IPv6 und Perl

Überblick

- IPv6 == Internet Protokoll Version 6
- Vortrag enthält kurze Intro zu IPv6
- zeigt Probleme bei der Migration unabhängig von Perl
- geht auf den Stand der Unterstützung durch Perl ein
- und zeigt Möglichkeiten, wie man den aktuellen Stand verbessern kann

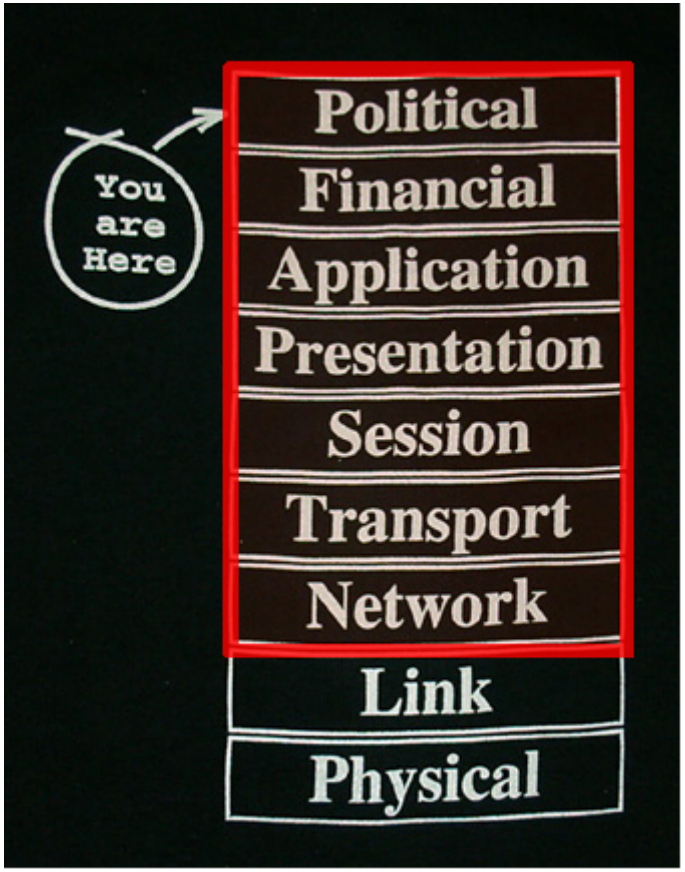
Hintergrund

- arbeite bei GeNUA mbH (Hochsicherheitsfirewalls u.a.)
- 05/2008 Internetaktionsplan der EU pusht IPv6
- 05/2008 "IVBB Nachfolger soll auch IPv6 verwenden"
- wir stellen die Firewalls für den IVBB her
- die Application Level Gateways auf diesen Firewalls sind i.A in Perl geschrieben
- "IPv6 Blog: Perl considered harmful"

Warum IPv6 ?

- IPv4 Adressen werden knapp, da nur 32-bit
- IPv6 bringt 128-bit statt 32-bit Adressierung
- ausreichend, das alles (Computer, Handy, MP3-Player, Kafeemaschine...) eine weltweit eindeutige Adresse bekommt
- kein NAT mehr nötig
- keine Workarounds mehr für P2P, VoIP,.. mehr nötig

Wo muss es unterstützt werden



Wo muss es unterstützt werden

- OSI Layer >=3
- Routing Protokolle
- DNS
- Betriebssystem Kernel und Tools
- Socket API
- Protokolle wie FTP, SIP, H.323...
- Firewalls, Router, Telefone...
- Anwendungen (Browser, Webserver, MUA, MTA...)
- ...

Inkompatibilitäten zu IPv4

- binäre Darstellung 128-bit statt 32-bit
- andere Textdarstellung der Adressen
- verschiedene Textdarstellungen für gleiche Adresse (verkürzte Darstellungen, z.B. 0:0:0:0:0:0:0:1 gleich ::1)
- einiges komplexer aber auch konsistenter: vieles, was Add-On zu IPv4 war ist jetzt im Protokoll integriert (Autokonfiguration, Unicast, Multicast, IPSec...)
- Link steht nicht gleich nach ifconfig zur Verfügung (Kollisionskontrolle)
- ...

Migration von IPv4 zu IPv6

Parallelbetrieb IPv4/IPv6

- wohin verbindet man, wenn Hostname als IPv4 (A) und auch IPv6 (AAAA) auflöst
- IPv6 im Tunnel geht andere Routen als IPv4
- Host evtl. über IPv4 aber nicht über IPv6 erreichbar

DNS im Mischbetrieb

- manche DNS Server antworten auf AAAA Anfragen NXDOMAIN wenn es keine IPv6, aber IPv4 Adressen gibt:
  - US-Cert VU#714121
  - PerlMonks: "ftp.perl.org not found".
- böse Spielchen mit AAAA und A records und searchdomains
  - in `/etc/resolv.conf` searchdomain Eintrag .local
  - DNS Server für .local liefert AAAA Adressen für *.local
  - lookup paypal.com liefert nicht A Record für paypal.com, sondern:
    - lookup AAAA:paypal.com -> keine Records
    - lookup AAAA:paypal.com.local -> Erfolg

Textdarstellung

- anders als IPv4
- gleiche Adresse kann auf verschiedene Weise ausgedrückt werden
- mapped IPv4, d.h. IPv4 kann innerhalb von IPv6 Adresse ausgedrückt werden:
  - `10.0.3.4`
  - `::FFFF:10.0.3.4`
  - `::FFFF:a00:304`
  - `0:0:0:0:0:ffff:a00:304`
- Kombination von IP und Port in verschiedenen Anwendungen verschieden umgesetzt. Bei IPv4 i.A IP:Port, bei IPv6 mal `[IP6]:Port`, mal `IP6.Port`.
- Probleme, wenn ':' als Seperator zwischen IP und anderen IPs oder Port o.a. genommen wurde (zB bei exim)
- ...

Textform wird verwendet in...

- PFL Regeln
- Zugriffslisten (z.B. squid ACLs)
- Konfiguration (z.B. Listenadresse)
- URLs (proto://host/)
- innerhalb der Protokolle, z.B.
  - SIP: SDP gibt Adressen an zum Transport der Mediadaten
  - Mail: user@[ip]
  - FTP: PORT,EPRT... geben Adressen für Datenverbindung an
  - DNS: A und AAAA Records
- X509 Zertifikaten
- ...

Perl spezifische Intro

CORE

- keinerlei Unterstützung durch CORE Module
- gethost* liefern was libc/libresolv liefert, d.h nur IPv4
- stattdessen muss getaddrinfo genutzt werden, welches aber nicht im CORE ist

Module

- Basis ist Socket API, d.h Socket6 und darauf aufbauend IO::Socket::INET6
- Net::DNS unterstützt AAAA records
- viele Module benutzen explizit IO::Socket::INET (Net::SMTP, Net::FTP, LWP...) und sind daher nicht IPv6 fähig
- diverse andere Module unterstützen IPv6, aber fraglich in welcher Qualität
- d.h. schlechtes out of the box Erlebnis :-((

ist nur Perl so schlecht?

- siehe IPv6 Blog: "IPv6 Support in Programming Languages"
- Python, Java, Ruby: kein spezielles IPv6 API - ist im normalen Socket API mit drin (so sollte es sein)
- PHP - ähnlich schlimm wie Perl

im weiteren....

- Benutzung von Socket6, IO::Socket::INET6
- SSL und IPv6
- Net::DNS
- Net::SMTP, Net::POP3...
- LWP
- Net::FTP
- Arbeiten mit Textdarstellung...: NetAddr::IP, Net::IP...

Socket6, IO::Socket::INET6

- API analog zu Socket, IO::Socket::INET
- kann mit IPv6 (AF_INET6) und IPv4 (AF_INET) umgehen
- getaddrinfo statt gethostbyname
- inet_pton löst im Gegensatz zu inet_aton keine Adressen auf
- Grob abwärtskompatibel zu Socket/IO::Socket::INET, kann auch IPv4

```
IO::Socket::INET6->new( 'ipv6.google.com' ); # IP6
IO::Socket::INET6->new( 'www.google.com' );  # IP4
```

- aktiv geplegt, stabil, aber immermal auch kleinere Bugfixes

SSL und IPv6

- ich bin Maintainer von IO::Socket::SSL
- bisher: entweder von IO::Socket::INET oder von IO::Socket::INET6 abgeleitet
  - use IO::Socket::SSL
  - use IO::Socket::SSL 'inet6'
- Sowas führt zu Problemen, da die Entscheidung bzgl IPv6 dann vom aufrufenden Programm/Modul getroffen werden muss: "Perl considered harmful"
- seit v1.17 (10/08): wenn IO::Socket::INET6 wird es per Default genutzt
- evtl. Probleme, wenn AAAA Record geliefert, aber keine IPv6 Verbindung möglich

Net::DNS

- kann mit AAAA Records umgehen
- kann auch IPv6 Resolver ansprechen
- funktioniert i.A. ohne Probleme
- kleinere Bugs durch nicht eindeutige Textdarstellung von IPv6 beim Erstellen und Verarbeiten eigener Net::DNS::Packet's

Net::SMTP, Net::POP3,...

- direkt von IO::Socket::INET abgeleitet
- funktionieren also nicht mit IPv6
- mögliche Lösung: @ISA hinterrücks anpassen:

```
@Net::SMTP::ISA =
  qw(Net::Cmd IO::Socket::INET6)
```

- Alternative: INET.pm auf INET6.pm patchen

```
use Net::INET6Glue::INET_is_INET6
```

Net::INET6Glue::INET_is_INET6

- macht IO::Socket::INET == IO::Socket::INET6
- durch Kopie der Symboltabelle
- Module, die direkt IO::Socket::INET Objekte erstellen (LWP) oder davon abgeleitet sind (Net::SMTP) werden damit IPv6 fähig...
- ...sofern sie nicht versuchen die (IPv4-)Adressen zu interpretieren oder tief in die Innereien von IO::Socket::INET einzugreifen

LWP

- benutzt für http direkt IO::Socket::INET
- mit Net::INET6Glue::INET_is_INET6 kann es auch IPv6

```
use Net::INET6Glue::INET_is_INET6;
use LWP::Simple;
print get( 'ipv6.google.com' ); # IPv6
print get( 'www.google.com' );  # IPv4
```

- für https wird *vielleicht* IO::Socket::SSL genutzt, dort wird IPv6 unterstützt.
- Force IO::Socket::SSL:

```
use Net::SSLGlue::LWP
```

- ftp via Net::FTP (siehe später)

Net::FTP

- ist ebenfalls von IO::Socket::INET abgeleitet
- Net::INET6Glue::INET_is_INET6 kümmert sich nur um Kontrollverbindung und Datenverbindung
- IP:Port der Datenverbindung werden bei IPv6 mittels EPTR und EPSV statt PORT und PASV ausgehandelt
- diese Kommandos werden von Net::FTP nicht unterstützt

Net::INET6Glue::FTP

- lädt Net::INET6Glue::INET_is_INET6
- fügt EPRT und EPSV Kommandos zu Net::FTP hinzu
- patched pasv() und port() sodaß bei IPv6 Verbindungen EPSV und EPRT genutzt werden
- patched _dataconn(), sodaß die von EPSV gewonnenen Infos genutzt werden
- und damit geht es dann

```
use Net::INET6Glue::FTP;
my $ftp = Net::FTP->new( '[::1]:1121' );
```

- fragiles Vorgehen, daher fest verdrahtet auf bestimmte Version von Net::FTP

Arbeiten mit der Textdarstellung

Gültigkeit

- bei IPv4 kam man noch mit einer Regex für den Syntaxcheck davon, bei IPv6 nicht mehr.

```
$valid_ip6 = eval { Socket6::inet_pton(...) }
```

Vergleich

- Stringvergleich reicht nicht, da IPv6 auch eine verkürzte Representation zuläßt.

```
$equal = inet_pton($a) eq inet_pton($b)
```

IP:Port

- IPv6 enthält Doppelpunkte, d.h das einfache split() am Doppelpunkt funktioniert nicht mehr. Stattdessen muss man komplexere Syntax unterstützen:
  - ip4
  - ip4:port
  - ip6
  - [ip6]:port
- nicht trivial, siehe `IO::Socket::INET6::_sock_info`
- kein Modul bekannt, was einem diese Aufgabe abnimmt

Module für Textdarstellung

- NetAddr::IP bietet viele Funktionen zur Arbeit mit IPv4 und IPv6 Adressen
  - Vergleich incl. Netzmaske
  - Rechnen mit Adressen
  - Umwandlung zwischen Text und Binärdarstellung
  - liefert Broadcast Adresse, Netzwerkmaske
  - ...
  - arbeitet mit Operator overloading :(
- Net::IP auch, aber scheint nicht mehr gepflegt zu werden (letzte Version 2006)
  - weniger Funktionalität als NetAddr::IP

weitere Module

- AnyEvent::Socket
  - scheint mit IPv6 im Hinterkopf entwickelt zu sein und behandelt es transparent
  - aber mit welchem Aufwand: kein Socket6 und viel Aufwand um packing für sockaddr_in und Konstante AF_INET6 für jedes Betriebssystem zu ermitteln
- POE
  - scheint ebenfalls IPv6 zu supporten. Viel nahezu gleicher Code für IPv4 und IPv6 um Unterschiede im Interface (inet_aton vs. inet_pton, gethostbyname vs. getaddrinfo etc) zu umgehen
- Danga::Socket (Basis Perlbal)
  - scheint einige Gedanken an IPv6 verwendet zu haben
  - aber zumindest local_ip_string() verwendet explizit inet_ntoa(), welches auf IPv6 nicht geht

weitere Module

- Net::SNMP unterstützt IPv4 und IPv6
  - man muss aber wohl explizit übergeben, wenn man IPv6 benutzen will (keine automatische Bestimmung anhand der Adresse)
- Net::SIP
  - muss IPv6 Adressen in Textform innnerhalb der SDP Pakete verarbeiten
  - Code ist dafür da, aber nicht getestet
- Mail::SPF, Mail::SPF::Iterator
  - SPF Standard beinhaltet IPv6 Unterstützung
  - dank der guten SPF Testsuite sollten beide Module das können

Zusammenfassung

- kein Support für IPv6 im Perl Core
- aber es ist möglich mit IPv6 in Perl zu arbeiten
- wobei komfortabel wird das nur mit unschönen Hacks (mit evtl. unerwünschten Nebenwirkungen)
- ansonsten muss man überall explizit checken ob IPv6 Module verfügbar und diese bei Bedarf explizit nutzen
- besser wäre nahezu transparenter IPv6 Support im CORE, analog zu Python...

weitere Resourcen

- PerlMonks Diskussion zum Forcieren von INET6 in vorhandene Module
- "Living with IPv6" Blog
- IPv6 support in Squid erst in squid3.1 (d.h. noch kein IPv6 in aktuellem Squid Web Proxy):
- Überblick über IPv6 in Samba

Danke!