IPv6 Labor

IPv6 Gateways (TRT)

Stephan Uhlmann <su@su2.info>
02.10.2002

1 Motivation

In der Übergangsphase von IPv4 zu IPv6 wird es den Bedarf geben langsam zu migrieren. Netzwerke die schon mit IPv6 arbeiten möchten weiterhin mit Hosts kommunizieren, die noch mit IPv4 funktionieren und andersrum. Nicht alle Hosts sind Dual-Stack fähig oder sollen es sein, da z.B. IP-Adressen für beide Protokoll benötigt werden. Der beste Ansatzpunkt sind Router, denn so bieten diese Gateways eine Möglichkeit, transparent für ganze Subnetze eine Übersetzung ins jeweils andere Protokoll vorzunehmen. Es gibt verschiedene Modelle für solche Gateways. Speziell soll es hier um TRT gehen, welches sich besonders für die Anbindung von reinen IPv6 Hosts an IPv4 Netze eignet.

2 Modelle

2.1 SOCKS64

SOCKS64 ist eine Erweiterung des schon bestehenden SOCKS Proxy Protokolls. Ein solches Gateway nimmt IPv4 Verbindungen an und leitet diese wahlweise als IPv4 oder IPv6 weiter. Vorteil ist, dass keine Modifikationen oder "Hacks" am DNS gemacht werden müssen wie bei einigen der anderen Lösungen. Nachteil ist, dass die Client Applikationen SOCKS-fähig (socksified) sein müssen. SOCKS64 macht daher eher in solchen schon schon bestehenden Umgebungen Sinn.

SOCKS64 ist in RFC 3089 (`http://www.ietf.org/rfc/rfc3089.txt`) beschrieben.

2.2 Application Layer Gateways (ALGs)

Bekannt aus heutigen Sicherheitslösungen sind sogenannte Application Layer Gateways. Dies sind Hosts auf denen für ein jeweiliges Protokoll eine Proxy-Software installiert ist. Anstatt direkt mit dem Server zu kommunizieren, werden Anfragen dieses Protokolls an den Proxy gestellt, welcher diese dann für den Client ausführt und entsprechend weiterleitet. Ist dieser Proxy-Host mit einem Dual-Stack ausgerüstet, dann können die Anfragen sowohl in IPv4 als auch in IPv6 weitergeleitet werden. Vorteil ist auch hier, dass keine besonderen Veränderungen am DNS notwendig sind. Nachteil ist, dass alle Clients umkonfiguriert werden müssen, wenn nicht schon vorher die Proxies benutzt wurden. Auch gibt es nicht für alle Protokolle entsprechende ALGs.

2.3 SIIT

Der Stateless IP/ICMP Translation Algorithm (SIIT) ist in RFC 2765 (`http://www.ietf.org/rfc/rfc2765.txt`) dargelegt und beschreibt eine allgemeine Methode zur Übersetzung von IPv4 und IPv6 Paketen in beide Richtungen. Durch die teilweise sehr unterschiedlichen Felder im IP-Header werden Probleme in der Übersetzung aufgeworfen, die dieser Algorithmus versucht möglichst elegant zu lösen. Informationsverlust, gerade bei der Übersetzung von IPv6 nach IPv4, kann jedoch nicht vermieden werden. SIIT ist die Grundlage für NAT-PT.

2.4 NAT-PT

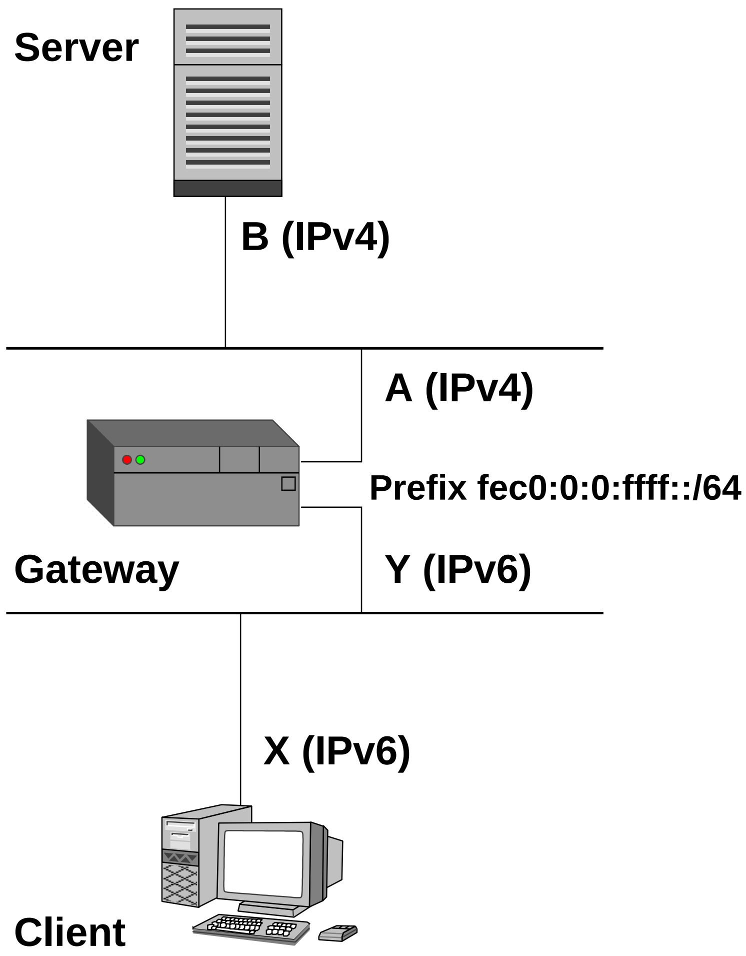
(detailreicher siehe Vortrag von Andreas Liebert)

NAT-PT bedeutet Network Address Translation / Protocol Translation. Es arbeitet wie die schon bekannten IPv4 NATs, fügt dem jedoch eine IPv4-IPv6 Protokollübersetzung auf Basis des SIIT hinzu. Ein Vorteil von NAT-PT ist, dass es sowohl IPv6 nach IPv4 als auch IPv4 nach IPv6 Kommunikation beherrscht. NAT-PT ist in RFC 2766 (`http://www.ietf.org/rfc/rfc2766.txt`) beschrieben. Eine Implementation ist "nat-pt" des Electronics and Telecommunications Research Institute, Korea.

2.5 TRT

TRT ist grundsätzlich nur dafür gedacht, Kommunikation von IPv6 nach IPv4 zu unterstützen. Theoretisch wäre auch die Richtung IPv4 nach IPv6 möglich, jedoch ist dies generell sehr viel schwieriger, da DNS-Anfragen auf temporäre IPv4 Adressen abgebildet werden müssen. In der Praxis wird wohl auch viel eher die Anforderung bestehen, neue IPv6-only Hosts an das bestehende IPv4 Internet anzuschliessen.

Wie auch bei NAT-PT werden IPv4 Adressen durch ein DNS ALG in IPv6 Adressen mit einem sogenannten "dummy Präfix" umgewandelt. Jede Kommunikation mit Adressen dieses Präfix wird vom TRT abgefangen. Das TRT terminiert die Verbindung in Richtung Client, baut eine neue IPv4-Verbindung zum eigentlichen Ziel auf und leitet die Daten in beiden Richtungen entsprechend weiter.


Server
B (IPv4)
A (IPv4)
Prefix fec0:0:0:ffff::/64
Gateway
Y (IPv6)
X (IPv6)
Client

Die Umsetzung findet also auf der Transportschicht statt und nicht wie bei NAT-PT auf der Netzwerkschicht. Dies hat eine Reihe von Vorteilen. Dadurch, dass TRT nicht auf SIIT angewiesen ist, gibt es weniger Probleme bei der Path MTU discovery / Fragmentation, denn die IPv4 und IPv6 Teile können getrennt betrachtet werden. Ein TRT-System ist wesentlich einfacher zu implementieren und zu verstehen. Es werden zwar hier auch ALGs für NAT-unfreundliche Protokolle, die IP-Adressen in höheren Schichten transportieren, wie z.B.. FTP und DNS, benötigt, theoretisch könnte man diese aber auch gleich innerhalb des TRT implementieren.

Modifikationen an den Hosts sind nicht notwendig.

Dadurch, dass TRT "stateful" ist muss die gesamte Kommunikation einer Verbindung über das gleiche TRT gehen. Dies könnte in der Praxis einen Single Point of Failure bilden. Wie beim SIIT gibt es auch hier einen Informationsverlust beim Übergang von IPv6 zu IPv4, so dass nicht alle Eigenschaften von IPv6 bei der Kommunikation zu IPv4 Hosts zur Verfügung stehen. Dies betrifft z.B. auch IPSec. Desweiteren werden nur bidirektionale Verbindungen unterstützt und keine unidirektionalen wie z.B. Multicast.

TRT ist im RFC 3142 (`http://www.ietf.org/rfc/rfc3142.txt`) beschrieben.

Implementationen sind "faith" des KAME Projekts, welches jedoch nur unter BSD läuft und "pTRTd" (Portable Transport Relay Translator Daemon) welches auch unter Linux läuft.

3 TRT in der Praxis

3.1 pTRTd

pTRTd ist eine portable TRT-Implementation, da es seinen eigenen IPv6 Stack mitbringt und nicht auf Änderungen des IPv6 Stacks im Betriebssystem des

Hosts angewiesen ist. In der default Einstellung benutzt es den Präfix fec0:0:0:ffff::/64, dies kann jedoch mit der Option -p geändert werden. Als externe Abhängigkeit benötigt pTRTd nur den TUN/TAP Gerätetreiber. Dieser lässt sich einfach mit den folgenden zwei Befehlen aktivieren.

```
# mknod /dev/net/tun c 10 200
# modprobe tun
```

Danach kann ptrtd problemlos gestartet werden.

3.2 totd

Der totd (Trick Or Treat Daemon) dient als DNS-Proxy. Er leitet alle Anfragen an ihn an einen anderen DNS-Server (Forwarder) weiter. Bevor er jedoch eine Antwort an den Client zurückgibt, wandelt er IPv4-Adressen in "fake" IPv6-Adressen um. Dazu benutzt er einen konfigurierbaren Präfix, welcher natürlich der gleiche sein sollte, wie er vom TRT verwendet wird. Eine Anfrage an den auf "localhost" (::1) auf Port 5005 laufenden Server sieht z.B. so aus:

```
# dig @::1 -p 5005 www.google.com aaaa
(...)
;; ANSWER SECTION:
www.google.com. 122 IN AAAA fec0::ffff:0:0:d8ef:2365
(...)
```

3.3 Erfahrungsbericht

Der totd funktioniert problemlos.

pTRTd ist noch im Alpha-Stadium, arbeitet aber in seiner Grundfunktionalität zufriedenstellend. Ein Problem das auftauchte war, dass der default Präfix (fec0:0:0:ffff::/64) nicht funktionierte:

```
# ping6 fec0::ffff:0:0:d8ef:2365
connect: Cannot assign requested address
```

Der `connect()` Systemaufruf lieferte einen `EADDRNOTAVAIL` Fehler. Das Problem konnte aber durch Verwendung eines global scope Präfix behoben werden. Hier der Mitschnitt, wie ich mich per SSH auf den IPv4 Host lisa.cs.uni-potsdam.de einlogge:

```
# ./totd -c totd.conf
# ./ptrtd -p 2001:7a0:101:888::
# ping6 2001:7a0:101:888::1 (zurückgepingt von ptrtd)
# dig @::1 -p 5005 lisa.cs.uni-potsdam.de aaaa

; ANSWER SECTION:
lisa.cs.uni-potsdam.de. 172800 IN AAAA 2001:7a0:101:888::8d59:30de

# ssh -6 -l uhlmann 2001:7a0:101:888::8d59:30de
uhlmann@lisa:~ >
```

4 Fazit

Ich kann TRT als Lösung für die Anbindung von IPv6-only Hosts an das IPv4 Internet empfehlen. Es geht das Problem auf einfache und damit leicht verständliche und implementierbare Weise an. Es hat für seinen Zweck eine gute Balance zwischen Vorteilen und Nachteilen. Die Nachteile sind in der Praxis für die Mehrheit der Anwender weniger relevant.

Die Implementation pTRTd im Zusammenspiel mit totd ist lauffähig, müsste jedoch vor einem Produktiv-Einsatz weiter getestet werden.

5 Links

- pTRTd: `http://www.litech.org/ptrtd/`
- totd: `http://www.vermicelli.pasta.cs.uit.no/ipv6/software.html`
- Modellvergleich IPv6-IPv4-Gateways: `http://web.mit.edu/peilei/www/nats2.htm`