basics

21. August 2010

Hubert Denkmair <hubert.denkmair@bingo-ev.de>
Thomas Jakobi <fake@bingo-ev.de>

... ist im Prinzip wie IPv4, nur die Adressen sehen anders aus.

Vielen Dank für Eure Aufmerksamkeit! *

* a tribute to Gert Döring

Ablauf

- Probleme von IPv4
- Wie funktioniert v6?
- Praxistest
- Fallstricke für User, Admins, Programmierer
- Woher bekomme ich v6 Connectivity?
- No questions - All answered :-)

Warum stinkt IPv4?

- Zu kleiner Adressraum!
- Von 4 Mrd IPv4-Adressen sind 1-2 Mrd nutzbar
- Raum stark fragmentiert -> Üble Routingtabellen
- Heute schon mehr User als verfügbare IPs
- Künftig: Viele Devices pro User
- Derzeitige Lösung: NAT
  Viele Benutzer / Ganze Netze hinter einer einzelnen IP-Adresse verborgen

Warum stinkt NAT?

- Ende der demokratischen Idee des Internet:
  Wenn ich keine öffentliche Adresse mehr habe, kann ich selbst keine Dienste anbieten!
- Kommunikation überhaupt schwierig:
  Ende-zu-Ende-Verbindungen nur über Tricks möglich
- Private Adressbereiche machen Probleme,
  wenn ich mehrere Netze verbinden will.
  (Beispiel: VPN)

Was ist IPv6?

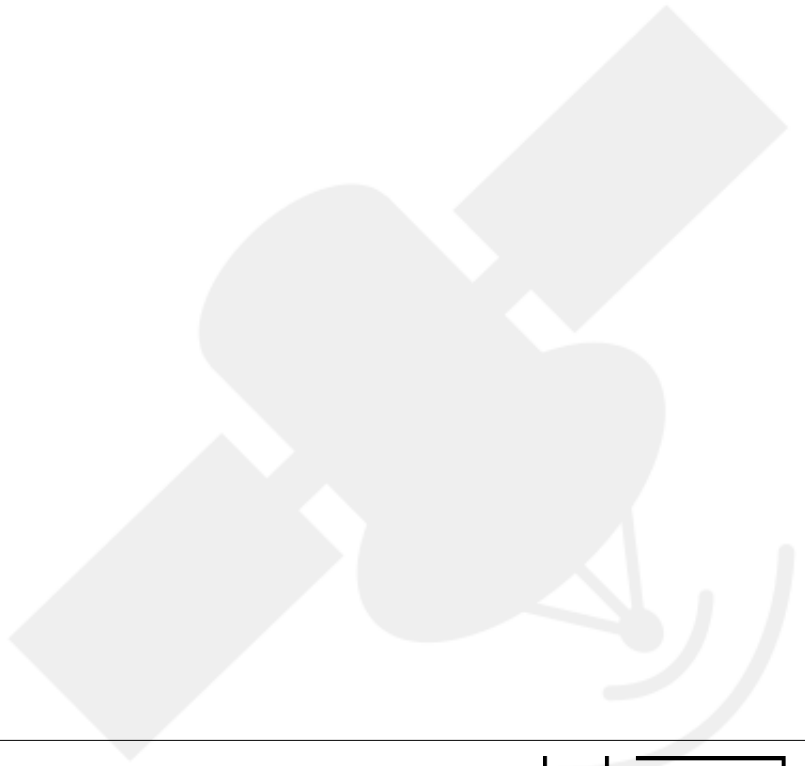
- Offizieller Nachfolger von IPv4
- Entwicklung seit 1995 von der IETF
- Seit 1998 spezifiziert (RFC2460)
- Eigenes Layer3-Protokoll, soll IPv4 ersetzen
- Läuft (natürlich) auch parallel zu IPv4
- Die Schichten darüber & darunter (Ethernet/ PPP, TCP/UDP, DNS, HTTP...) bleiben unangetastet
- Grundlegende Prinzipien gleich: Bestehendes Grundlagenwissen bleibt gültig!

Was ist neu bei IPv6?

- VIEL größerer Adressraum („fröhliche Verschwendung“)
- NAT unnötig (nicht: Unmöglich!)
- Einfacherer Header
- Automatische Adresskonfiguration (kein DHCP nötig)
- Mobile IP
- Multihoming
- IPsec Teil des Protokolls
- QoS und Multicast von Anfang an integriert

Was ist **wichtig** bei IPv6?

- VIEL größerer Adressraum („fröhliche Verschwendung“)
- NAT unnötig (nicht: Unmöglich!)



IPv6 Adressraum

- IPv4: 2^{32} = 4 294 967 296 Adressen
- IPv6: 2^{128} = **genug** Adressen

340.282.366.920.938.463.463.374.607.431

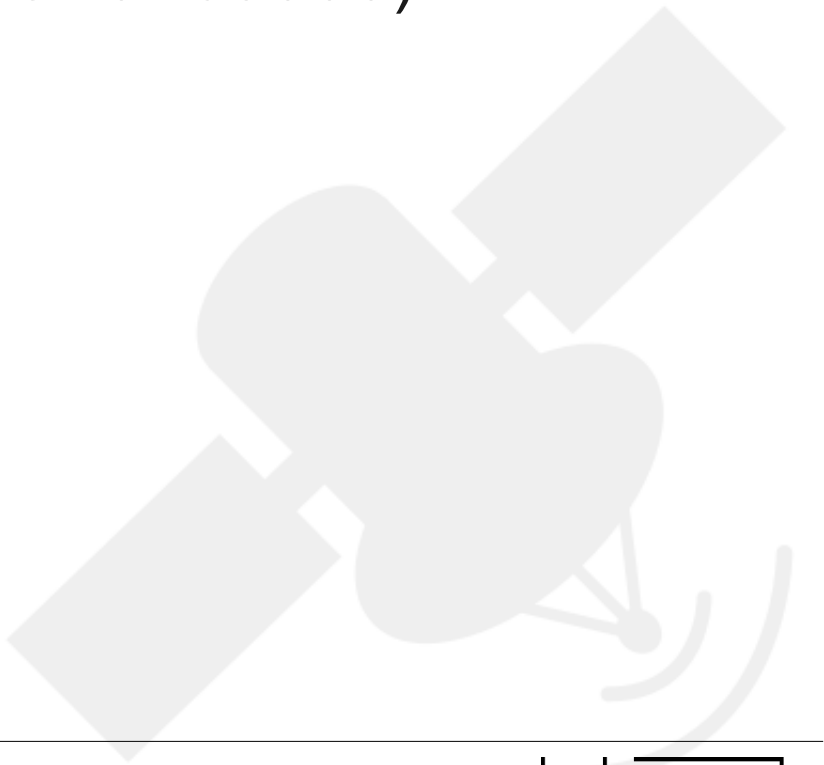
Milliarden

Adressnotation

- 8 Blocks à 4 Hex-Ziffern:
  `2001:0a60:f0a0:feed:fefe:c0de:c0ff:ee11`
  oder:
  `2001:0a60:f0a0:0000:0000:0000:0000:0001`
- gekürzt: `2001:a60:f0a0::1`
- Als URL: `http://[2001:a60:f0a0::1]:8080/`
- Subnetz-Präfix: `2001:a60:f0a0::/48`
- Host mit Präfix: `2001:a60:f0a0::1/48`
- DNS funktioniert auch für IPv6!

Adressbereiche

- undefiniert - `::/128`
- local loopback - `::1/128` (entspricht `127.0.0.1`)
- link local - `fe80::/10` (entspricht `192.168...`)
- unique local (`fc00::/7`)
- multicast (`ff00::/8`)
- global unicast (der Rest)

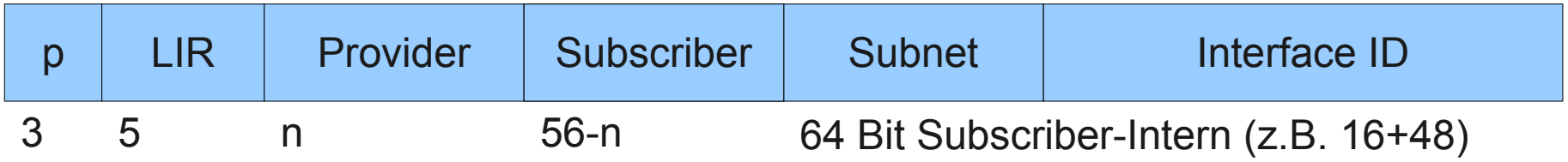


Global Unicast Adressen

- `0:0:0:0:0:ffff::/96` - IPv4 mapped
- `2000::/3` - IANA
  - `2001` - An Provider vergeben via LIR
  - `2001:db8::/32` - example / Dokumentation
  - `2002` - 6to4 Tunnel
  - `2003, 2400, 2600, 2800, 2A00, 2C00`
    An Provider vergeben via LIR

Quelle: http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.txt

Adressraum



- 5 Bit für die Local Registry (z.B. RIPE)
- Ausreichend Adressraum für die Provider:
  - Minimum: `/32` == 64K `/48`-Subnetze für Kunden
  - DTAG: `/19` == 500 Mio `/48`-Subnetze für Kunden
- Default-Netz für ein LAN immer „groß genug“ (64 Bit)

Autokonfiguration (1)

- Jeder Rechner weist sich selbst eine link-local-Adresse, basierend auf seiner MAC, zu:

```
MAC-Adresse:          00:13:02:25:d8:bc
IPv6-Adresse: fe80::213:2ff:fe25:d8bc
```

- `fe80::` + „Interface Identifer“ (IEE EUI-64)
- Weltweit eindeutige, gleichbleibende Adresse:
  Besser als jedes tracking cookie!
- Deshalb: privacy extensions.

Autokonfiguration (2)

- Host schickt einen Broadcast an ff02::2
  (alle lokalen Router)
- Router antworten mit Liste von verfügbaren Präfixen
  (z.B. `2001:a60:f0a0::/48`)
- Host gibt sich selbst eine Adresse, bestehend aus Präfix und Interface Identifer:

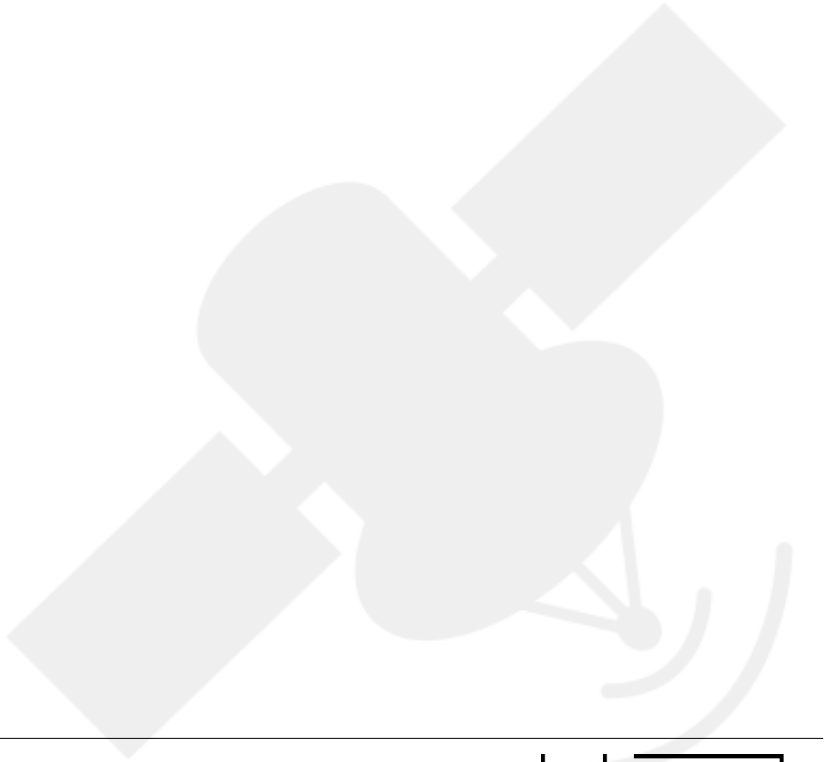
  `2001:a60:f0a0::213:2ff:fe25:d8bc`
- „Duplicate Address Detection“ (DAD) verhindert die doppelte Vergabe einer Adresse
- Router verschicken regelmäßig Updates ihrer Daten

Autokonfiguration (3)

- Vorteil: Stateless autoconfiguration!
- Es muss nicht darüber Buch geführt werden, wem welche Adressen zugeteilt sind.
- Problem: Keine Konfiguration von DNS, Hostname, NTP...
- Darum: Trotzdem wieder DHCP(v6), aber stateless
- Notfalls auch ohne:
  Lokale DNS-Server lassen sich per Multicast finden

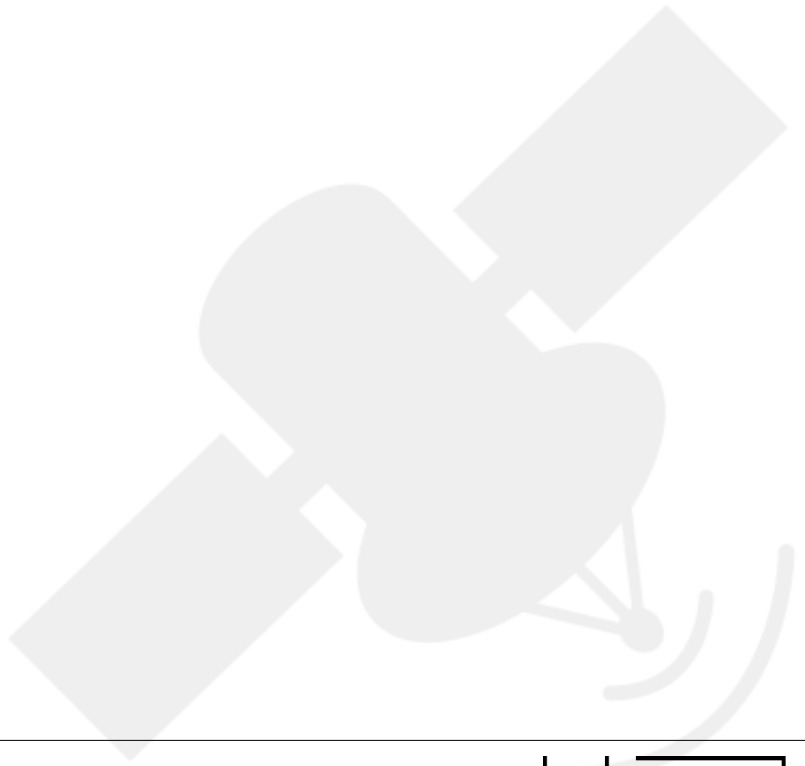
DNS

- Eigener Ressource Type (AAAA) für IPv6
  (z.B. `www IN AAAA 2001:a60:f0a0::23`)
- Reverse lookup via Domain ip6.arpa.



That's it!

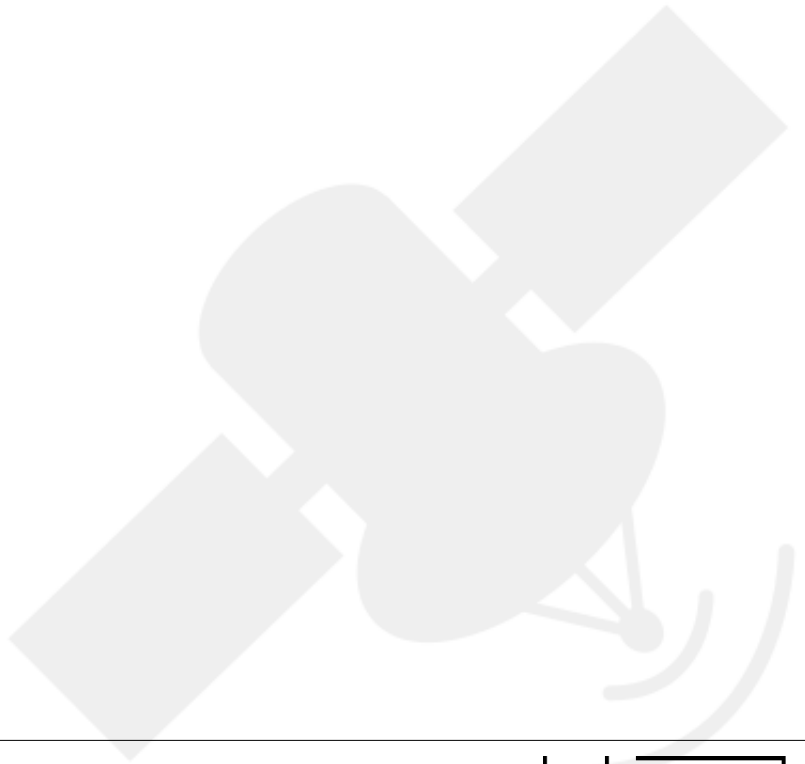
- Alle aktuellen Betriebssysteme sprechen IPv6 per default
- Alles andere bleibt, wie's ist.
- In der Theorie!
- Für den User: Auch in der Praxis.
- Sind User anwesend?



Praxistest

Probleme in der Praxis: User

- Es gibt (noch immer) kein IPv6 vom ISP
- Firewalls droppen IPv6-SYNs SILENTIY m-)
- Kaputte Protokolle übermitteln IPv4-Adressen und funktionieren nicht mit v6



Probleme in der Praxis: Admin

- NAT überflüssig. Brauche ich jetzt eine Firewall? m-)
- Doppelter Aufwand, um Routingtabellen, Firewall-Regeln... für v4 und v6 zu pflegen & testen
- Aktuelle Serversoftware unterstützt meist problemlos IPv6, will aber konfiguriert werden.
  (Apache „Listen 1.2.3.4:80“ horcht nicht auf v6)
- OpenVZ zählt wohl nicht zur aktuellen Serversoftware :-(
- IP-Listen (blacklists, whitelists, hosts.allow...) auf v6-Adressen ausdehnen
- Firewalls droppen IPv6-SYNs SILENTIY m-)

IPv6 in Debian/Ubuntu

- `apt-get install aiccu radvd`
- `sysctl -w net.ipv6.conf.all.forwarding=1`
- `cp /usr/share/doc/radvd/examples/simple-radvd.conf /etc/radvd.conf`

```
interface eth0
{
   AdvSendAdvert on;
   prefix 2001:a60:f0a0::/64
   {
   };
};
```

- `service radvd start`
- `ping6 ipv6.google.com`

```
PING ipv6.google.com(2a00:1450:8007::63) 56 data bytes
64 bytes from 2a00:1450:8007::63: icmp_seq=1 ttl=56 time=33.0 ms
```

Probleme in der Praxis: Coder

- Webanwendung:
  Berücksichtigen, dass die Client-Adresse anders aussehen könnte als xxx.xxx.xxx.xxx
  Sonst alles gut.
- Socket-Programmierung:
  Je nach Anwendung keine Änderung notwendig oder wirklich Arbeit zu investieren...

IPv6 Sockets in Python

Server:

```
 1 from socket import *
 2
 3 UDPSock = socket(AF_INET6, SOCK_DGRAM)
 4 UDPSock.bind(("::1",9000))
 5
 6 while 1:
 7   data,addr = UDPSock.recvfrom(1024)
 8   if not data:
 9     break
10   print data
11
12 UDPSock.close()
```

Client:

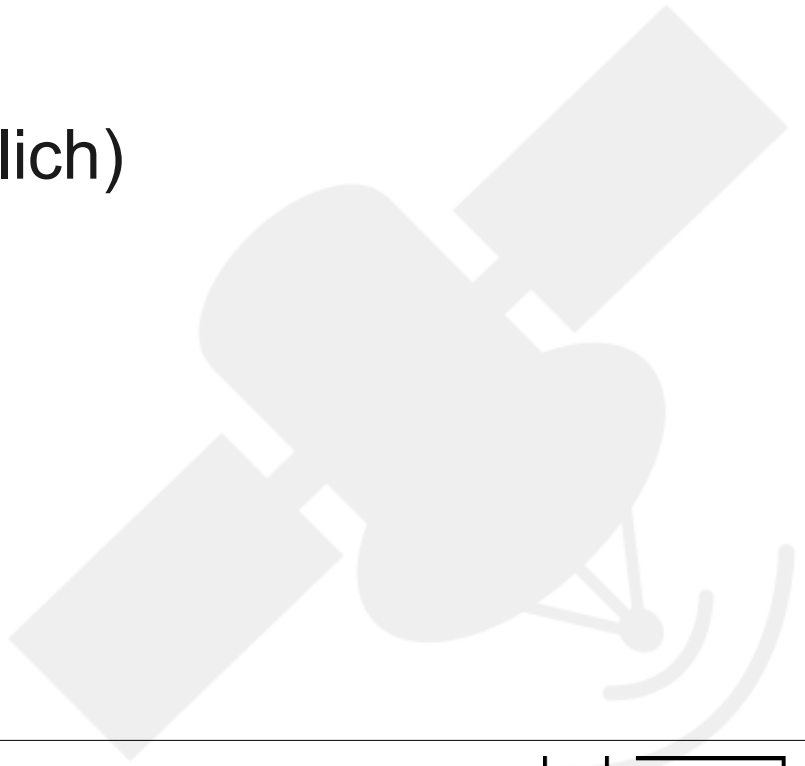
```
 1 from socket import *
 2
 3 UDPSock = socket(AF_INET6,SOCK_DGRAM)
 4 addr = ("::1", 9000)
 5
 6 while (1):
 7   data = raw_input('>> ')
 8   if not data:
 9     break
10   UDPSock.sendto(data,addr)
11
12 UDPSock.close()
```

getaddrinfo() in Python

```python
import sys
import socket

# usage: python echo_server.py <addr> <port>
host = sys.argv[1]
port = sys.argv[2]

l = socket.getaddrinfo(host, port, 0, 0, socket.SOL_TCP)

if len(l)<1:
  print "getaddrinfo() failed"
  sys.exit(1)

af, socktype, proto, canonname, sa = l[0]
sock = socket.socket(af, socktype, proto)
sock.bind(sa)
sock.listen(1)

while 1:
  conn, addr = sock.accept()
  print 'Connected by', addr
  while 1:
    data = conn.recv(1024)
    if not data: break
    conn.send(data)
    print data
  conn.close()
```

Woran scheitert's?

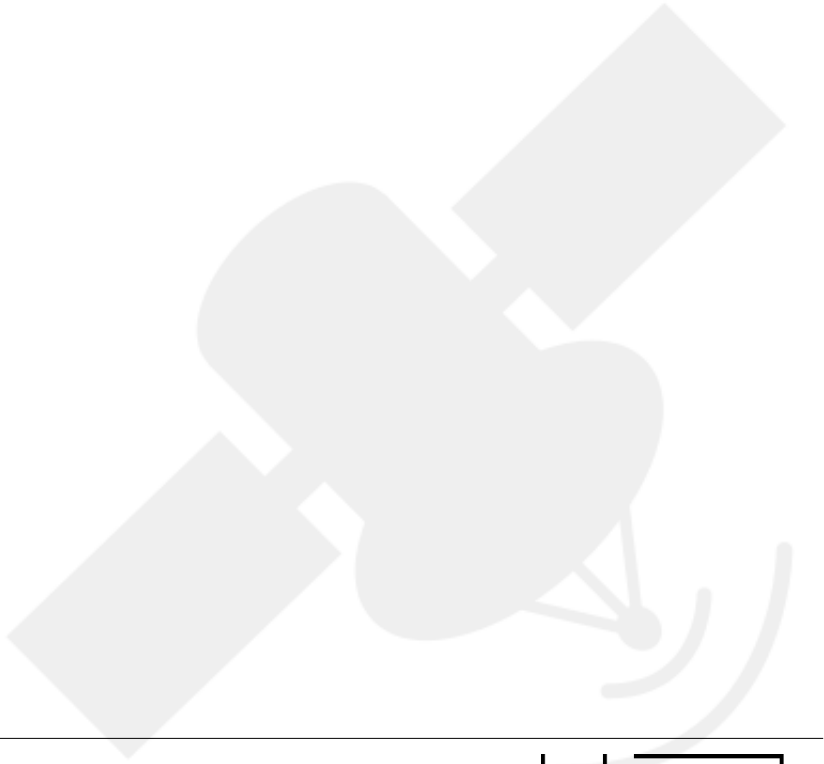
- Henne-Ei-Problem
- Es wird erst was passieren,
  wenn es nicht mehr anders geht
- **DAS ENDE IST NAH!** ™
- Alle großen Provider wollen (angeblich)
  bis Ende 2010 (sic) IPv6 anbieten.
  Ehrlich.



IPv6 jetzt. Wie?

- Geschäftskunden hätten i.d.R. kein Problem, an IPv6 zu kommen - Kaum Nachfrage?
- Rootserver-Provider bieten bereits IPv6 an, z.B. jeder Hetzner-Rootserver mit /64-Subnet
- Spezielle T-DSL-Reseller
- 6to4-Tunnel-Anbieter, z.B. www.sixxs.net
- „bald“: beta-Test bei m'net. Sie melden sich bei uns :-)

Fragen?



... ist im Prinzip wie IPv4, nur die Adressen sehen anders aus.

Vielen Dank für Eure Aufmerksamkeit! *

* a tribute to Gert Döring