IPv6 – Theorie, Praxis, Realität

Felix Krohn
fx@kro.hn.org

UnFUG

19.1.2006

Übersicht

IPv6?

- Wieso?
- Wo?
- Wann?
- und vor allem: Wie?

IPv6 kurz & bündig

- Entwicklung begann um 1995, Parallelentwicklung in IPv4: NAT, DHCP
- Adressknappheit, v.a. im asiatischen Raum
  v4: $2^{32} = 4294967296$ IPs,
  v6: $2^{128} = 340282366920938463463374607431768211456$ IPs
- effizienteres Routing
  fixe Headergröße
  Wegfall von Fragmentierung
- Security
  Integriertes IPSec, integriertes QoS
- andere Features
  Autoconfiguration, Multicast, Mobility Extensions, ...
- ICMPv6
  sehr mächtig ausgebaut, übernimmt Funktion von ARP

IPv6 kurz & bündig

- Entwicklung begann um 1995, Parallelentwicklung in IPv4: NAT, DHCP
- Adressknappheit, v.a. im asiatischen Raum
  v4: $2^{32} = 4294967296$ IPs,
  v6: $2^{128} = 340282366920938463463374607431768211456$ IPs
- effizienteres Routing
  fixe Headergröße
  Wegfall von Fragmentierung
- Security
  Integriertes IPSec, integriertes QoS
- andere Features
  Autoconfiguration, Multicast, Mobility Extensions, ...
- ICMPv6
  sehr mächtig ausgebaut, übernimmt Funktion von ARP

IPv6 kurz & bündig

- Entwicklung begann um 1995, Parallelentwicklung in IPv4: NAT, DHCP
- Adressknappheit, v.a. im asiatischen Raum
  v4: $2^{32} = 4294967296$ IPs,
  v6: $2^{128} = 340282366920938463463374607431768211456$ IPs
- effizienteres Routing
  fixe Headergröße
  Wegfall von Fragmentierung
- Security
  Integriertes IPSec, integriertes QoS
- andere Features
  Autoconfiguration, Multicast, Mobility Extensions, ...
- ICMPv6
  sehr mächtig ausgebaut, übernimmt Funktion von ARP

IPv6 kurz & bündig

- Entwicklung begann um 1995, Parallelentwicklung in IPv4: NAT, DHCP
- Adressknappheit, v.a. im asiatischen Raum
  v4: $2^{32} = 4294967296$ IPs,
  v6: $2^{128} = 340282366920938463463374607431768211456$ IPs
- effizienteres Routing
  fixe Headergröße
  Wegfall von Fragmentierung
- Security
  Integriertes IPSec, integriertes QoS
- andere Features
  Autoconfiguration, Multicast, Mobility Extensions, ...
- ICMPv6
  sehr mächtig ausgebaut, übernimmt Funktion von ARP

IPv6 kurz & bündig

- Entwicklung begann um 1995, Parallelentwicklung in IPv4: NAT, DHCP
- Adressknappheit, v.a. im asiatischen Raum
  v4: $2^{32} = 4294967296$ IPs,
  v6: $2^{128} = 340282366920938463463374607431768211456$ IPs
- effizienteres Routing
  fixe Headergröße
  Wegfall von Fragmentierung
- Security
  Integriertes IPSec, integriertes QoS
- andere Features
  Autoconfiguration, Multicast, Mobility Extensions, ...
- ICMPv6
  sehr mächtig ausgebaut, übernimmt Funktion von ARP

IPv6 kurz & bündig

- Entwicklung begann um 1995, Parallelentwicklung in IPv4: NAT, DHCP
- Adressknappheit, v.a. im asiatischen Raum
  v4: $2^{32} = 4294967296$ IPs,
  v6: $2^{128} = 340282366920938463463374607431768211456$ IPs
- effizienteres Routing
  fixe Headergröße
  Wegfall von Fragmentierung
- Security
  Integriertes IPSec, integriertes QoS
- andere Features
  Autoconfiguration, Multicast, Mobility Extensions, ...
- ICMPv6
  sehr mächtig ausgebaut, übernimmt Funktion von ARP

Wie sieht so etwas aus?

128 bit in Hexadezimaldarstellung, zusammengesetzt aus 8 “nubbles” zu je 16bit
z.B. 2001:0DB8:0f00:0042:02c0:4fff:fe83:22a9

Es hätte auch *noch* schlimmer kommen können:

9S0uFTvzT2&w)7an>TI*

(nach RFC1924: “A Compact Representation of IPv6 Addresses” vom 1. April 1996)

Wie sieht so etwas aus?

128 bit in Hexadezimaldarstellung, zusammengesetzt aus 8 “nubbles” zu je 16bit
z.B. 2001:0DB8:0f00:0042:02c0:4fff:fe83:22a9

Es hätte auch *noch* schlimmer kommen können:

9S0uFTvzT2&w)7an>TI*

(nach RFC1924: “A Compact Representation of IPv6 Addresses” vom 1. April 1996)

Wie sieht so etwas aus?

128 bit in Hexadezimaldarstellung, zusammengesetzt aus 8 “nubbles” zu je 16bit
z.B. 2001:0DB8:0f00:0042:02c0:4fff:fe83:22a9

Es hätte auch *noch* schlimmer kommen können:

9S0uFTvzT2&w)7an>TI*

(nach RFC1924: “A Compact Representation of IPv6 Addresses” vom 1. April 1996)

Wie sieht so etwas aus?

Führende Nullen innerhalb eines nubbles dürfen weggelassen werden. Außerdem darf man genau einmal innerhalb einer Adresse eine Folge von 0-Blöcken durch "::" ersetzen.
Folgende Schreibweisen sind also gleichwertig:
2001:DB8:f00::1
2001:DB8:f00:0:0:0:0:1
2001:DB8:f00:0000:0000:0000:0000:1

Wie sieht so etwas aus?

- wie bei IPv4 (CIDR) gewohnt, wird die Netzmaske durch ein "/" getrennt angehängt und bezeichnet die Anzahl bits, die Adressen aus diesem Netz gemein haben.
- "normale" Subnetze haben i.d.R. die Netzmaske "/64"
- Gängig sind /32 für Provider und (sehr) grosse Organisationen, /48 oder /64 für normale Benutzer.

verschiedene Adresspräfixe

Prefix	Bedeutung
fe8x	link local
fe9x - febx	reserviert für link local
fecx - fefx	reserviert für site local (obsolet)
2xxx, 3xxx	2000::/3 - aggretable global unicast
3ffe::/16	6bone-Test-Adressen
ffxy	Multicast-Bereich
ff02::1	ip6-allnodes
ff02::2	ip6-allrouters
ff02::3	ip6-allhosts

Beispiele

Schreibweise oft in eckigen Klammern:

```
http://[2001:DB8:f00:02c0:4fff:fe83:22a9]/foo
```

u.U. diese noch escapen:

```
scp foo@\[2001:DB8:f00:02c0:4fff:fe83:22a9::42\]:bar /baz
```

Header

Feld	Bit	Beschreibung
Version	4	momentan 4 oder 6
Priorität	4	
flow Label	24	
Length	16	Nutzdaten in Bytes
Next Header	8	entspricht dem v4-*Protocol*-Header
Hop Limit	8	TTL bei IPv4
Quell- und Ziel-Adresse	je 128	...

Ablauf der Autoconfiguration

1. link-local-Adresse wird mithilfe des Präfix fe80::/64 und der EUI-64-codierten MAC-Adresse gebildet
2. DAD mit "neighbourhood solicitation message"
3. Wenn noch frei: Adresse wird benutzt
   Wenn nicht: dumm gelaufen.
4. "Router Solicitation Message" an ff02::2
5. Router schickt Präfix, Host bastelt sich Adresse zusammen
6. DAD (Duplicate Address Detection)

Ablauf der Autoconfiguration

1. link-local-Adresse wird mithilfe des Präfix fe80::/64 und der EUI-64-codierten MAC-Adresse gebildet
2. DAD mit “neighbourhood solicitation message”
3. Wenn noch frei: Adresse wird benutzt
   Wenn nicht: dumm gelaufen.
4. “Router Solicitation Message” an ff02::2
5. Router schickt Präfix, Host bastelt sich Adresse zusammen
6. DAD (Duplicate Address Detection)

Ablauf der Autoconfiguration

1. link-local-Adresse wird mithilfe des Präfix fe80::/64 und der EUI-64-codierten MAC-Adresse gebildet
2. DAD mit “neighbourhood solicitation message”
3. Wenn noch frei: Adresse wird benutzt
   Wenn nicht: dumm gelaufen.
4. “Router Solicitation Message” an ff02::2
5. Router schickt Präfix, Host bastelt sich Adresse zusammen
6. DAD (Duplicate Address Detection)

Ablauf der Autoconfiguration

1. link-local-Adresse wird mithilfe des Präfix fe80::/64 und der EUI-64-codierten MAC-Adresse gebildet
2. DAD mit "neighbourhood solicitation message"
3. Wenn noch frei: Adresse wird benutzt
   Wenn nicht: dumm gelaufen.
4. "Router Solicitation Message" an ff02::2
5. Router schickt Präfix, Host bastelt sich Adresse zusammen
6. DAD (Duplicate Address Detection)

Ablauf der Autoconfiguration

1. link-local-Adresse wird mithilfe des Präfix fe80::/64 und der EUI-64-codierten MAC-Adresse gebildet
2. DAD mit "neighbourhood solicitation message"
3. Wenn noch frei: Adresse wird benutzt
   Wenn nicht: dumm gelaufen.
4. "Router Solicitation Message" an ff02::2
5. Router schickt Präfix, Host bastelt sich Adresse zusammen
6. DAD (Duplicate Address Detection)

Ablauf der Autoconfiguration

1. link-local-Adresse wird mithilfe des Präfix fe80::/64 und der EUI-64-codierten MAC-Adresse gebildet
2. DAD mit "neighbourhood solicitation message"
3. Wenn noch frei: Adresse wird benutzt
   Wenn nicht: dumm gelaufen.
4. "Router Solicitation Message" an ff02::2
5. Router schickt Präfix, Host bastelt sich Adresse zusammen
6. DAD (Duplicate Address Detection)

Tunnel

- IPv6-Netze werden via Tunnel über das IPv4-Netz verbunden
- Für IPv6 Knoten ist das Tunneling völlig transparent
- Alle IPv6 Erweiterungen können benutzt werden
- Tunnel sind in Bandbreite und Durchsatz nur durch das zugrundeliegende IPv4 beschränkt
- Nachteile:
  kleinere MTU, größere Latenz

Tunnel

- IPv6-Netze werden via Tunnel über das IPv4-Netz verbunden
- Für IPv6 Knoten ist das Tunneling völlig transparent
- Alle IPv6 Erweiterungen können benutzt werden
- Tunnel sind in Bandbreite und Durchsatz nur durch das zugrundeliegende IPv4 beschränkt
- Nachteile:
  kleinere MTU, größere Latenz

Tunnel

- IPv6-Netze werden via Tunnel über das IPv4-Netz verbunden
- Für IPv6 Knoten ist das Tunneling völlig transparent
- Alle IPv6 Erweiterungen können benutzt werden
- Tunnel sind in Bandbreite und Durchsatz nur durch das zugrundeliegende IPv4 beschränkt
- Nachteile:
  kleinere MTU, größere Latenz

Tunnel

- IPv6-Netze werden via Tunnel über das IPv4-Netz verbunden
- Für IPv6 Knoten ist das Tunneling völlig transparent
- Alle IPv6 Erweiterungen können benutzt werden
- Tunnel sind in Bandbreite und Durchsatz nur durch das zugrundeliegende IPv4 beschränkt
- Nachteile:
  kleinere MTU, größere Latenz

Tunnel

- IPv6-Netze werden via Tunnel über das IPv4-Netz verbunden
- Für IPv6 Knoten ist das Tunneling völlig transparent
- Alle IPv6 Erweiterungen können benutzt werden
- Tunnel sind in Bandbreite und Durchsatz nur durch das zugrundeliegende IPv4 beschränkt
- Nachteile:
  kleinere MTU, größere Latenz

Tunnelarten

- statische Tunnel
  Knoten braucht statische und weltweit gültige IPv4-Adresse, Routen werden zum Tunnel-Endpunkt gesetzt.
- automatische Tunnel
  IPv4 kompatible IPv6 Adressen wurden durch den 6to4-Mechanismus ersetzt. Ermöglichte ehemals Netze der Form ::a.b.c.d/96
- 6to4-Tunnel (RFC 3056)
  jede IPv4-Adresse kann auch in ein /48-IPv6-Netz umgewandelt werden, benötigt jedoch einen 6to4-Gateway für den Upstream
  z.B. 6to4.ipv6.microsoft.com oder bei Xaranet (gegen €)
  mögliches Sicherheitsloch im Netz!

Tunnelarten

- statische Tunnel
  Knoten braucht statische und weltweit gültige IPv4-Adresse, Routen werden zum Tunnel-Endpunkt gesetzt.
- automatische Tunnel
  IPv4 kompatible IPv6 Adressen wurden durch den 6to4-Mechanismus ersetzt. Ermöglichte ehemals Netze der Form ::a.b.c.d/96
- 6to4-Tunnel (RFC 3056)
  jede IPv4-Adresse kann auch in ein /48-IPv6-Netz umgewandelt werden, benötigt jedoch einen 6to4-Gateway für den Upstream
  z.B. 6to4.ipv6.microsoft.com oder bei Xaranet (gegen €)
  mögliches Sicherheitsloch im Netz!

Tunnelarten

- statische Tunnel
  Knoten braucht statische und weltweit gültige IPv4-Adresse, Routen werden zum Tunnel-Endpunkt gesetzt.
- automatische Tunnel
  IPv4 kompatible IPv6 Adressen wurden durch den 6to4-Mechanismus ersetzt. Ermöglichte ehemals Netze der Form ::a.b.c.d/96
- 6to4-Tunnel (RFC 3056)
  jede IPv4-Adresse kann auch in ein /48-IPv6-Netz umgewandelt werden, benötigt jedoch einen 6to4-Gateway für den Upstream
  z.B. 6to4.ipv6.microsoft.com oder bei Xaranet (gegen €)
  mögliches Sicherheitsloch im Netz!

Beispiel 6to4

6to4-Prefix + hexadezimale IPv4-Adresse + 80 bit Adressraum

```
$ echo "2002:\
   ‘printf "\%02x\%02x:\%02x\%02x\n" 141 28 12 34‘::"
2002:8d1c:0c22::
$
```

Dual-Stack

- Die einfachste und beste Lösung. ABER: der Provider muss es anbieten
- Ausweg: einen 2. Provider, der den IPv6-Uplink bietet, und dann selber routen
  z.B. bei "titan-dsl.de" [42] oder regionale Provider

[42] bietet inzwischen eine kostenlose nur-IPv6-Flatrate an

Dual-Stack

- Die einfachste und beste Lösung. ABER: der Provider muss es anbieten
- Ausweg: einen 2. Provider, der den IPv6-Uplink bietet, und dann selber routen
  z.B. bei “titan-dsl.de” [42] oder regionale Provider

[42] bietet inzwischen eine kostenlose nur-IPv6-Flatrate an

andere Techniken

- NAT-PT
  weist IPv6-Nodes dynamisch IPv4-Adressen zu
- TRT (Transport Relay Translator)
  beendet die Verbindungen auf der einen Seite und baut diese auf der anderen Seite wieder auf
  Sehr aufwendig und nichts, was man haben will

andere Techniken

- NAT-PT
  weist IPv6-Nodes dynamisch IPv4-Adressen zu
- TRT (Transport Relay Translator)
  beendet die Verbindungen auf der einen Seite und baut diese auf der anderen Seite wieder auf
  Sehr aufwendig und nichts, was man haben will

iproute2

vereinigt die Funktionen von *ifconfig*, *arp*, *route*, *iptunnel* und noch mehr (IPv6!) in einem Befehl mit einer Syntax.

- ifconfig
- arp
- route
- iptunnel
- und noch mehr ...
- und: es vereinfacht die IPv6-Konfiguration!

Für uns ist insbesondere der Switch "-6" von Bedeutung
Beispiel:

iproute2

vereinigt die Funktionen von *ifconfig*, *arp*, *route*, *iptunnel* und noch mehr (IPv6!) in einem Befehl mit einer Syntax.

- ifconfig
- arp
- route
- iptunnel
- und noch mehr ...
- und: es vereinfacht die IPv6-Konfiguration!

Für uns ist insbesondere der Switch "-6" von Bedeutung
Beispiel:

iproute2 - Beispiel

```
# ip -6 r s
2000::/3 dev tun  metric 1  mtu 1280 advmss 1220 hoplimit 429496
fe80::/64 dev eth0  metric 256  mtu 1500 advmss 1220 hoplimit 42
ff00::/8 dev eth0  metric 256  mtu 1500 advmss 1220 hoplimit 429
unreachable default dev lo  proto none  metric -1  error -101 ad
# ip r s
192.168.36.0/24 dev eth0  proto kernel  scope link  src 192.168.
default via 192.168.36.1 dev eth0
# ip -6 r a default via 2001:DB8:f00:: dev tun
#
```

radvd: Autoconfiguration unter Linux

Daemon im Userspace kuemmert sich um die im Protokoll vorgesehene Konfiguration im Netz:

- Adressvergabe
- Bekanntmachung von Routen
- denkbar einfache Konfiguration
- Zusatzprogramm *radvdump* um radv-Meldungen anzuzeigen (debuggen!)
- Hierfür wird dann DHCPv6 benötigt, wenn keine manuelle Konfiguration verwendet werden soll
- Nachteil: keine DNS-Konfiguration

radvd-config

```
interface eth1
{
        AdvSendAdvert on;
        MinRtrAdvInterval 30;
        MaxRtrAdvInterval 60;
        prefix 2001:DB8:f00::/64
        {
                AdvOnLink on;
                AdvRouterAddr on;
                AdvAutonomous on;      # unsolicited rtadv
        };
};
```

andere nützliche tools, die IPv6 koennen

- nmap
- tcpdump
- ping6
- traceroute6
- ipv6calc
  ist nicht unabdingbar, aber verdammt nützlich:

ipv6calc

```
$ ipv6calc --addr_to_base85 2001:0DB8:0f00:0042:02c0:4fff:fe83:2
9R}vSUrOB|CfeOs%cHZ(

$ ipv6calc --in ipv6addr --out revnibbles.arpa 2001:0DB8:0f00:00
9.a.2.2.3.8.e.f.f.f.f.4.0.c.2.0.2.4.0.0.0.0.f.0.8.b.d.0.1.0.0.2.
```

IPv6-aware Software

- DNS: Bind9, NSD, djbdns (gepatcht)
- smtp: exim, courier, postfix, qsmtp
- imap: courier, cyrus, dovecot, binc
- ftp: pureftpd, proftpd, vsftpd, ncftp, ...
- http: apache2, boa, thttpd, wget (gepatcht)
- irc: irssi, bitchx, mIRC (gepatcht)
- rsync, ssh, jabberd, xinetd, ...

nützliche Links

Howtos von JOIN

Linux IPv6 HOWTO

v4_v6_translation

Sixxs

gute & ausführliche Doku zu iproute

Tests des eigenen Nameservice

RFC 3513 - IP Version 6 Addressing Architecture

RFC 3587 - An IPv6 Aggregatable Global Unicast Address Format

RFC 3056 - Connection of IPv6 Domains via IPv4 Clouds (6to4)

RFC 3964 - Security considerations for 6to4

RFC 2461 - Neighbor Discovery for IP Version 6

Fragen?

- ...