Documentatie NetAVR

Datacommunicatie practicum
2009 - 2010

Roy van Dam
David Vaessen
Ramon Kleiss

https://github.com/royvandam/NetAVR

Aangepast: Oct 26, 2011

Inhoudsopgave

Samenvatting

Voor het practicum datacommunicatie is het de opdracht om een TCP/IP stack te implementeren op een microcontroller in combinatie met een embedded ethernet controller. Dit om een alternatief te kunnen bieden op het huidige aanbod van embedded netwerk stacks.

Omdat er aardig wat kennis van netwerk communicatie binnen de project aanwezig is. Is er besloten om een compleet nieuwe netwerk stack te ontwikkelen. Er worden wel routines geanalyseerd uit bestaande netwerk stacks onderzocht om zo een aanzienlijke tijdwinst te kunnen boeken. Eh het wiel immers opnieuw uitvinden levert meestal vrij weinig op, het wiel verbeteren des te meer. Om een demonstratie te kunnen geven de van stack is er tevens een modulaire prototype kit opworpen. Deze kit is voorzien van een ATMega32 (TQFP), USB, LCD, en socket voor een SPINet Ethernet kaart.

In dit verslag gaan wordt er op de technische aspecten van de stack ingegaan. Het ontwerp van de modulaire kit die tevens voor educatieve doeleinden gebruikt kan worden. En wordt er een globale beschrijng van de werking van het TCP/IP model geven. Meer diepgaande informatie hierover is te vinden in betreffende RFC (Request for Comments) documenten.

1 Introductie

In toenemende maten gaat embedded apparatuur gebruik maken van het internet. Zeker met de komst van het IPv6 protocol waarmee de problemen van de netwerk adressering zijn opgelost en er plaats is voor letterlijk miljarden kleine 'slimme' apparaten die met elkaar communiceren.

Om er voor te zorgen dat al deze communicatie in goede banen verloopt zijn er diverse protocollen in de loop der jaren ontwikkeld. Deze protocollen bieden de mogelijkheid om verschillende platformen op een relatief simpele manier met elkaar te laten communiceren.

Een netwerk stack is een verzameling van deze verschillende protocollen in een pakket. Dit kan vervolgens gebruikt worden om op een hoog niveau verbindingen te leggen tussen verschillende apparaten. Zonder dat hiervoor diepgaande kennis van netwerken vereist is.

Er zijn reeds diverse stacks ontwikkeld die volgens het zelfde principen werken. Maar allen hebben een andere implementatie afhankelijk van de doelgroep. De NetAVR stack richt zich vooral op de hele kleine microcontrollers, zoals de Atmel AVR waar weinig resources beschikbaar zijn.

De NetAVR stack is dus ook ontworpen om zo efficiënt en snel mogelijk een netwerk pakket te kunnen afhandelen. Dit in tegenstelling tot veel andere stacks die juist weer meer mate van controle geven.

Om alles van het begin tot het einde zelf in handen te hebben is de stack van af de grond af aan opgebouwd. Hierbij is er wel gekeken naar de implementatie van andere stacks. Zodat er geen dingen uitgevonden worden die er al bestaan.

Als basis is er een globaal model van de stack ontworpen. Dit model is ontworpen aan de hand van de verschillende netwerk protocollen. Het schema hiernaast beschrijft dit model.

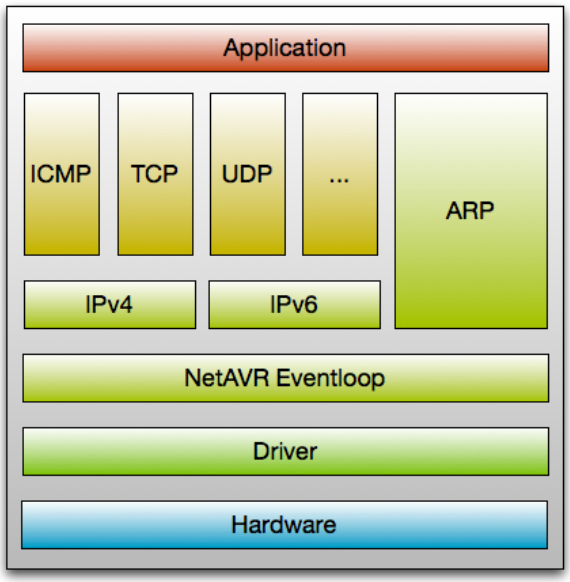

Fig. 1: schematisch overzicht.

Waneer er een pakket ontvangen wordt door de hardware zal er een interrupt routine aangeroepen worden. Deze zal het pakket naar een buffer schrijven welke vervolgens verwerkt wordt in de eventloop. Hier wordt bepaald welk type pakket het betreft. Namelijk IPv4, IPv6 of ARP. Dit kan tevens gemakkelijk verder worden uitgebreid. Vervolgens wordt er door de betreffende protocollen het pakket verder verwerkt en uiteindelijk aan de applicatie gepresenteerd. Waarna de applicatie een actie kan ondernemen en eventueel een antwoord op het bericht kan sturen.

2 Implementatie

Hier wordt beschreven wat er tot nu toe gerealiseerd is en nog dient te worden door ontwikkeld.

Het de ontwikkeling van de stack is er vooral rekening gehouden met de snelheid en omvang van de software. De volledige netwerk stack beslaat tot nu toe slechts 5 kilobyte aan ROM code. Wat in vergelijking met andere stacks in het zelfde segment zeer klein is.

De stack is op het moment in staat om UDP en ICMP pakketen af te vangen en te verwerken. Ook de snelheid van de stack is aanzienlijk te noemen. Het verwerken van een ping pakket op een AVR duurt minder dan één milliseconden.

2.1 Hardware

Voor ontwikkelingsdoeleinden en om de werking van de stack te kunnen demonstreren is er een hardware kit ontwikkeld. De stack draait in dit geval op een Atmel AVR microcontroller. Middels een ethernet module die geplaatst kan worden op de kit, kan er verbinding gemaakt worden met een netwerk.

Op de kit zijn verschillende interface poorten aanwezig waarmee de functionaliteit van de kit kan worden uitgebreid. De kit is simplistisch en robuust ontworpen zodat deze erg degelijk en reproduceerbaar is. Tevens voorzien van USB zodat er geen afzonderlijke com poort meer nodig is.

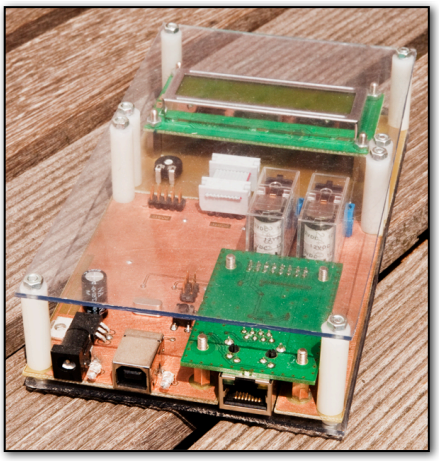

Fig. 2: Hardware kit

2.2 Hardware drivers

De drivers zorgen er voor dat de software de hardware kan aansturen op een hoger niveau. Er is gekozen voor een modulair systeem waardoor drivers eenvoudig toegevoegd of verwijderd kunnen worden. Voor deze kit zijn er drivers geschreven voor de UART, SPI, ENC28J60 (netwerk controller).

2.3 Eventloop

De eventloop is een eindeloze lus welke constant kijkt of er nieuwe pakketen klaar staan om verwerkt te worden. Zodra er een pakket binnenkomt om verwerkt te worden zorgt deze lus er voor dat het pakket verwerkt word op de juiste plaats. Volgens het schematisch overzicht kunnen dit zijn of IPv4 of IPv6 of ARP. Op dit moment zijn alleen IPv4 en ARP geïmplementeerd.

Tevens houd de eventloop de statistieken bij van de stack. Hierbij moet gedacht worden aan aantal pakketten, verplaatste data, huidige transmissiesnelheid, etc. Hier is nog ruimte voor uitbreidingen. Ook word de status van de ethernet controller in de gaten gehouden.

2.4 Address resolution protocol

ARP wordt gebruikt voor het achterhalen van netwerk adressen op het netwerk. Het houd een lijst bij van actieve apparaten binnen het netwerk. Wanneer er een ARP request binnen komt welke vraagt om het IP adres van de stack. Zal het reageren met het huidige MAC adres van de netwerkkaart.

Wanneer er een pakket verstuurd wordt van uit de applicatie gebeurd dit meestal aan de hand van een IP adres. ARP dient dus eerst het juiste MAC adres bij het IP address op te zoeken in de tabel. Waneer deze niet gevonden wordt Dient er dus eerst een APR request te worden geplaatst op het netwerk. En moet het pakket in een wachtrij gezet worden. Het pakket wordt weer van de wachtrij gehaald zodra er een timer is verstreken of er binnen die tijd een antwoord op het request is binnengekomen. Deze functionaliteit is echt nog niet vollig

geïmplementeerd en dient te worden door ontwikkeld.

2.5 Internet protocol

Er zijn verschillende functies van IPv4 geïmplementeerd waaronder CRC check-summing. Echter fragmentatie dient nog volledig te worden ontwikkeld. Dit betekent niet dat er geen gebruik gemaakt worden van bijvoorbeeld ICMP en UDP. Enkel kunnen nu pakketen met de maximale standaard lengte worden ontvangen van 1500 bytes. IPv6 is in zijn geheel nog niet geïmplementeerd.

2.6 Internet control message protocol

ICMP is een zeer omvangrijk en krachtig protocol voor netwerk analyse. Er is tot nu toe echter alleen ICMP-Echo-Reply (ping) geïmplementeerd. Dit omdat er nog verder geen andere functies nodig zijn geweest voor het ontwikkelen van de stack. Echter is het ontwerp wel zo opgezet dat eventuele andere functies gemakkelijk kunnen worden toegevoegd.

2.7 User datagram protocol

UDP is voor het grootste gedeelte geïmplementeerd. Er kunnen applicaties in de vorm van een functie worden gebonden aan specifieke UDP netwerk poorten. Deze functie wordt aangeroepen op het moment dat er informatie wordt ontvangen op de desbetreffende netwerk poort. Als voorbeeld is er een UDP echo server geïmplementeerd die simpelweg alles retour afzender zend.

Hoewel het mogelijk is om al UDP pakketen op te bouwen. Zou het makkelijker zijn als er nog een functie wordt geschreven waar men simpel weg een IP adres en een data pakket naar kan wegschrijven.

2.8 Transmission control protocol

TCP is een van de meest complexe en uitgebreide protocollen binnen een netwerk stack. Het verijst veel timing en slim geheugen management wat zeker op een microcontroller als de AVR niet gemakkelijk is. Van TCP is er dus echter ook alleen nog maar de mogelijkheid om applicaties te laten binden op een netwerk poort. Hier kan dan verder helaas nog niets mee gedaan worden.

3 Protocollen

3.1 Internet Protocol

Het internet protocol (IP) is een “connectionless” protocol om verschillende systemen (hosts) met elkaar op een netwerk te laten communiceren. Het is de basis van de “Internet Protocol Suite” (TCP/IP), een verzameling van protocollen die worden gebruikt voor communicatie op een netwerk zoals het internet.

Het idee achter het protocol is dat netwerk datagrammen op basis van een adres (IP-adressen) tussen verschillende host kunnen worden verzonden. Het internet protocol definieert de methode van adressering en de structuren voor het inkapselen van datagrammen. Het inkapselen van datagrammen betekend dat informatie van een hoger gelegen protocol (zoals TCP of UDP) wordt ingekapseld door een IP pakket.

Het internet protocol heeft geen mogelijkheden tot het herstellen van fouten. Pakketten die als fout worden beschouwd worden simpel weg verworpen. Dit betekent dus dat eventuele fouten in hoger gelegen protocollen dienen te worden afgevangen. Mogelijke fouten die kunnen optreden tijdens het verzenden van pakket en afhandeling van deze fouten door het protocol zijn:

```
- Data corruptie
  Wordt gecontroleerd met behulp van
  de checksum die mee wordt gezonden
  in een IP-pakket.
- Verloren pakketten
  Volledige pakket wordt verworpen.
- Dubbele aankomst
  Volledige pakket wordt verworpen.
- Gesegmenteerde pakketten die niet in
  de juiste volgorde aankomen
  Zullen door het protocol in de
  juiste volgorde worden aangeboden
  aan hoger gelegen protocollen.
```

IPv4

IPv4 is de versie van het internet protocol dat op het moment het meest wordt gebruikt, het wordt beschreven in IETF RFC 791 (zie http://www.ietf.org).

Adressering

IPv4 gebruikt een 32-bits adressering (vier bytes), wat het aantal beschikbare adressen limiteert tot 4,295,967,296 unieke adressen. Hiervan zijn er ongeveer 18 miljoen gereserveerd voor private netwerken en ongeveer 270 miljoen gereserveerd voor multicast adressen. IPv4 adressen zijn meestal geschreven door middel van “dot-decimal notation” wat betekend dat die vier bytes die het adres samenstellen apart worden opgeschreven als een decimaal nummer gescheiden door een punt zoals:

192.0.2.235

Er zijn echter nog andere manieren om dit op te schrijven, hieronder een lijstje:

```
Dot-decimal notation
- 192.0.2.235
- Dit is de standaard notatie voor
  adressen
Dotted hexadecimal
- 0xC0.0x00.0x02.0xEB
- Elke byte als hexadecimaal getal

Dotted octal
- 0300.0000.0002.0353
- Elke byte als octaal getal

Hexadecimal
- 0xC00002EB
- Samenvoeging van de waarden van elke
  byte

Decimal
- 3221226219
- Het gehele 32-bits getal als
  decimale waarde

Octal
- 030000001353
- Het gehele 32-bits getal als octale
  waarde
```

Pakket structuur

Een IPv4 pakket bestaat uit een header sectie en een data sectie. De header heeft 13 velden waarvan er 12 verplicht zijn. Het 13e veld is optioneel (rood in de tabel) en veelzeggend genaamd “options”. De pakketten zijn big endian gecodeerd en voor de uitleg wordt er vanuit gegaan dat de MSB (most-significant byte) als eerste komt.

Bit offset	0–3	4–7	8–15	16–18	19–31
0	Version	Header length	Differentiated Services	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options (if Header Length > 5)				
160 or 192+	Data				

(Bron: Wikipedia)

Hieronder een beschrijving van de belangrijkste velden uit de header.

Version
De versie van het protocol, aangezien dit een IPv4 pakket is, zal de waarde van dit veld altijd 4 zijn.

Header length
In dit veld wordt de lengte van de header aangegeven als het aantal 32-bits words. Dit veld is aanwezig omdat de header van een variabele lengte kan zijn (met een minimum van 5). Wanneer er options worden meegegeven zal de waarde van dit veld anders zijn.

Total length
Dit veld bevat de totale lengte van het pakket, dus de lengte van de data in bytes plus de lengte van de header (minstens 20 bytes). De minimum waarde van dit veld is 20 (20 header bytes + 0 data bytes), maar het minimum aantal bytes dat een ontvanger moet kunnen behandelen is 567.

Identification
De voornaamste reden dat dit veld wordt gebruikt is om fragmenten van een gefragmenteerd pakket in de juiste volgorde te te construeren.

Flags
Het flags veld uit de header is een 3-bits veld waarin kan worden aangegeven wat er moet gebeuren met het pakket mocht het gefragmenteerd worden. 2 van de 3 bits worden gebruikt:

D-bit
Dit bit staat voor “don’t fragment”, wat betekend dat wanneer het pakketje moet worden gefragmenteerd

M-bit
Dit bit staat voor “more”, wanneer een pakket wordt gefragmenteerd wordt dit bit in het laatste deel van het pakket op 1 gezet zodat de ontvanger weet dat het ontvangen pakket het laatste deel van een geheel pakket is.

Fragment Offset
Dit veld wordt gebruikt voor het defragmenteren van pakketten. De waarde van dit veld is de “offset” van de meegeleverde data ten opzichte van het eerste databyte. De waarde wordt gegeven in blokken van 8 bytes, dit zorgt ervoor dat wanneer data gefragmenteerd wordt, dit altijd in groottes van tenminste 8 bytes (of een meervoud daarvan) moet gebeuren, dus 8 bytes, 16 bytes, 24 bytes, 32 bytes, enz.

Protocol
In dit veld wordt het upper-level protocol aangegeven, de meest gebruikte waarden hiervan zijn:

```
1: Internet Control Message Protocol (ICMP)
2: Internet Group Management Protocol (IGMP)
6: Transmission Control Protocol (TCP)
17: User Datagram Protocol (UDP)

Header Checksum
  Door middel van dit veld kan worden gecontroleerd of het pakket tijdens verzending niet gecorrumpeerd is.

Source Address
  Dit veld bevat het adres van de host waar het pakket vandaan komt. Het is belangrijk op te merken dat dit gegeven adres niet het “echte” adres is, omdat er soms gebruik wordt gemaakt van NAT (Network Address Translation) tabellen. Hierbij wordt dit veld veranderd in het adres van de machine die NAT gebruikt. Wanneer er gereageerd wordt op een pakket, ontvangt deze host dit pakket en stuurt het door naar de oorspronkelijke source host.

Destination Address
  Met dit veld wordt aangegeven naar welk adres het pakket verzonden moet worden.

Data
  Dit veld bevat mogelijke data die met het pakket wordt meegezonden. Dit kan plain-text zijn, maar dat is niet gebruikelijk. Meestal is het een header of pakket van een upper-leven protocol (zoals TCP of UDP), die op zijn beurt ook weer data bevat.
```

Fragmentatie

Elk netwerk heeft een MTU (Maximum Transmission Unit). Dit is de maximale grootte van een pakket, in bytes. Wanneer een pakket bij een gateway aankomt en het doorgestuurd moet worden, controleert de gateway de grootte van het ontvangen pakketje. Mocht deze groter zijn dan de opgegeven MTU, dan moet het pakket gefragmenteerd worden. Wanneer een pakket gefragmenteerd worden, worden de volgende velden aangepast:

```
- Het total length veld wordt aangepast naar de nieuwe grootte van de pakketjes
- De MF-flag wordt geset bij alle nieuwe fragmenten (behalve de laatste)
- De fragment offset wordt bijgesteld voor alle fragmenten
- De header checksum wordt opnieuw berekend
```

Bijvoorbeeld, een 4,500-bytes data pakket wordt over een netwerk gestuurd met een MTU van 2,500. Ook wordt het ingekapseld in een IP-pakket zonder opties. De totale grootte van het te versturen pakket is dus 4,520. Om dit over het gegeven netwerk te sturen (met een MTU van 2,500) moet het in twee verschillende pakket verstuurd worden:

Pakket	Totale length		MF-flag is geset?	Fragment offset
	Header	Data		
1	2500		Ja	0
	20	2480		
2	2040		Nee	310

(Bron: Wikipedia)

Het is voor een host duidelijk dat hij een gefragmenteerd pakket binnen krijgt wanneer:

```
- MF-flag gelijk is aan ‘1’
- Fragment offset is anders dan ‘0’
```

De ontvanger moet de ontvangen data dan opslaan samen met het identificatie nummer totdat hij alle gefragmenteerde data ontvangt. Wanneer alle data ontvangen is, kan de host het in de juiste volgorde defragmenteren.

3.2 Internet Control Message Protocol

Het Internet Control Message Protocol (ICMP) wordt vaak door hosts gebruikt om diagnostische berichten te versturen. Deze kunnen bijvoorbeeld aangeven dat een aangevraagde service niet beschikbaar is of dat een router of host niet te vinden is.

ICMP is afhankelijk van IP voor zijn functioneren, en is ook een belangrijk onderdeel van TCP/IP. Een verschil tussen ICMP en bijvoorbeeld TCP of UPD is dat ICMP over het algemeen niet gebruikt wordt om data te versturen tussen host. In feite is ICMP meer een notificatie systeem dan een daadwerkelijk communicatie systeem. Daarnaast wordt ICMP meestal niet direct door een gebruiker gebruikt, met uitzondering van “ping” en “traceroute”.

Pakket structuur

Een ICMP header wordt altijd ingekapseld door een IP header, ondanks het feit dat ICMP op dezelfde laag functioneert als IP. Hieronder een overzicht van een ICMP header:

Bits	0-7	8-15	16-23	24-31
0	Type	Code	Checksum	
32	ID		Sequence	

(Bron: Wikipedia)

```
Type
  Dit veld geeft het type ICMP bericht aan.

Code
  Dit veld is geeft de code van het bericht aan. Dit om het bericht verder te specificeren, zo kan bijvoorbeeld een bericht met het type Destination Unreachable verschillende codes hebben, waarbij elke code zijn eigen betekenis heeft.

Checksum
  Checksum bevat de checksum van het pakket, welke berekend wordt met de waarde van dit veld op nul. Het algoritme om de checksum te berekenen is hetzelfde als bij IPv4.
```

```
ID
  Dit veld bevat een (meestal) willekeurige ID waarde, welke terug gestuurd moet worden bij een reactie op “echo request”.

Sequence
  Ook dit veld moet terug gestuurd worden wanneer er een reactie wordt gegeven op “echo request”.
```

ICMP wordt het meest gebruikt om andere hosts te pingen, wat inhoudt dat er gecontroleerd wordt wat de “round-trip delay” naar een bepaalde host is. De volgende twee pakketten worden dan verstuurd over het netwerk:

Echo Request:

Veld	Waarde
Type	8
Code	0
Checksum	*Hangt af van ID en Sequence*
ID	Willekeurig getal
Sequence	Willekeurig getal

Echo Reply:

Veld	Waarde
Type	0
Code	0
Checksum	*Hangt af van ID en Sequence*
ID	Willekeurig getal
Sequen	Willekeurig getal

3.3 User Datagram Protocol

Het User Datagram Protocol is een van de belangrijkste protocollen van TCP/IP. Met UDP kunnen applicaties berichten (datagrammen) naar andere hosts op het netwerk versturen. Het versturen gebeurt direct zonder dat er eerste een verbinding hoeft worden opgezet.

UDP gebruikt een simpel transmissie model waar een “hand-shaking”-algoritme (voor betrouwbaarheid, nummering van berichten en data integriteit). Daardoor verzocht UDP dus een onbetrouwbare service, aangezien berichten niet in volgorde hoeven aan te komen, dubbel aan kunnen komen of helemaal niet aan hoeven te komen. Hierdoor wordt wel de overhead minder, aangezien er geen gehele controle hoeft worden uitgevoerd op de ontvangen pakketen. Tijd afhankelijke applicaties gebruiken vaak UDP omdat pakketten verliezen liever gedaan wordt dan lange tijd wachten op pakketten. Wanneer er fout controle moet plaats vinden, gaat UDP er vanuit dat dit gebeurd door de application.

Het stateless principe van UDP is ook handig voor servers die kleine aanvragen moeten behandelen voor veel gebruikers. Een voorbeeld hiervan is een DNS server. Ook wordt UDP vaak gebruikt voor bijvoorbeeld DHCP of het streamen van media.

Poorten

UDP gebruikt datagram sockets voor communicatie tussen twee hosts. Sockets binden applicaties aan service poorts, die functioneren als een eindpunt van data transmissie. Een poort is een software constructie die geïndentificeerd wordt door een 16-bit integer, wat de poorten 0 tot 65,535 toestaat. Port 0 is gereserveerd, maar is een toegestaan poort nummer als de verzender geen antwoorden verwacht.

De IANA heeft deze poorten in verschillende groepen opgedeeld:

```
- Groep 1: poort 0 tot poort 1,023 zijn de bekende (permanente) poorten. Deze poorten worden ook toegewezen door de IANA.

- Groep 2: poort 1,024 tot poort 49,151 zijn geregistreerde poorten Deze poorten worden niet toegewezen door de IANA. Wel kunnen ze worden geregistreerd om dubbele toewijzingen te voorkomen.

- Groep 3: poort 49,152 tot poort 65,535 zijn de dynamische of private poorten. Deze poorten hebben niets met de IANA te maken. Ze staan ook wel bekend als kort levende poorten, aangezien ze kunnen worden gebruikt door UDP applicaties op de client.
```

Pakket structuur

De structuur van een UDP pakket wordt hieronder geschetst:

Bits	0 – 15	16 – 31
0	Source Port Number	Destination Port Number
32	Length	Checksum
64+	Data	

```
Source Port Number
  Dit veld geeft de poort aan vanaf waar het bericht verzonden is. Dit mag nul zijn, in het geval dat er niet gereageerd wordt. Als de verzender een server is, is het poort nummer waarschijnlijk een veel gebruikt port nummer. Dit veld is optioneel.

Destination Port Number
  Dit veld geeft aan naar welke port het pakket gestuurd moet worden.

Length
  Length geeft aan hoeveel data er in het pakket zit, inclusief de header. De minimum lengte is
  8 bytes (8 bytes header + 0 bytes data).

Checksum
```

```
  Dit veld bevat de checksum van het
pakket, wanneer het weggelaten wordt
zal het volledig bestaan uit nullen.
In IPv4 mag dit veld weggelaten
worden, in IPv6 is het echter
verplicht.
```

Conclusie en aanbevelingen

Ondanks dat er nog verschillende functies ontbreken in de stack. Is het zeker een goed begin waar de komende jaren op door ontwikkeld kan worden. Vooral binnen educatieve doeleinde kan er veel kennis worden opgedaan met het uitbreiden van deze stack.

Wel is het belangrijk om constant in de gaten te houden wat het geheugen gebruik van de stack is. En wat nieuwe routines doen met de snelheid van de stack. Dit om te voorkomen dat het als nog een log stuk software wordt.

We hopen ondersteuning te kunnen blijven geven aan de door ontwikkeling van deze stack. Om dit te verwezenlijken is er op het moment al een software repository beschikbaar gesteld voor gebruik. En we stellen dan ook voor dat toekomstige ontwikkelaars contact op nemen met een van ons voor verdere instructies wat betreft het gebruik hiervan.

Ondanks dat er relatief veel kennis van netwerken en haar protocollen aanwezig was binnen de projectgroep. Hebben we toch nog een aanzienlijke hoeveelheid kennis opgedaan. En we hopen dat toekomstige ontwikkelaars, gebruikers, en studenten er tevens veel positiefs door ontvangen.