Selbstorganisierende Ad-hoc-Netze mit „babeld“

Beitrag von „Sys_RoBOTer“ vom 8. Januar 2021, 08:09

> Zitat von Hamspirit.de
>
> Beim Vernetzen von Computern via Funk kann sich die Anzahl und Auswahl der direkt erreichbaren Gegenstellen im Verlaufe der Zeit ändern: Neue Knoten können hinzukommen, bestehende entfallen oder vorübergehend unerreichbar sein. Zeitgemäße Netzwerk-Protokolle, z.B. das Internet-Protokoll (IP), in Verbindung mit geeigneten Routing-Protokollen, z.B. Babel, können die Erreichbarkeit aller Knotenpunkte eines vermaschten Funknetzes (und ggf. weiterer angeschlossener Teilnetze) ermöglichen.
>
> Insbesondere für uns Funkamateure ist ein solcher Ansatz deshalb interessant, weil einige Stationen nicht dauerhaft zur Verfügung stehen (z.B. weil sie nur besetzt betrieben werden), sich die Anzahl von Knotenpunkten jederzeit ändern kann und Überreichweiten oder experimentelle Aufbauten bestimmte Teilnetze verbinden könnten, die ansonsten isoliert voneinander sind.
>
> In diesem Artikel möchte ich einen Lösungsansatz zur IP-Adressvergabe und Vernetzung via IP-Routing vorstellen, der
>
> - keine zentrale Planungsstelle benötigt,
> - auf dem existierenden IPv6-Standard aufsetzt bzw. mit dem in Standardisierung befindlichen Babel-Protokoll arbeitet,
> - Open-Source-Software verwendet, die derzeit unter den Betriebssystemen Linux und FreeBSD verfügbar ist,
> - die Eigenarten des Mediums „Funk“ berücksichtigt und
> - einfach einzurichten ist.
>
> Die das Netz aufspannenden Stationen sind hierbei in einem vermaschten Netzwerk (Mesh-Netz) miteinander verbunden. Wie die dazu notwendigen Direktverbindungen aufgebaut werden können, soll nicht Teil dieses Artikels sein, wird jedoch z.B. von DL1PZ in einem anderen Beitrag behandelt. Stattdessen beschäftige ich mich hier mit der Einrichtung des Routings, so dass eine Funkstation A, die Kontakt mit einer Funkstation B hat, auch die Funkstation C erreichen kann, falls die Stationen B und C eine Verbindung haben. Auch eine Weiterleitung über mehr als eine Zwischenstation (genannt: „Hop“) ist so möglich.

Dezentrale IP-Adressvergabe

In einem IP-Netzwerk werden die Teilnehmer mittels sogenannter IP-Adressen angesprochen. Im Falle von IPv4, der älteren Version des Internet-Protokolls, sind dies vier Zahlen von 0 bis 255 getrennt durch Punkte, also z.B. „*10.169.145.125*“. Bei IPv6 kommen längere Adressen zum Einsatz, die in hexadezimaler Schreibweise notiert werden, z.B. „*fdb7:5c3c:1e90:e4c1:0000:0000:00d5:85ed*“. Die Punkte bzw. Doppelpunkte dienen nur der Lesbarkeit bei Darstellung in Textform und werden bei einer binären Übertragung nicht mitgesendet. Zur Vereinfachung besteht bei IPv6 die Möglichkeit, mittels eines doppelten Doppelpunktes „*::*“ die Auslassung mehrerer 0000-Blöcke zu kennzeichnen. Außerdem ist es möglich, führende Nullen innerhalb eines Vier-Zeichen-Blockes wegzulassen, so dass vorstehende Adresse auch verkürzt als „*fdb7:5c3c:1e90:e4c1::d5:85ed*“ geschrieben werden könnte.

Um eine zentrale Adressvergabe zu vermeiden, aber gleichzeitig jedem Teilnetz (einschließlich vorübergehend isolierter Netze) permanente Adressen zuweisen zu können, die dennoch mit hoher Wahrscheinlichkeit im Rahmen des Einsatzzweckes eindeutig sind, wird IPv6 genutzt. IPv6 ermöglicht es aufgrund der vergleichsweise langen Adressen, einfach einen zufälligen Adressbereich zu wählen, der dann mit großer Wahrscheinlichkeit eindeutig ist. Genau dies ist bei IPv6 im Rahmen der sogenannten „Unique Local Addresses“ (ULA) auch vorgesehen: Jede Person oder Organisation, die einen Adressbereich benötigt, erzeugt einfach eine 40 Bit lange Zufallsfolge. Wird dieser noch die Bitfolge *11111101* (hexadezimal „*fd*“) vorangestellt, ergeben sich 48 Bit, die den sogenannten „Präfix“ für den Adressbereich bilden. Die Person bzw. Organisation, die sich diesen Adressbereich zufällig ausgesucht hat, kann nun alle damit beginnenden Adressen beliebig verwenden, ohne große Gefahr zu laufen, mit den Adressen zu kollidieren, die andere Personen oder Organisationen verwenden. Ein solcher Präfix wird z.B. als *fd4a:eeb2:7cea::/48* notiert, wobei 48 die Präfixlänge in Bits in dezimaler Schreibweise darstellt.

Die Wahrscheinlichkeit einer Kollision, d.h. die Wahrscheinlichkeit, dass in einem Netzwerk der gleiche Präfix zweimal vergeben wird, steigt zwar vergleichsweise schnell mit der Anzahl der vergebenen Präfixe (vgl. Geburtstagsparadoxon), allerdings beträgt z.B. bei 10.000 solcher miteinander vernetzten Präfixe die Kollisionswahrscheinlichkeit aufgrund der vielen Möglichkeiten (2 hoch 40 = 1.099.511.627.776) immer noch weniger als 0,05 Promille, also etwa 1:20.000. Verglichen mit anderen potentiellen Störungsursachen scheint dies vernachlässigbar klein zu sein. Die Berechnungsformeln für die Kollisionswahrscheinlichkeit sind identisch mit denen der sogenannten Geburtstagsangriffe, wobei wir hier vom wohlwollenden Verhalten aller Teilnehmer ausgehen.

Sobald ein 48-Bit langer IPv6-Adress-Präfix per Zufall ermittelt wurde, können, wie zuvor erwähnt, aus dem sich ergebenden Bereich beliebige IP-Adressen gewählt werden, z.B. die Adresse

fd4a:eeb2:7cea:0000:0000:0000:0000:0001

welche in verkürzter Schreibweise als *fd4a:eeb2:7cea::1* geschrieben werden kann. Auch ist es möglich ganze Netzbereiche unterzuverteilen.

Unter Linux lässt sich dann beispielsweise diese Adresse einem Funk-Interface (hier im Beispiel *tun0*) wie folgt zuweisen:

Code

```
ip -6 addr add fd4a:eeb2:7cea::1/128 dev tun0
```

Die *128* steht hierbei für eine Präfixlänge von 128 Bit, d.h. wir teilen dem Betriebssystem mit, dass zunächst keine andere Adresse via Funk erreichbar ist. Welche anderen Adressen wir tatsächlich erreichen können, wird dem Betriebssystem vom Routing-Dienst mitgeteilt, der im Folgenden beschrieben werden soll.

Der Routing-Dienst „babeld“

Sofern die Installation von „babeld“ in aktueller Version nicht durch das Paketmanagement des Betriebssystems erfolgen kann, finden sich Hinweise zum Download der Software auf der Homepage des Babel-Projektes.

Die folgenden Beschreibungen beziehen sich auf die zum Zeitpunkt der Veröffentlichung dieses Artikels aktuelle Version 1.9.2 der Software. Es gibt auch andere Routing-Software, z.B. BIRD, die das Babel-Protokoll ebenfalls umsetzen kann. Diese sollte ebenso eingesetzt werden können, ist aber anders zu konfigurieren.

Die Einrichtung von „babeld“ gestaltet sich sehr einfach. Wir legen dazu die Konfigurationsdatei */etc/babeld.conf* an. Für den Fall, dass wir unseren Präfix wie im vorhergehenden Abschnitt als *fd4a:eeb2:7cea::/48* festgelegt hätten (in der Praxis muss dies natürlich ein eigener, zufälliger Präfix sein), sähe die Datei wie folgt aus:

Code

```
in                    ip                    fd00::/8                    allow
in                                                                      deny
out                    ip                    fd00::/8                   allow
out                                                                     deny
redistribute             ip             fd4a:eeb2:7cea::/48             local
redistribute         ip         fd4a:eeb2:7cea::/48         metric         256
redistribute                              local                         deny
redistribute deny
```

Den Routing-Dienst können wir dann z.B. mit folgendem Kommando starten:

Code

```
babeld -c /etc/babeld.conf wl3sp0
```

Hierbei wäre *wl3sp0* das Funk-Interface, mit dem die Vernetzung erfolgt. (Mit „*ip link show*“ lassen sich alle verfügbaren Interfaces anzeigen.) Wird, wie in diesem Beispiel, ein WLAN-Interface (*wl3sp0*) verwendet, kann babeld anhand der Hardwarekonfiguration selbst erkennen, dass eine Funkverbindung vorliegt. Wird stattdessen eine Vernetzung mittels einer schmalbandigen Verbindung über ein herkömmliches Funkgerät vorgenommen, kommt oft ein generischer TUN-Gerätetreiber zum Einsatz, bei dem die automatische Erkennung nicht möglich ist. Hier sollten wir nachhelfen, z.B. mit:

Code

```
babeld -c /etc/babeld.conf -C "interface tun0 type wireless channel interfering hello-inte
```

So sagen wir dem Routing-Dienst, dass es sich bei der verwendeten Netzwerkschnittstelle um eine Funkverbindung handelt („*type wireless*“), die sich ggf. mit anderen Funkverbindungen einen Kanal teilt („*channel interfering*“) und das Kommunikationsintervall für eine schonendere Bandbelegung reduziert wird („*hello-interval 60*“). Anstelle der Option „*-C*“ lässt sich der auf diese Option folgende Text („*interface tun0* […]“) auch als eigene Zeile in die Konfigurationsdatei schreiben. (Details hierzu in der Anleitung von babeld.)

Falls wir Pakete für andere Funkknoten weiterleiten wollen (was ja Sinn und Zweck eines Mesh-Netzes ist), müssen wir unserem Betriebssystem noch mitteilen, dass Packet-Forwarding (also Routing) aktiviert werden soll. Dies lässt sich z.B. für IPv6 unter Linux bis zum nächsten Neustart mit folgendem Kommando einschalten:

Code

```
sysctl -w net.ipv6.conf.all.forwarding=1
```

Soll Packet-Forwarding dauerhaft aktiviert bleiben, so ist die Datei */etc/sysctl.conf* entsprechend zu bearbeiten.

Sicherheitsbedenken

Sobald wir Routing auf unserem Computer aktivieren, wird dieser beliebige Pakete weiterleiten. Es wäre dann also auch möglich, via Amateurfunk auf ein ebenfalls angeschlossenes privates Heim- oder Firmennetz (oder das Internet) zuzugreifen, was

in der Regel nicht erwünscht sein wird. In einem solchen Fall ist also unbedingt eine Firewall einzurichten, deren Konfiguration hier nicht weiter behandelt werden soll und vom konkreten Szenario abhängt.

Ebenso sollte verhindert werden, dass der Routing-Dienst über Funk Einträge für unser privates Netz akzeptiert. Liegt unser privates Netz außerhalb des ULA-Adressbereiches (beginnt also nicht mit hexadezimal „*fd*“), dann sorgt die oben vorgestellte Konfiguration bereits automatisch dafür, dass entsprechende Routing-Informationen abgelehnt werden. Verwenden wir bei unserem privaten Heim- oder Firmennetz jedoch ebenfalls ULAs, dann müssen wir dem Routing-Dienst noch mitteilen, dass für diese Adressen keine Informationen aus dem Funknetz angenommen werden sollen. Dies lässt sich, z.B. wenn *fdf2:c215:20a4::/48* unser privates Netz sei, mit der Zeile

Code

```
in ip fdf2:c215:20a4::/48 deny
```

an oberster Stelle in der */etc/babeld.conf* erreichen.

Einen zusätzlichen Filter für die ausgehende Richtung benötigen wir nicht. Wir haben bereits vermerkt, für welche Netzwerke wir Routing-Informationen einspeisen wollen („*redistribute ip fd4a:eeb2:7cea::/48* […]“). Für alle anderen Netzwerke haben wir festgelegt, dass diese von babeld nicht eingespeist werden („*redistribute local deny*“ und „*redistribute deny*“).

Anbindung von weiteren angeschlossenen Geräten über den selben Funkknoten

Der Zugriff auf bestimmte lokale IP-Adressbereiche kann aber durchaus auch erwünscht sein, denn vielleicht möchten wir nicht nur ein einzelnes Gerät an das Funknetz anbinden, sondern ein ganzes Netzwerk. Die bereits erwähnte Konfigurationszeile „*redistribute ip* […] *metric 256*“ (ohne die Option „*local*“) ermöglicht die Einspeisung von IP-Adressen, die nicht dem eigenen Endgerät, sondern anderen Endgeräten zugehörig sind.

Die Zahl *256* gibt hier die sogenannte Metrik an, die ein Maß für die Güte der Anbindung eines Netzwerks ist. Dieser Wert ist dann von Bedeutung, wenn ein Netzwerk auf verschiedenen Wegen erreichbar ist, z.B. weil es mit mehreren Funkknoten direkt verbunden wird. Dann kann bei jedem dieser Funkknoten ein unterschiedlicher Wert für dieses Netzwerk festgelegt werden, wobei ein niedrigerer Wert für eine höhere Güte der Anbindung steht. In einfachen Fällen, also bei Netzwerken ohne Mehrfachanbindung, können wir einfach *256* festlegen.

Für die lokale Vernetzung können wir nun einen Unterbereich des Adressraumes ausweisen, z.B. *fd4a:eeb2:7cea:5555::/64*. Unserem eigenen Netzwerk-Interface (also z.B. unserer Netzwerkkarte) weisen wir eine IP aus diesem Unteradressraum zu und geben mit der Präfixlänge */64* an, dass weitere Adressen aus dem gleichen Unteradressraum (beginnend mit den 64 Bits *fd4a:eeb2:7cea:5555*) direkt zu erreichen sind. Heißt unser Netzwerk-Interface für kabelgebundenes Ethernet z.B. „*enp0s25*“, so wäre das Kommando hierfür:

Code

```
ip -6 addr add fd4a:eeb2:7cea:5555::1/64 dev enp0s25
```

Andere Geräte am gleichen Ethernet-Segment können dann z.B. die Adressen mit der *2*, *3*, *4* usw. am Ende zugewiesen bekommen, also z.B. die *fd4a:eeb2:7cea:5555::4*, und sind dann aus dem Funk-Mesh-Netz erreichbar. Diesen Geräten muss jetzt nur noch mitgeteilt werden, dass Pakete an *fd00::/8* (also Adressen beginnend mit „*fd*“) über unseren Funkknoten mit der IP *fd4a:eeb2:7cea:5555::1* geroutet werden müssen, sofern für diese keine andere Route existiert. Hierzu ist das Routing der Geräte entsprechend einzustellen oder ggf. eine Default-Route zu setzen. Wie dies funktioniert, hängt vom Betriebssystem des jeweiligen Endgerätes ab. Auch ließe sich ein DHCPv6-Server verwenden, dessen Konfiguration jedoch den Rahmen dieses Artikels sprengen würde.

IPv6 und die Paketgröße (MTU)

Leider sieht IPv6 vor, dass auf jeder Verbindung mindestens Pakete der Größe 1280 Byte übertragen werden können. Dies ist die minimal mögliche Einstellung der sogenannten „Maximum Transmission Unit“ (MTU). Eine Fragmentierung, d.h. Aufteilung in kleinere Pakete, ist durch Router auf dem Transportweg bei IPv6 im Gegensatz zu IPv4 nicht vorgesehen (siehe RFC 8200, Sektion 5). Das kann zu Schwierigkeiten z.B. bei bestimmten Hardware-TNCs führen, die Pakete dieser Größe nicht handhaben können. Umgehen ließe sich dieses Problem mit einer zusätzlichen Fragmentierungsschicht, die aus Sicht von IPv6 unsichtbar sein müsste. Die von DL1PZ vorgestellten Programme ermöglichen keine zusätzliche Fragmentierung, jedoch sind die dort vorgestellten Software-TNCs Dire Wolf und pktfec-tnc ohne Probleme in der Lage, größere Pakete auf’s Band zu senden.

Skalierbarkeit

Da jeder Knoten in regelmäßigen Abständen Informationen zu allen erreichbaren Adressbereichen mit allen direkten Nachbarknoten austauschen muss, skaliert das hier vorgestellte Verfahren nicht beliebig. Mit wachsender Knotenanzahl wird die Menge an zu übertragenden Informationen ebenfalls wachsen. Bei langsamen Packet-Radio-

Verbindungen dürften bereits bei einer zweistelligen Anzahl von Netzknoten Probleme auftreten. Weniger problematisch wäre dies bei einer Vernetzung mit schnellerer Technik.

Um dieses Problem zu lösen bzw. zu umgehen, sehe ich verschiedene Alternativen:

- Mesh-Netze bleiben lokal begrenzt und werden mit anderen Mesh-Netzen ausschließlich manuell verbunden. Routing-Informationen zu anderen Mesh-Netzen werden nur für das gesamte jeweils andere Mesh-Netz in das eigene Mesh-Netz eingespeist, so dass jedes andere Mesh-Netz nur mit einem einzigen Präfix in den Routing-Tabellen auftaucht, was die Routing-Tabellen kleiner hält. Damit dies funktioniert, müssten die einzelnen lokalen Mesh-Netze jedoch jeweils einen gemeinsamen Präfix verwenden und sich dann auf eine Vergabemethode von IP-Adressen innerhalb ihres jeweiligen Präfixes einigen.
- Mesh-Netze bleiben lokal begrenzt und werden untereinander über sogenannte Border-Router und ein Backbone verbunden. Routing-Informationen zu anderen Mesh-Netzen werden gar nicht im eigenen Mesh-Netz verbreitet. Stattdessen werden die Border-Router als zuständig für den Bereich *fd00::/8* angegeben.
- Das Routing-Protokoll wird erweitert (bei Babel grundsätzlich vorgesehen), oder ein anderes, zukünftiges Routing-Protokoll wird benutzt, das solche Szenarien noch besser abbilden kann.

Fazit und Ausblick

Es macht Spaß, mit Ad-hoc-Vernetzung zu experimentieren. IPv6 ermöglicht es, ohne zentrale Koordinierung feste IP-Adressen zu verwenden. Mit dem Routing-Protokoll *Babel* bzw. der Software „*babeld*“ und IP-Packet-Forwarding können wir eine automatische Vernetzung vornehmen lassen. Inwieweit sich diese Verfahren im größeren Maßstab in der Praxis, insbesondere bei langsamen Packet-Radio-Verbindungen, einsetzen lassen, bleibt zu erforschen.

Alles anzeigen

Quelle: https://www.hamspirit.de/12299…-ad-hoc-netze-mit-babeld/