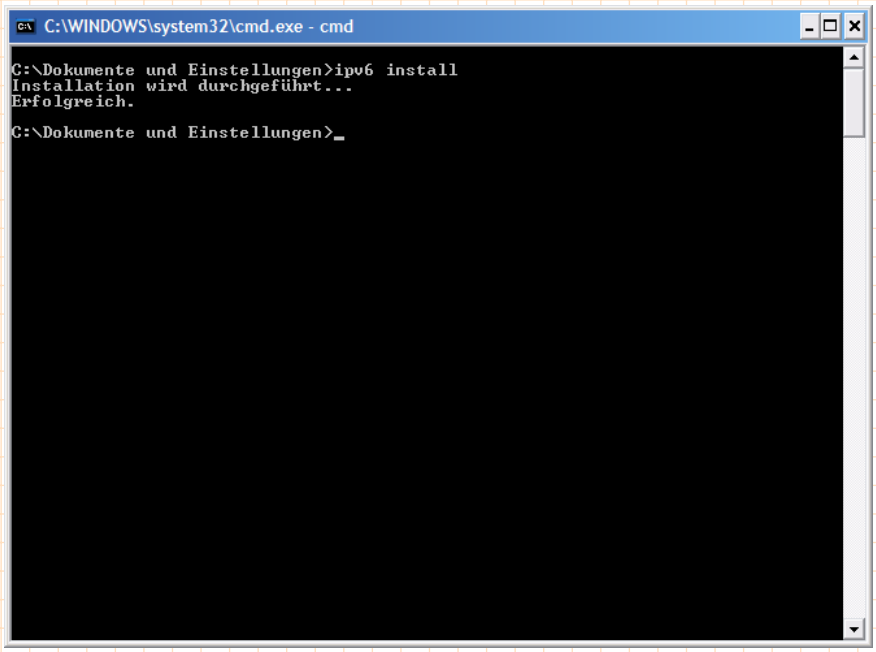
[1] Links unten, oder woanders

Da geht immer noch nix automatisch!



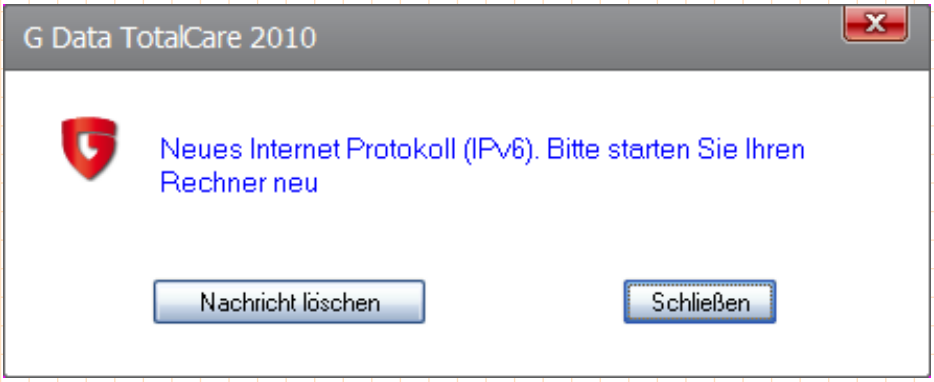
Hier ist nix zum klicken, erst mal nur ansehen!
Alles klar?

Und immer noch nix automatisch!



Gefühlte Wartezeit: 1 min,
ansonsten bleibt fast alles unsichtbar. Hm.

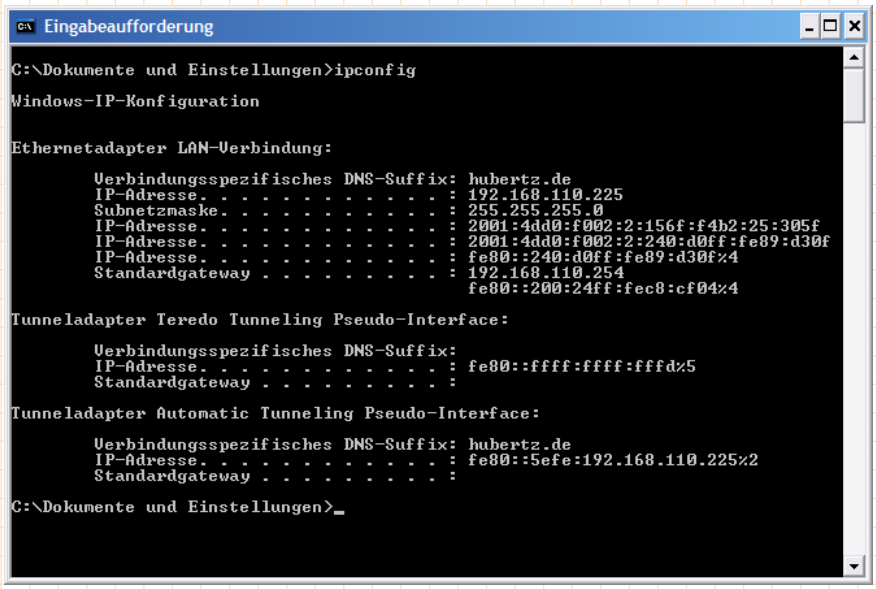
Wer behindert mich hier?



Oh, welche Nachricht darf / kann / soll / muß ich löschen?

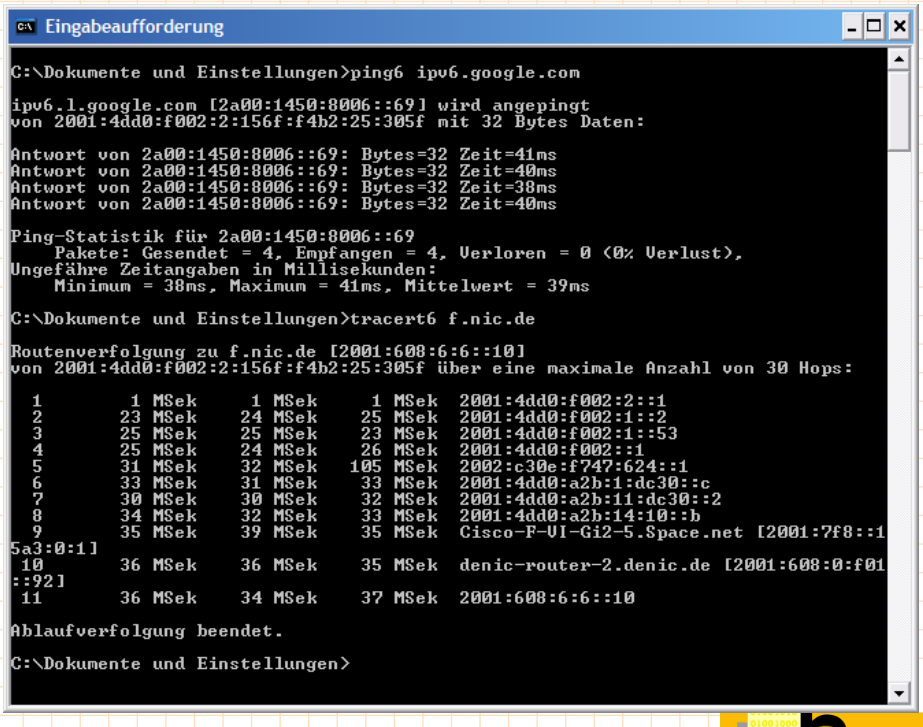
Nur ein Boot später . . .

. . . start, ausführen, cmd, Sie wissen schon . . .



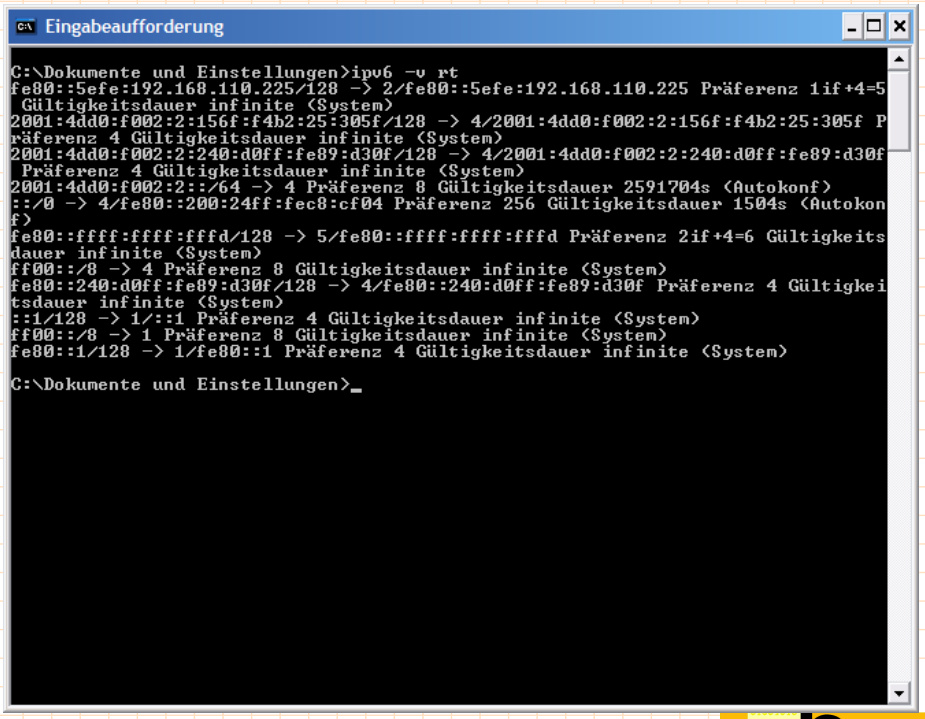
Das freut ET: Nach Hause telefonieren ist auch dabei!
Teredo sei Dank!

Nur ein Boot später . . .



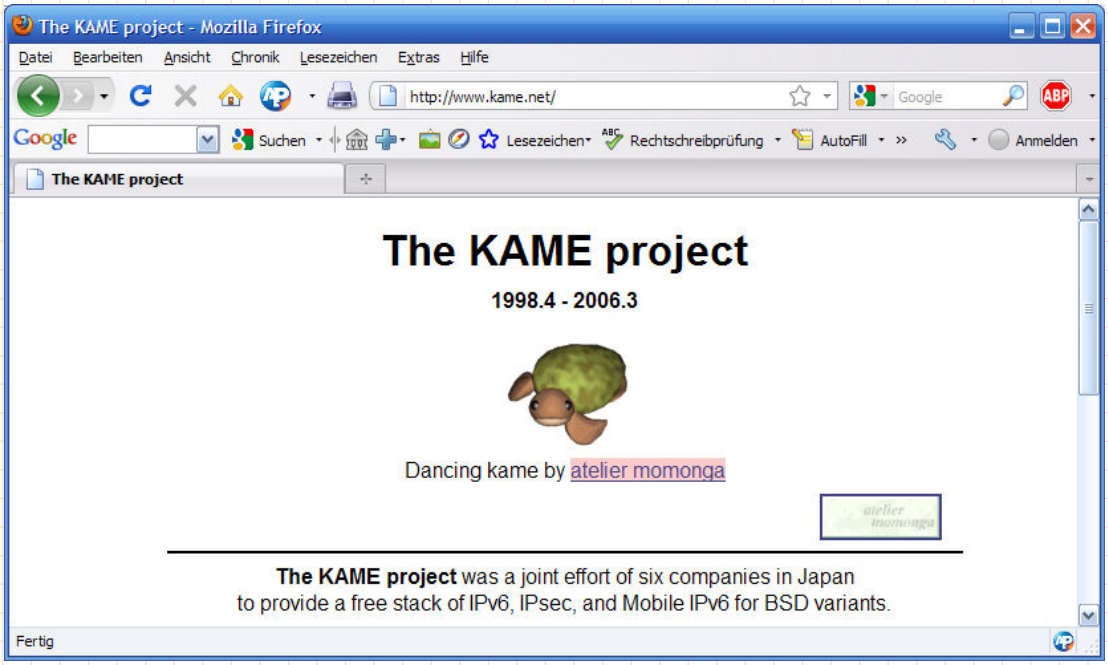
Da geht schon mal was. Aber: MSek oder ms oder wie?

Connectivity . . .



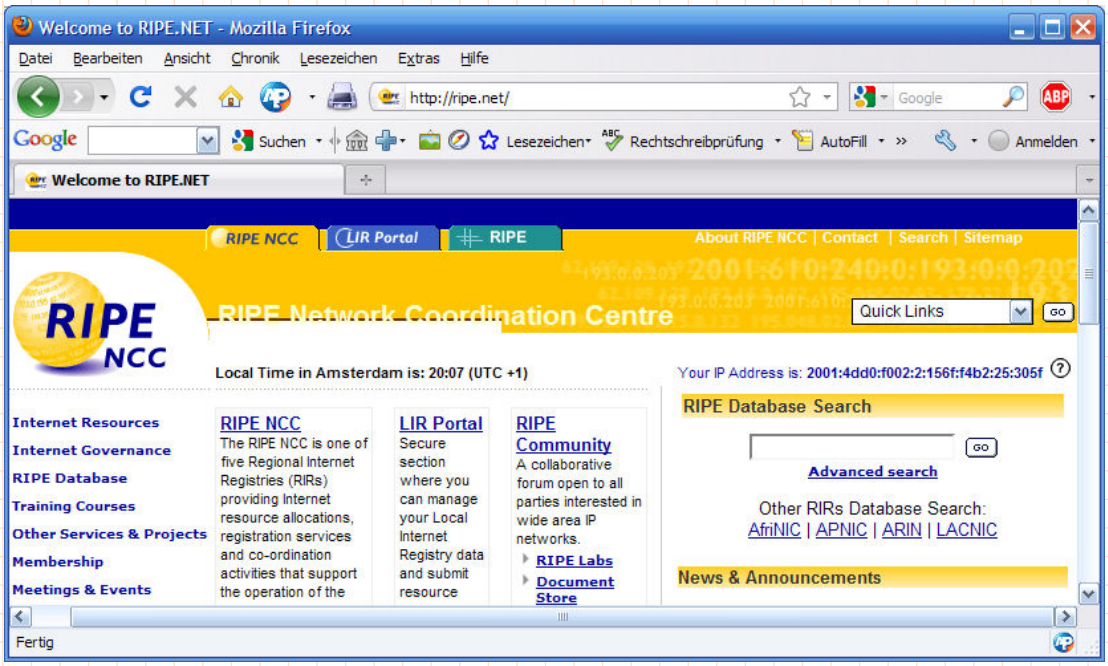
Bisserl unleserlich, aber vermutl. korrekt!

Connectivity . . .



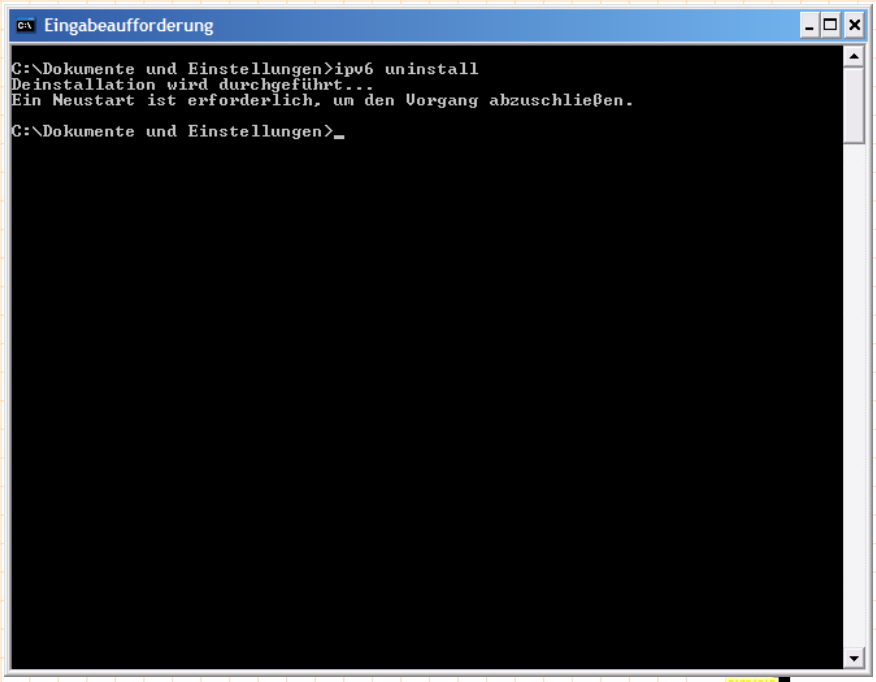
Die rennt aber schnell für eine Schildkröte (IPv6)!

Connectivity . . .



Wer bin ich?

. . . jetzt aber genug!



Gefühlte Wartezeit: Wieder ca. 1 min,
hier bleibt alles unsichtbar. Hm.

Client Autoconfiguration . . .

Server and clients view

IPv6 clients autoconfiguration (magic)

— Server —

radvd server-config-file[2] follows:

```
interface eth2 {
   AdvSendAdvert on;
   prefix 2001:db8:beef:fc00::/64    { };
};
```

rtadvd server-config-file[3] follows:

```
# cat /etc/rtadvd.conf
sis1: \
        prefix: 2001:db8:beef::/48
        prefix: 2001:db8:beef:0020::/64
#
```

Remark: radvd – router advertisement daemon

[2] Debian GNU/Linux: /etc/radvd.conf

[3] OpenBSD: /etc/rtadvd.conf

IPv6 clients autoconfiguration (magic)

— Server —

dibbler server-config-file[4] follows:

```
log-level 8
log-mode short
preference 0
iface "eth2" {
 // also ranges can be defines, instead of exact values
 t1 1800-2000
 t2 2700-3000
 prefered-lifetime 3600
 valid-lifetime 7200
 class {
    pool 2001:db8:beef:fc00:0100:0000::/96
 }
 ta-class {
    pool 2001:db8:beef:fc00:0200:0000::/96
 }
 pd-class {
     pd-pool 2001:db8:beef:fc00:0300::/80
     pd-length 96
 }
 option dns-server 2001:db8:beef:1::53
 option domain hubertz.de
 option vendor-spec 5678-0x0002aaaa
 option ntp-server 2001:db8:beef:1::1
 option time-zone  CET
}
```

Remark: dibbler-server is a portable implementation of the DHCPv6 server.

[4] Debian GNU/Linux: /etc/dibbler/server.conf

IPv6 client autoconfiguration (magic only)

— Client —

dibbler client-config-file[5] follows:

```
# 8 (Debug) is most verbose. 7 (Info) is usually the best option
log-level 7
# To perform stateless (i.e. options only) configuration, uncomment
# this line below and remove any "ia" keywords from interface definitions
#
# stateless
#
iface eth0 {
# ask for address
    ia
# ask for options
   option dns-server
   option domain
#   option ntp-server
#   option time-zone
#   option sip-server
#   option sip-domain
#   option nis-server
#   option nis-domain
#   option nis+-server
#   option nis+-domain
}
```

Remark: dibbler-client is a portable implementation of the DHCPv6 client.

[5] Debian GNU/Linux: /etc/dibbler/client.conf

Alle Clients sind konfiguriert,

Was nun?

Wir haben eine Firewall!

Da ist alles sicher.

?

NetFilter

ip6tables, pf – und wie?

NetFilter – linux 2.4, 2.6, . . .

altbewährte Technik mit 3 Standard-chains

- INPUT
- OUTPUT
- FORWARD
- userdefined chains

Ein Kommando für alles

man ip6tables

```
/sbin/ip6tables --help
```

/sbin/ip6tables – linux 2.4, 2.6, . . .

```
#
# /sbin/ip6tables --help

Usage: ip6tables -[AD] chain rule-specification [options]
       ip6tables -I chain [rulenum] rule-specification [options]
       ip6tables -R chain rulenum rule-specification [options]
       ip6tables -D chain rulenum [options]
       ip6tables -[LS] [chain [rulenum]] [options]
       ip6tables -[FZ] [chain] [options]
       ip6tables -[NX] chain
       ip6tables -E old-chain-name new-chain-name
       ip6tables -P chain target [options]
       ip6tables -h (print this help information)

. . .
```

Die man-page liest sich noch mühsamer!

Der Mensch ist faul und infolgedessen auch erfindungsreich . . .

Paketfilter I: Hier ist Handarbeit gefragt

Definitionen: Wer ist hier beteiligt?

#**Name**	**Adresse**	**Kommentar**
any	::/0	# Alle Welt
many	2000::/3	# erreichbare Welt
localhost	::1/128	#
srv	2001:db8:beef:2::10/128	# service
ns	2001:db8:beef:1::53/128	# 1.nameserver
ns	2001:db8:beef:3::23/128	# 2.nameserver
nc-dns	2001:4dd9:abcd:0:0:2:b351:f602/128	# NetCologne DNS
admin	2001:db8:beef:3:a00:27ff:fe67:4649/128	# administration
#admin	fe80::a00:27ff:fe67:4649/128	# administration

Regeln: Wer darf mit wem was?

#**Source**	**Destination**	**protocol**	**port**	**action**	**Options**	**#Kommentar**
admin	ns	tcp	22	accept	NOIF	
many	ns	udp	53	accept		
any	any	ip6	all	drop		# no def. log

Paketfilter II: Handarbeit nicht mehr so sehr gefragt

Generierung von Paket-Filtern für Linux und OpenBSD

Output Stufe I Linux/OpenBSD

```
!mach!rn!me_s!me_d!src!src_if!dst!dst_if!prot!port!acti!varying options!
!ns!1!0!0!2001:db8:beef:3:a00:27ff:fe67:4649/128!!2001:db8:beef:1::53/128!eth1!tcp!22!accept!NOIF!
!ns!1!0!1!2001:db8:beef:3:a00:27ff:fe67:4649/128!!2001:db8:beef:3::23/128!eth1!tcp!22!accept!NOIF!
!ns!2!0!0!2000::/3!!2001:db8:beef:1::53/128!eth1!udp!53!accept!!
!ns!2!0!1!2000::/3!!2001:db8:beef:3::23/128!eth1!udp!53!accept!!
!ns!3!0!0!::/0!!::/0!!ip6!all!drop!!
```

Output Stufe II Linux

```
export IPI="/sbin/ip6tables -A INPUT "
export IPO="/sbin/ip6tables -A OUTPUT "
export IPF="/sbin/ip6tables -A FORWARD "
$IPI -m rt -rt-type 0 -j DROP
$IPF -m rt -rt-type 0 -j DROP
echo -e “Rule 1 \r“>2
$IPF -s 2001:db8:beef:3:a00:27ff:fe67:4649/128 -d 2001:db8:beef:1::53/128 -p tcp -dport 22 -j ACCEPT
$IPF -d 2001:db8:beef:3:a00:27ff:fe67:4649/128 -s 2001:db8:beef:1::53/128 -p tcp -sport 22 -j ACCEPT
$IPI -s 2001:db8:beef:3:a00:27ff:fe67:4649/128 -d 2001:db8:beef:3::23/128 -p tcp -dport 22 -j ACCEPT
$IPO -d 2001:db8:beef:3:a00:27ff:fe67:4649/128 -s 2001:db8:beef:3::23/128 -p tcp -sport 22 -j ACCEPT
echo -e “Rule 2 \r“>2
...
/bin/ip6tables -P INPUT DROP
/bin/ip6tables -P OUTPUT DROP
/bin/ip6tables -P FORWARD DROP
```


IPv6 – transportation over IPv4

IPv6 transportation

6to4 tunnel

Linux: IPv6 transportation via 6to4 tunnel

taken from /etc/network/interfaces:

```
auto sit1
iface sit1 inet manual
        pre-up ip tunnel add sit1 mode sit local 192.168.111.115 remote 192.168.110.176 ttl 64
        pre-up ip link   set sit1 up
        pre-up ip addr   add 2001:db8:beef:3::1/64 dev sit1
        pre-up ip route  add 2001:db8:beef:2::/64  dev sit1
        down   ip link   set sit1 down
```

taken from the commandline:

```
r-ex:/etc/network# ip link show dev sit1
16: sit1@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1480 qdisc noqueue state UNKNOWN
    link/sit 192.168.111.115 peer 192.168.110.176

r-ex:/etc/network# ip tunnel show sit1
sit1: ipv6/ip  remote 192.168.110.176  local 192.168.111.115  ttl 64

r-ex:/etc/network# ifconfig sit1
sit1      Link encap:IPv6-in-IPv4
          inet6 addr: 2001:db8:beef:3::1/64 Scope:Global
          inet6 addr: fe80::574f:173/128 Scope:Link
          UP POINTOPOINT RUNNING NOARP  MTU:1480  Metric:1
          RX packets:32024 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24048 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2783336 (2.6 MiB)  TX bytes:6493188 (6.1 MiB)
```

OpenBSD: IPv6 transportation via 6to4 tunnel

taken from /etc/rc.local:

```
/sbin/ifconfig gif0 inet6 2001:db8:beef:3::2 2001:db8:beef:3::1 prefixlen 128 alias
/sbin/ifconfig gif0 giftunnel 192.168.110.176 192.168.111.115
/sbin/route    add -inet6 2000:: -prefixlen 3 2001:db8:beef:3::1
```

taken from the commandline:

```
obi-wan # ifconfig sis0
sis0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        lladdr 00:00:24:c8:cf:04
        priority: 0
        groups: egress
        media: Ethernet autoselect (100baseTX full-duplex)
        status: active
        inet 192.168.110.176 netmask 0xffffff00 broadcast 192.168.110.255
        inet6 fe80::200:24ff:fec8:cf04%sis0 prefixlen 64 scopeid 0x1
        inet6 2001:db8:beef:2::1 prefixlen 64
obi-wan #
obi-wan # ifconfig gif0
gif0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1280
        priority: 0
        groups: gif
        physical address inet 192.168.110.176 --> 192.168.111.115
        inet6 fe80::200:24ff:fec8:cf04%gif0 ->  prefixlen 64 scopeid 0x6
        inet6 2001:db8:beef:3::2 -> 2001:db8:beef:3::1 prefixlen 128
```


IPv6 transportation

OpenVPN encrypted tunnel

using X.509 certificates for mutual authentication

Linux: IPv6 transportation via IPv4

OpenVPN is our friend, server-config-file follows:

```
daemon dns
local 192.0.2.126 # Test-Net of RFC 3330 - Special-Use IPv4-Addresses
port 4712
dev tun
tun-ipv6
proto tcp-server
ifconfig-noexec
redirect-gateway
up          /etc/openvpn/up-c
down        /etc/openvpn/down-c
log-append /var/log/ovpn-c.log
writepid    /var/run/ovpn-c.pid
dh          /etc/openvpn/dh2048.pem
ca          /etc/openvpn/cacert.pem
cert        /etc/openvpn/r-exCert.pem
key         /etc/openvpn/r-exReq.pem
crl-verify /etc/openvpn/crl.pem
tls-server
tls-verify /etc/openvpn/tls-verify
tls-exit
persist-tun
ping 15
ping-restart 45
ping-timer-rem
persist-key
verb 3
```

Remark: one server-instance for each possible client connection

Linux: IPv6 transportation via IPv4

OpenVPN is our friend, server-config-script up-c follows:

```
#!/bin/bash
#
INTERFACE=$1; shift;
TUN_MTU=$1;   shift;
UDP_MTU=$1;   shift;
LOCAL_IP=$1;  shift;
REMOTE_IP=$1; shift;
MODUS=$1;     shift;
#
ip link set ${INTERFACE} up
ip link set mtu ${TUN_MTU} dev ${INTERFACE}
#
for myad in fc00
do
        ip -6 addr  add fe80:0:${myad}::1/64             dev ${INTERFACE}
        ip -6 route add 2001:db8:beef:${myad}::/56  via fe80:0:${myad}::2  dev ${INTERFACE}
done
exit 0
```

Remark: one server-instance for each possible client connection

Linux/OpenBSD: IPv6 transportation via IPv4

OpenVPN is our friend, serverside tls-verify-script follows:

```
#!/usr/bin/perl -w
#
($depth, $cn) = @ARGV;
if ($depth == 1 && $cn eq '/C=DE/ST=Germany/L=Cologne/O=hubertz-it-consulting_GmbH/OU=consulting/CN=
                                hubertz_consulting_CA/emailAddress=ca@hubertz.de') {
        exit 0;
}
exit 1 if (!($depth == 0));
exit 0 if ($cn eq '/C=DE/ST=Germany/O=hubertz-it-consulting_GmbH/OU=IPv6-Tunnel/CN=fa00');
exit 0 if ($cn eq '/C=DE/ST=Germany/O=hubertz-it-consulting_GmbH/OU=IPv6-Tunnel/CN=fb00');
exit 0 if ($cn eq '/C=DE/ST=Germany/O=hubertz-it-consulting_GmbH/OU=IPv6-Tunnel/CN=fc00');
exit 0 if ($cn eq '/C=DE/ST=Germany/O=hubertz-it-consulting_GmbH/OU=IPv6-Tunnel/CN=fd00');
exit 0 if ($cn eq '/C=DE/ST=Germany/O=hubertz-it-consulting_GmbH/OU=IPv6-Tunnel/CN=fe00');
exit 0 if ($cn eq '/C=DE/ST=Germany/O=hubertz-it-consulting_GmbH/OU=IPv6-Tunnel/CN=ff00');
exit 1;
```

Remark: one server-instance for each possible client connection

Linux/OpenBSD: IPv6 transportation via IPv4

OpenVPN is our friend, client-config follows:

```
dev tun
tun-ipv6
remote 192.0.2.126 # Test-Net of RFC 3330 - Special-Use IPv4-Addresses
port 4712
proto tcp-client
script-security 2
up   /etc/openvpn/client-up-script
down /etc/openvpn/client-down-script
ca /etc/openvpn/cacert.pem
cert /etc/openvpn/cert.pem
key /etc/openvpn/key.pem
tls-client
tls-verify /etc/openvpn/tls-verify
log-append /dev/shm/openvpn.log
persist-tun
ping 15
ping-restart 45
ping-timer-rem
persist-key
verb 3
```

Linux: IPv6 transportation via IPv4

OpenVPN is our friend, client-up-script follows:

```
1  #!/bin/bash
2
3  INTERFACE=$1; shift;
4  TUN_MTU=$1;   shift;
5  UDP_MTU=$1;   shift;
6  LOCAL_IP=$1;  shift;
7  #REMOTE_IP=$1; shift;
8  #MODUS=$1;     shift;
9
10 /usr/sbin/openvpn --mktun --dev ${INTERFACE}
11 /sbin/ip link  set ${INTERFACE} up
12 /sbin/ip link  set mtu ${TUN_MTU} dev ${INTERFACE}
13 /sbin/ip addr  add fe80:0:fc00::2/64         dev ${INTERFACE}
14 /sbin/ip route add 2000::/3 via fe80:0:fc00::1 dev ${INTERFACE}
15 /sbin/ip addr  add 2001:db8:beef:fc00:4711::1/128 dev eth2
16 /sbin/ip addr  add 2001:db8:beef:fc00::1/64 dev eth2
17 /sbin/ip route add 2001:db8:beef:fc00::/56 via 2001:db8:beef:fc00::2
18 exit 0
```

Remark: Line 15 assures to have least one global IP independant from status of eth2, needed for keepalive

OpenBSD: IPv6 transportation via IPv4

OpenBSD and OpenVPN are our friends

configuration is similar except: **client-up-script**

```
1  #!/bin/sh
2
3  INTERFACE=$1; shift;
4  TUN_MTU=$1;   shift;
5  UDP_MTU=$1;   shift;
6  LOCAL_IP=$1;  shift;
7  #REMOTE_IP=$1; shift;
8  #MODUS=$1;     shift;
9
10 /sbin/ifconfig ${INTERFACE} up
11 /sbin/ifconfig ${INTERFACE} mtu ${TUN_MTU}
12 /sbin/ifconfig ${INTERFACE} inet6 fa80:0:fc00::2/64
13 /sbin/route -n add -inet6 2001:: -prefixlen 3 -link tun0
14 exit 0
```

Remark: sometimes keepalive fails for unknown reason, ⇒ not yet ready for distribution

Quellen und Lesestoff

... only a few of more than 200 ...
RFC 2460 Internet Protocol, Version 6 (IPv6) Specification
RFC 2461 Neighbor Discovery for IP Version 6 (IPv6)
RFC 2462 IPv6 Stateless Address Autoconfiguration
RFC 2463 Internet Control Message Protocol for the Internet Protocol Version 6 (IPv6) Specification
RFC 2464 Transmission of IPv6 Packets over Ethernet Networks
RFC 3315 Dynamic Host Configuration Protocol for IPv6 (DHCPv6)
RFC 3756 IPv6 Neighbor Discovery (ND) Trust Models and Threats
RFC 3775 Mobility Support in IPv6
RFC 3971 SEcure Neighbor Discovery (SEND)
RFC 3972 Cryptographically Generated Addresses (CGA)
RFC 4429 Optimistic Duplicate Address Detection (DAD) for IPv6
RFC 4443 Internet Control Message Protocol for the Internet Protocol Version 6 (IPv6) Specification
RFC 4861 Neighbor Discovery for IPv6
RFC 4890 Recommendations for Filtering ICMPv6 Messages in Firewalls
RFC 5095 Deprecation of RH0

Linux:
http://www.bieringer.de/linux/IPv6/IPv6-HOWTO/IPv6-HOWTO.html
OpenVPN-tunnelbroker: http://blog.ghitr.com/index.php/archives/673
http://www.6net.org/publications/presentations/strauf-openvpn.pdf

Books:
IPv6 in Practice, Benedikt Stokebrand, Springer, ISBN 978-3-540-24524-7
IPv6, Sylvia Hagen, Sunny Edition, 2. Auflage, ISBN 978-3-9522842-2-2
Deploying IPv6 Networks, Ciprian Popoviciu et.al., Cisco Press, ISBN 1587052105

Tests:
http://freeworld.thc.org/thc-ipv6/
http://lg.he.net/

Security:
http://www.wecon.net/files/48/GUUG-RT_WEST2010-SvI.pdf
http://seanconvery.com/ipv6.html

Ich bedanke mich für Ihre Aufmerksamkeit

hubertz-it-consulting GmbH jederzeit zu Ihren Diensten

Ihre Sicherheit ist uns wichtig!

Frohes Schaffen

Johannes Hubertz

it-consulting _at_ hubertz dot de
H-alpha ∈ { kompetenzspektrum.de }

powered by LaTeX 2ε
and PSTricks