IP-Vernetzung per Funkgerät

Beitrag von „Sys_RoBOTer“ vom 7. Januar 2021, 12:45

> Zitat von Hamspirit.de
>
> Wenn im Zusammenhang mit dem Amateurfunk von IP-Vernetzung die Rede ist, werden viele sicherlich zuerst an das HAMNET denken, ein Hochgeschwindigkeitsnetzwerk auf IP-Basis. Doch darum soll es in diesem Artikel gar nicht gehen. Denn obwohl die Möglichkeiten des HAMNET, insbesondere die verfügbaren Datenraten, überzeugend sind, gibt es doch einige praktische Einschränkungen. Die vermutlich größte Hürde in das HAMNET einzusteigen sind die Ausbreitungseigenschaften des hierfür verwendeten 2,4 GHz-Bandes. Ohne direkten Sichtkontakt geht da nicht viel. Das heißt in der Praxis: Eine flächendeckende Versorgung mit HAMNET steht in ländlichen genauso wie städtischen Gebieten noch in den Sternen, obwohl bereits ein erheblicher Aufwand getrieben wurde, um sogenannte Benutzereinstiege bereitzustellen. Auch der neuerdings zusätzlich verfolgte Ansatz mit Breitbanddatenübertragungen auf dem 70cm-Band mittels NPR70 findet seine Grenzen in dem knappen zur Verfügung stehenden Frequenzspektrum, das sich viele Anwendungen teilen müssen.
>
> Deshalb haben Silvio DM9KS, Jan DL9JBE und Björn DL1PZ in den letzten zwei Jahren diverse Versuche zur IP-Vernetzung unternommen und dabei „normale“ Funkkanäle verwendet. Es wurden FM-Verbindungen mit 7 bzw. 12,5 kHz Kanalbreite auf dem 10m-, dem 2m- und dem 70cm-Band genutzt. Die dort erreichbaren Datenraten kommen natürlich nicht an die Geschwindigkeiten des HAMNET heran. Dafür sind aber IP-Verbindungen auch unter wesentlich widrigeren Funkbedingungen überhaupt herstellbar. In diesem Artikel sollen die verschiedenen Hard- und Softwarekomponenten bzw. Protokolle vorgestellt werden, die im Rahmen der Versuche erprobt wurden.
>
> [IMG: https://www.hamspirit.de/wp-content/uploads/2021/01/funkvernetzung-schema-page001.png]
>
> Schematischer Versuchsaufbau
>
> ## Versuchsaufbau
>
> Der grundsätzliche technische Aufbau folgte stets dem gleichen Schema. Eine oder mehrere IP-fähige Anwendungen (z. B. IRC-Chatprogramme, Telnet, Schachprogramme, usw.) erzeugen IP-Datenpakete. Diese werden durch einen Einpacker um ein Rufzeichen und Protokollinformationen ergänzt an einen Terminal Node Controller

(TNC) weitergegeben, der die Datenpakete für die Übertragung auf dem Band vorbereitet und dann über eine Audioschnittstelle an einen Transceiver übergibt. Empfangene Übertragungen werden andersherum durch den Transceiver über die Audioschnittstelle an den TNC zur Dekodierung übergeben. Dann werden Rufzeichen und Protokollinformationen durch den Auspacker entfernt und die empfangenen Nutzdaten als IP-Paket (über das Betriebssystem) an die IP-Anwendung weitergegeben.

In dem immer gleichen Schema war – neben der Verwendung von IP – die Konstante bei allen Versuchen das sogenannte KISS-Protokoll, auf das später noch eingegangen wird. Alle anderen Komponenten wurden in den Versuchen durch wechselnde Hard- bzw. Software implementiert.

AX.25

Der älteste Vertreter der Lösungen zur IP-Vernetzung über den Amateurfunk ist das AX.25-Protokoll. Dieses ursprünglich für Packet-Radio geschaffene Protokoll bringt auch die Möglichkeit des Transports von IP-Paketen mit. AX.25 ist ein umfassendes Netzwerkprotokoll, das selber auch ganz ohne IP-Netzwerk in der Lage ist, Computer und angebotene Dienste (wie Mailboxen) zu verbinden. Bis zum Ende des letzten Jahrtausends wurde es bei Funkamateuren rege genutzt und es bestand praktisch flächendeckender Zugang zum Packet-Radio-Netzwerk über Funk.

Das Protokoll enthält eine Vielzahl an Funktionen, z. B. zu Multiplexing, Flusskontrolle oder Routing, die für die IP-Vernetzung jedoch keine Rolle spielen. Diese Aspekte werden bereits von IP (bzw. genau genommen TCP/IP) selber gelöst. Daher beschränkt sich dieser Artikel auf die für den Anwendungsfall relevanten Teile von AX.25.

Im Folgenden wird nur der Sendeweg beschrieben. Der Empfangsweg verhält sich entsprechend genau andersherum. Das heißt die sendeseitig hinzugefügten Informationen werden in umgekehrter Reihenfolge wieder entfernt bzw. die vor der Aussendung vorgenommene Modulation wird entsprechend demoduliert.

AX.25 definiert gleich zwei der im schematischen Aufbau dargestellten Komponenten: Den Ein- und Auspacker für IP-Pakete sowie den TNC (aber nicht das Modem). Im AX.25-Protokoll ist diese Aufgabentrennung zwar überhaupt nicht vorgesehen, doch in der Praxis ist sie weit verbreitet. Auch das KISS-Protokoll ist nicht Teil von AX.25, doch dazu – wie schon versprochen – später mehr.

Bei den ersten Versuchen wurde die in Linux eingebaute AX.25-Unterstützung verwendet, um IP-Pakete ein- bzw. wieder auszupacken. Praktisch passiert dabei folgendes: Zu jedem IP-Paket, das auf die Funkschnittstelle gesendet werden soll, werden zunächst zusätzliche Informationen hinzugefügt, bevor das Paket an den TNC übergeben wird. Hierzu werden zwei der durch AX.25 spezifizierten Funktionalitäten

genutzt, nämlich die Adressierung und die Protokollidentifikation.

Rufzeichen und Protokollidentifikation

AX.25 verwendet Amateurfunkrufzeichen, um den Absender und den oder die Empfänger einer Übertragung anzugeben. Zur Unterscheidung mehrerer Stationen des gleichen Rufzeicheninhabers ist es vorgesehen, optional eine Zahl von 1 bis 15 als sogenannten „Secondary Station Identifier“ („SSID”) anzuhängen, z. B. DL1PZ-14. Die Kodierung des Rufzeichens erfolgt als 7-Bit-ASCII-Zeichenkette, wobei das jeweils freibleibende 8. Bit anderweitig genutzt wird. Die Erzeugung und Verarbeitung gestaltet sich daher etwas komplizierter, zudem die sieben ASCII-Bits auch noch um ein Bit zur Seite verschoben werden. AX.25 beschränkt die Länge der Rufzeichen strikt auf 6 Zeichen, was die Verwendung von längeren Rufzeichen (z. B. Sonderrufzeichen) leider vollkommen unmöglich macht. AX.25 sieht als Adressangabe mindestens eine Absender- und eine Zieladresse vor.

In der AX.25-Konfiguration von Linux wurde jeweils das eigene Rufzeichen hinterlegt. Dieses wird dann automatisch zu jedem Paket hinzugefügt. Da AX.25 zwingend auch eine Empfängeradresse vorschreibt, welche jedoch für die IP-Vernetzung nicht nötig ist, wird das entsprechende Feld durch Linux automatisch mit dem Text „QST“ belegt. Diese Q-Gruppe steht in CW für eine Sendung an alle Stationen.

Weiterhin setzt AX.25 noch einen sogenannten „Protocol identifier“ („PID“) vor die eigentlichen Nutzdaten. Im behandelten Falle, also der Übertragung von IP-Paketen, wird hier hexadezimal 0xCC für IPv4 vermerkt. Ein PID für IPv6 ist in der letzten Version 2.2 des AX.25-Standards nicht vorgesehen. Dementsprechend kann mit diesem Ansatz auch keine Vernetzung per IPv6 sondern nur per IPv4 vorgenommen werden.

[IMG: https://www.hamspirit.de/wp-content/uploads/2021/01/picoaprs_tnc-576x246.jpeg]

PicoAPRS – TNC und Transceiver in einem Gerät

KISS-Protokoll

Nachdem nun diese beiden Informationen hinzugefügt wurden, wird das Datenpaket an den TNC weitergegeben. Dazu wird das schon erwähnte KISS-Protokoll verwendet, meist über eine serielle Schnittstelle. Dieses ist mit dem SLIP-Protokoll verwandt, bei dem bestimmte Steuerzeichen einen seriellen Datenstrom in einzelne Pakete unterteilen. Die Abkürzung KISS bezieht sich hier auf das sogenannte KISS-Prinzip, Dinge einfach zu halten (engl. „Keep it simple, stupid!“, zu Deutsch also „Halte es einfach, Dummerchen!“ oder etwas sachlicher formuliert: „Mach‘ eine Sache nicht komplizierter als nötig“). Entsprechend ist das Protokoll auch überschaubar definiert.

Am Anfang und am Ende der Übertragung wird jeweils ein Byte mit dem hexadezimalen Wert 0xC0 als Begrenzung eines Datenpakets angefügt. Sofern in den Daten selber der Wert 0xC0 vorkommt, wird dieser durch die Bytefolge 0xDB 0xDC ersetzt. Wenn wiederum ein 0xDB vorkommt, wird dieses durch die Folge 0xDB 0xDD ersetzt.

TNC für AX.25

Die Aufgabe des TNC ist es nun, das Datenpaket für die Übertragung auf der Funkschnittstelle vorzubereiten. Ein für AX.25 ausgelegter TNC erledigt dabei zunächst die folgenden Aufgaben:

- Bitstopfen (engl. „bit stuffing“): Das Einfügen von einem oder mehreren Füll-Bits in einen Datenstrom. Im Falle von AX.25 konkret das Einfügen eines zusätzlichen 0-Bits, falls fünf 1-Bits unmittelbar aufeinander folgen.
- Rahmensynchronisation: Dient dem Erkennen von Anfang und Ende eines AX.25-Datenpakets (streng genommen gemäß OSI-Schichtmodell als „Datenframe“ zu bezeichnen). Konkret beginnt und endet jedes AX.25-Datenframe mit der Bit-Folge 01111110 (hexadezimal 0x7E). Daher auch das vorgenannte Bitstopfen, das sicherstellt, dass diese Sequenz niemals innerhalb der tatsächlich übertragenen Nutzdaten vorkommt, sondern nur als Rahmensynchronisation.
- Fehlererkennung: Das Erkennen von Übertragungsfehlern (z. B. aufgrund von Störungen auf dem Band) durch den Empfänger. Hierzu fügt der Sender im Falle des AX.25-Protokolls vor dem Ende eines jeden Datenframes (unmittelbar vor der Rahmensynchronisation) eine Prüfsumme ein, die vom Empfänger kontrolliert werden kann.

Als letzten Schritt der Vorbereitung führt das im TNC integrierte Modem nun die Modulation durch. Dieser Schritt ist nicht mehr Teil der AX.25-Spezifikation. Aus der langjährigen Praxis des Betriebs von Packet-Radio haben sich jedoch zwei Varianten der Modulation herauskristallisiert: Einerseits AFSK mit 1200 Baud entsprechend des Half-Duplex-Modus der Bell-202-Modem-Spezifikation und andererseits FSK mit 9600 Baud nach G3RUH. Praktisch alle Hardware-TNCs unterstützen eines dieser beiden Verfahren. Software-Lösungen bieten hingegen neben diesen beiden Verfahren auch noch eine Reihe weiterer Modulationsarten.

Anschluss an das Funkgerät

Für die Verbindung zwischen dem TNC und dem Funkgerät wird eine herkömmliche analoge Audioverbindung genutzt. Fast jedes Funkgerät verfügt über einen Anschluss für Mikrofon und externen Lautsprecher, welche mit dem TNC verbunden werden können. Im Falle von Software-TNCs ist entsprechend die Soundkarte des Computers mit dem Funkgerät zu koppeln.

Manche hochwertigere Funkgeräte verfügen auch über einen sogenannten 9600-Port. Dieser bietet einen Ein- und Ausgang für Audiosignale, bei denen das Signal jedoch an den üblichen NF-Filtern (Bandpass) vorbeigeführt wird und somit ein größeres Frequenzspektrum durchgelassen werden kann. Der Verzicht auf Filter vermeidet außerdem ein nichtlineares Verhalten hinsichtlich des Phasenganges und stellt damit eine einheitliche Verzögerung des gesamten Signals unabhängig von den übertragenen NF-Frequenzen sicher. Mit einem 9600-Port ist es einfacher, hohe Datenübertragungsraten zu erreichen. Dieser Anschluss ist die Voraussetzung für die Verwendung eines 9600-Baud-Modems nach G3RUH.

[IMG: https://www.hamspirit.de/wp-content/uploads/2021/01/baofeng_data-338x450.jpeg]

Ein Baofeng UV-5R mit Audio-Datenkabel

Audiopegel

Grundsätzlich gilt: Der Audioausgang des TNCs oder – im Falle eines Software-TNCs – des Computers ist mit dem Audioeingang des Funkgeräts und andersherum der Audioausgang des Funkgeräts wiederum mit dem Audioeingang von TNC bzw. Computer zu verbinden.

In der Praxis hat es sich als sehr wichtig herausgestellt, die Audio-Lautstärke in beide Richtungen korrekt einzupegeln. Sendeseitig sollte die in das Funkgerät eingegebene Audiolautstärke so gewählt sein, dass sie so laut wie möglich ist ohne das es zu Verzerrungen des Signals kommt. Bei Geräten mit eingebautem Audio Level Controller (ALC) kann dies leicht erreicht werden, indem die Lautstärke soweit angehoben wird, dass die ALC-Anzeige gerade so nicht ausschlägt. Empfangsseitig sollte die Lautstärke so eingestellt werden, dass die empfangenen Signale keinesfalls übersteuern.

Verbesserungspotential von AX.25-TNCs

Bei den Versuchen wurden unterschiedliche AX.25-TNCs verwendet. Eine Besonderheit war dabei der PicoAPRS, der TNC und Transceiver zu einem winzigen Gerät vereint. Als reine Software-Lösung kam immer wieder Dire Wolf zum Einsatz.

Leider sieht AX.25 für die Fehlererkennung lediglich eine 2 Byte lange CRC-Prüfsumme vor. Diese erlaubt es mit relativ hoher Wahrscheinlichkeit, eine fehlerhafte Übertragung als solche zu erkennen. Die Möglichkeit einer Fehlerkorrektur, weil z. B. der Kanal kurz gestört war, erlaubt dies aber nur sehr eingeschränkt. Der technische Stand dazu hat sich seit der Spezifikation von AX.25 aber deutlich weiterentwickelt.

So unterstützt der Software-TNC Dire Wolf schon seit einiger Zeit eine Fehlerkorrektur mittels Reed-Solomon-Codes. Diese Fehlerkorrektur kommt in einem zu AX.25

rückwärtskompatiblen Kodierungsverfahren mit der Bezeichnung FX.25 zum Einsatz. Dem aktuellen Stand der Technik entspricht dies jedoch auch noch nicht.

Ein heute übliches Verfahren zur Fehlererkennung und auch -korrektur sind sogenannte Low-Density-Parity-Check-Codes (LDPC), die z. B. bei DVB-S2 oder FT8 verwendet werden. Jan DL9JBE hat sich mit diesem Verfahren näher beschäftigt und dabei die experimentelle Software pktfec-tnc entwickelt. Das ist ein Software-TNC, der dieses Verfahren zur Fehlerkorrektur einsetzt. Da pktfec-tnc ebenfalls die KISS-Schnittstelle verwendet, kann es einfach anstelle eines AX.25-TNC verwendet werden. Als Modulation zur Übergabe per Audiokanal an das Funkgerät wird Quadraturamplitudenmodulation (QAM) verwendet, welche hinsichtlich Symbolrate (Baudrate) und Symbolanzahl (Bits je Symbol) relativ frei konfiguriert und somit an die Kanaleigenschaften angepasst werden kann. Die genauen technischen Hintergründe hat Jan im Artikel Schnellere Datenübertragung ohne Data-Port beschrieben. Praktisch gibt es also mehrere Methoden wie sich das AX.25-typische Ein- und Auspacken von IP-Paketen mit einem modernen TNC verbinden lässt.

Einschränkungen von AX.25

Aber auch bei der Verwendung von AX.25 für das Ein- und Auspacken von IP-Paketen stellten sich einige praktische aber auch theoretische Probleme. Wie bereits erwähnt unterstützt AX.25 ausschließlich IPv4 und ist nicht auf IPv6 vorbereitet. Das ist auch nicht verwundernswert, da das letzte Update des AX.25-Standards aus dem Jahre 1998 stammt, dem gleichen Jahr in dem IPv6 erstmals spezifiziert wurde. Leider scheint AX.25 IP-Datenpakete praktisch auch auf eine maximale Länge von 255 Bytes zu beschränken. Dies würde die Ergänzung einer IPv6-Unterstützung erschweren. Denn IPv6 verlangt, dass jeder Übertragungsweg darauf vorbereitet ist, Pakete mit 1280 Byte Länge zu unterstützen. Es wäre also nötig, einen neuen Mechanismus zur Aufteilung größerer Datenpakete in mehrere Teilpakete zu implementieren.

Bei der Adressierung ist AX.25 für den vorgesehenen Anwendungsfall nur eingeschränkt geeignet bzw. ineffizient. Eigentlich wird nur eine Angabe das Rufzeichens der Sendestation benötigt. AX.25 verlangt jedoch zusätzlich auch zwingend die Angabe eines Zielrufzeichens. Neben der erwähnten unpraktischen um ein Bit verschobenen Kodierung kommt erschwerend noch hinzu, dass AX.25 für das Absender- wie Empfängerrufzeichen jeweils eine feste Breite von 6 Zeichen zuzüglich SSID vorsieht und dafür jeweils 7 Byte benötigt. Das führt einerseits dazu, dass auch bei kurzen Rufzeichen stets 14 Byte für die Adressierung verbraucht werden. Anderseits macht es die Nutzung von Sonderrufzeichen unmöglich, die länger als 6 Zeichen sind.

Abschließend sei noch erwähnt, dass keine Umsetzung von AX.25-IP-Vernetzung für andere Betriebssysteme als Linux gefunden werden konnte. Damit war dann z. B. das

von DL9JBE benutzte FreeBSD außen vor. Eine Portierung erschien aufgrund der Mängel hinsichtlich IPv6 und der Adressierung sowie der Gesamtkomplexität des Standards nicht vielversprechend.

[IMG: https://www.hamspirit.de/wp-content/uploads/2021/01/2m_wireshark-scaled.jpeg]

Debugging von IP-Verbindungen mit dem Netzwerkanalyse-Werkzeug Wireshark

Alternativen zum Verpacken von IP-Paketen

Eine weitere Möglichkeit zum Ein- und Auspacken der IP-Pakete ist tncattach, welches auf einfache Art und Weise die beiden Aufgaben der Rufzeichennennung und Protokollidentifikation übernimmt und die Daten per KISS-Protokoll an einen TNC weiterreicht. Um IPv4 bzw. IPv6-Pakete zu kennzeichnen, stellt tncattach die zwei Byte lange Markierung gemäß dem Ethernet-Standard IEEE 802.3 vor jedes Paket, also hexadezimal 0x08 0x00 für ein IPv4- und 0x86 0xDD für ein IPv6-Paket. Es werden vorab noch 2 weitere Bytes hinzugefügt, die Linux-spezifisch definiert sind, deren genauer Zweck sich im Rahmen der Versuche nicht erschlossen hat. Die Rufzeichennennung erfolgt bei tncattach nicht in jedem Paket. Stattdessen werden von Zeit zu Zeit separate Datenpakete erzeugt, die nur das Rufzeichen erhalten. Das bietet den Vorteil, dass bei jeder Übertragung einige Byte eingespart werden können.

Die Software tncattach ist für die reine IP-Vernetzung gegenüber AX.25 als Fortschritt anzusehen. Die Funktionalität beschränkt sich weitgehend auf die IP-Vernetzung (zusätzlich ist auch die Vernetzung auf Ethernet-Ebene möglich) und IPv6 wird unterstützt. Rufzeichen werden ohne Bitverschiebung kodiert, also im Klartext. Es ist nur das Rufzeichen der Sendestation anzugeben und lange Sonderrufzeichen stellen auch kein Problem dar.

Nicht so gut gefallen hat, dass das Rufzeichen nicht in jedes Paket einkodiert wird. Rein formal wird das die Verpflichtung zur Rufzeichennennung im Amateurfunk möglicherweise erfüllen. Zumindest praktisch ist es so aber nicht ohne weiteres möglich, jede Sendung klar zuzuordnen. Es ist nicht erkennbar, welches IP-Datenpaket zu welchem Rufzeichen-Paket gehört. Eine Zuordnung wie im Sprechfunk anhand der Stimme ist ja nicht möglich.

Als größter Nachteil wurde allerdings festgestellt, dass tncattach ausschließlich für Linux entwickelt wird. Nach einer kurzen Betrachtung des Quellcodes ist der Eindruck entstanden, dass eine Portierung auf andere Betriebssysteme bisher leider nicht vorgesehen ist.

hamtunnel

Eine weitere Alternative zum Verpacken von IP-Paketen hat Jan DL9JBE als Ergänzung zu pktfec-tnc implementiert und dabei folgende Design-Kriterien zugrunde gelegt:

- Unterstützung für Linux und FreeBSD
- Kompatibilität zum KISS-Protokoll, um bestehende TNCs nutzen zu können
- Einfaches Protokoll, das leicht für den Menschen lesbar ist
- Rufzeichennennung in jedem Paket im Klartext
- Unterstützung beliebig langer Rufzeichen (auch z. B. Rufzeichen aus dem CB-Funk oder anderen Funkdiensten)

Die Software heißt hamtunnel und ist kann zusammen mit pktfec-tnc heruntergeladen werden. Genauso wie AX.25 und tncattach übernimmt hamtunnel die Anreicherung von IP-Paketen um eine Protokollinformation und das Rufzeichen. Dazu wird vor jedes Datenpaket als erstes das Rufzeichen in beliebiger Länge im ASCII-Klartext vorangestellt und mit einem Null-Byte (also hexadezimal 0x00) abgeschlossen. Es folgt dann die Markierung des IP-Protokolls im Klartext. Konkret wird die Zeichenfolge „IP4“ bzw „IP6“ verwendet, die sich von selber erklären dürften. Auch diese Sequenz wird mit einem Null-Byte abgeschlossen. Im Anschluss folgt unverändert das eigentliche IP-Paket.

Lesbarkeit von Daten für den Menschen

Die Lesbarkeit von Daten für den Menschen ist dabei ein interessanter Punkt, der hier noch etwas weiter ausgeführt werden soll. Viele Datenformate machen es leider schwer, Rohdaten ohne Spezialsoftware zu betrachten und zu verstehen. Das soll hier an einem Vergleich üblicher Datenpakete von AX.25 und hamtunnel dargestellt werden. Dazu wird jeweils der Anfang eines typischen Datenpaket wiedergegeben, wie es von einem TNC empfangen wurde. Es handelt sich dabei jeweils um ein IPv4-Paket, gesendet von der Station DL1PZ. Dabei werden druckbare ASCII-Zeichen als Text dargestellt und nicht-druckbare Zeichen als Hexadezimalzahl nach dem Muster <00>. In Fettschrift wird das Rufzeichen und die Protokollinformation hervorgehoben.

Beispiel AX.25:

Code

```
<a2><a6><a8>@@@`<88><98>b<a0><b4>@a<03><cd><00><03><00><cc><07><04>...
```

Beispiel hamtunnel:

Code

```
DL1PZ<00>IP4<00><49><50><34><00><45><00><00><54><FB><06>@<00>@<01>...
```

> Im Falle von AX.25 ist praktisch überhaupt nicht zu erkennen, was für Daten vorliegen. Die Sequenz <88><98>b<a0><b4> ist übrigens DL1PZ in ASCII-Kodierung um ein Bit nach links verschoben und das Byte <cc> steht für IPv4. Die Kodierung von hamtunnel hingegen lässt das Rufzeichen leicht erkennen und gibt mit „IP4“ einen klaren Hinweis darauf, welche Art von Daten anschließend zu erwarten sind. Zumindest für alle mit Programmiererfahrung ist das Null-Byte <00> leicht als Trennzeichen zu identifizieren. Die anschließend folgenden IPv4-Daten sind (genauso wie IPv6) leider nicht für den Menschen lesbar, da bei IP die Kopfdaten stets binär kodiert werden.
>
> [IMG: https://www.hamspirit.de/wp-content/uploads/2021/01/2m_chat-450x338.jpeg]
>
> Die Früchte der Arbeit: IP-basierter IRC-Chat auf 144,850 MHz im Channel #2m-talk
>
> ## Fazit
>
> Viele Wege führen zum Ziel. Zum Vorbehandeln der IP-Pakete mit Rufzeichen und Protokollinformationen können AX.25, tncattach oder hamtunnel genutzt werden. TNCs gibt es in Hardware, wie der PicoAPRS, und in Software, wie Dire Wolf oder pktfec-tnc, teils mit klassischer AX.25-Kodierung, teils mit moderneren Verfahren. Bei der Modulation ermöglichen die TNCs neben Klassikern wie 1200-Baud-AFSK gemäß Bell-202 oder 9600-Baud nach G3RUH viele weitere Verfahren. Dank KISS-Protokoll lassen sich die Komponenten beliebig kombinieren.
>
> Eines darf dabei aber nicht vergessen werden: Nur wenn alle beteiligten Stationen exakt die selbe Art verwenden um IP-Pakete einzupacken, für die Übertragung vorzubereiten und zu modulieren ist ein erfolgreicher Datenaustausch möglich. Denn die Verfahren sind nicht untereinander kompatibel. Das ist leider ein Dilemma. Denn entweder verwenden wir möglichst etablierte („gut abgehangene“) Verfahren wie AX.25 mit 1200-Baud-AFSK-Übertragung und bleiben auch mit alten Hardware-TNCs kompatibel – oder wir verzichten auf die Abwärtskompatibilität und setzen auf neuere Verfahren mit eleganteren Protokollen und moderner Fehlerkorrektur.
>
> Alles anzeigen

Quelle: https://www.hamspirit.de/12311/ip-vernetzung-per-funkgeraet/