Radboud Universiteit

Bachelorscriptie

# Het testen van smart meters

*Auteur:*
Mark Spreeuwenberg

*Begeleider:*
Dr. Engelbert Hubbers

12 juli 2010

**Samenvatting**

Voor mijn bachelorscriptie heb ik een prototype gemaakt van een applicatie die met verschillende (types) smart meters kan communiceren. Deze applicatie kan gebruikt worden bij het testen van de meters, men wil tenslotte de functionaliteit van de meter testen zonder zich druk te hoeven maken over het formaat van het te sturen bericht. Smart meters zijn de nieuwe, digitale, energiemeters die meterstanden automatisch kunnen doorgeven. Ze maken gebruik van verschillende technieken om data te verzenden. Zo kan dit bijvoorbeeld via GPRS, ethernet of PLC. Omdat de smart meters verschillende protocollen gebruiken ben ik op zoek gegaan naar deze protocollen. Ik heb twee protocollen gevonden: COSEM en LON. Aan de hand van deze twee protocollen heb ik een opbouw voor een universeel bericht opgesteld. De structuur die ik voor het bericht heb gekozen is XML. Naast XML, heb ik ook gekeken naar andere structuren, zo had ik ook kunnen kiezen voor een structuur als EDIFACT of TLV. Uiteindelijk bleek XML toch de beste optie te zijn. Er zijn diverse standaard softwarepakketten verkrijgbaar die je kunt gebruiken voor het vertalen van berichten. Het enige dat je dan moet doen is deze pakketten op de juiste manier configureren. Helaas waren Apache ActiveMQ, Fuse Message Broker en OpenAMQ niet geschikt voor mij. Een ander pakket, Microsoft Biztalk Server, voldeed wel aan mijn eisen, het is alleen niet volledig gratis doordat er veel aanvullende software nodig is. In deze scriptie beschrijf ik hoe ik tot de opbouw van het bericht en de structuur ben gekomen en waarom ik voor deze structuur heb gekozen. Daarna leg ik ook nog uit dat het beter was om zelf een applicatie te maken omdat bestaande software niet voldoende toereikend is. Tenslotte maak ik duidelijk hoe ik de applicatie heb opgebouwd, dus hoe de database eruit ziet, hoe de modules in elkaar zitten, enzovoorts. Helaas ben ik er tijdens het maken van de applicatie achter gekomen dat er te weinig informatie beschikbaar was over het LON protocol om het goed en volledig te kunnen implementeren.

# Inhoudsopgave

# 1 Inleiding

Deze scriptie gaat over het uit uitvoeren van securitytests op smart meters (ook wel slimme meters genoemd). Ik heb onderzocht of het mogelijk is om een applicatie te ontwerpen zodat er security tests op een smart meter kunnen worden uitgevoerd, onafhankelijk van het type, en hoe deze er dan uit komt te zien.
Smart meters zijn de toekomstige energiemeters die voor een aantal voordelen moeten zorgen voor zowel de consument als de producent van energie. Zo krijgen de leveranciers meer inzicht in het verbruik en heeft de consument geen last van meteropnemers. In hoofdstuk 2 wordt beschreven wat een smart meter is en hoe deze ongeveer werkt. Doordat er meer informatie beschikbaar is voor verschillende partijen zijn er ook risico's voor de veiligheid. In hoofdstuk 3 vertel ik iets over de securityaspecten die van belang zijn bij het uitvoeren van de tests en waarom deze van belang zijn. Hoofdstuk 4 bevat vervolgens informatie over de verschillende manieren waarop er gecommuniceerd kan worden met de smart meters. Dit is namelijk niet voor iedere smart meter hetzelfde. In hoofdstuk 5 zal een structuur voor berichten worden ontworpen waarmee de berichten, onafhankelijk van het communicatieprotocol, naar de smart meters kunnen worden gestuurd. De structuur die ik in dit hoofdstuk ontwerp zal vertaald moeten worden naar een bericht, van het juiste protocol, dat een smart meter begrijpt. Hiervoor zal ik in hoofdstuk 6 naar standaard programma's kijken die dit werk voor mij kunnen doen. Omdat de resultaten hiervan tegenvielen, beschrijf ik in hoofdstuk 7 vervolgens een ontwerp van een applicatie die de berichten voor mij kan omvormen naar een bericht van het gewenste protocol. In hoofdstuk 8 beschrijf ik de implementatie van dit ontwerp en verklaar ik waarom ik bepaalde dingen op een specifieke manier heb geïmplementeerd. In hoofdstuk 9 trek ik vervolgens conclusies over de keuzes die ik gemaakt heb gedurende het schrijven van deze scriptie en het maken van de applicatie. Achteraf weet je immers vaak hoe het anders en mogelijk beter had gekund. Tenslotte wordt in hoofdstuk 10 beschreven wat iemand anders kan doen om deze applicatie uit te breiden en te verbeteren.

# 2 Inleiding smart meters

## 2.1 Wat is een smart meter?

Een smart meter is een energiemeter die op afstand bediend kan worden, bijvoorbeeld via het elektriciteitsnet of via GPRS (hierover meer in hoofdstuk 4). Meestal verwijst de term smart meter naar een elektriciteitsmeter, maar het kan ook een meter zijn voor het meten van het verbruik van water of gas. Dit zijn dan wel verschillende meters die zijn aangesloten op een centraal punt in het huis. Als een dergelijke meter in een huishouden is geïnstalleerd, kan de leverancier van de energie de meter op afstand bedienen. Zo kan hij bijvoorbeeld de meterstand opvragen of de levering van de energie stopzetten.

## 2.2 Het doel van smart meters

Er zijn verschillende redenen te noemen waarom men de smart meters wil invoeren. Zo heeft het ministerie van Economische Zaken een aantal dingen die het met de invoering wil bereiken op hun site staan [1]. Ik heb deze redenen in eigen woorden hieronder neergezet:

1. Verbetering van de administratieve processen doordat de leverancier hier meer invloed op heeft dan voorheen.

2. Verbetering van de marktwerking doordat het gemakkelijker wordt om van leverancier te wisselen.

3. De voordelen van de mogelijkheid om meters op afstand uit te lezen moeten zoveel mogelijk benut worden. Eén van de dingen waar men heel goed naar moet kijken is de mogelijkheid voor energiebesparing.

4. Een verbeterde toegang tot de meter doordat er niemand meer langs hoeft te gaan. Hierdoor kunnen er nog andere zinvolle diensten worden aangeboden, denk hierbij aan het geven van tips om energie te besparen of het in kaart brengen wanneer iemand veel energie verbruikt.

5. Kostenbesparing door efficiënter netbeheer en efficiëntere uitwisseling van data.

Naast de hierboven genoemde voordelen zitten er ook een paar mogelijke nadelen aan een invoering van de smart meter. Zo zijn ze, in tegenstelling tot de “oude” analoge meters, gevoelig voor hackers (mensen die de meters proberen te kraken om hier zelf beter van te worden). Als dit soort mensen

in staat is om op een meter in te breken kunnen ze bijvoorbeeld het energieverbruik van de eigenaar zien. Dit is uiteraard gevoelige informatie die niet in de verkeerde handen mag vallen. Als je deze informatie gedurende een langere periode bekijkt kun je er namelijk een patroon uit afleiden. Je kunt uit dit patroon bijvoorbeeld aflezen wanneer iemand wel of niet thuis is. Dit is natuurlijk handige informatie voor een inbreker. Echter, deze informatie kan verder ook nog gebruikt worden bij het oplossen van misdrijven [22, 20]. Je wilt dus niet dat deze informatie zomaar bij iedereen terecht kan komen.

## 2.3 Werking

Smart meters registreren het elektriciteitsverbruik één keer per 15 minuten, bij water en gas is dit één keer per 60 minuten [20]. Omdat deze meters op afstand kunnen worden uitgelezen is er natuurlijk ook een communicatiemethode nodig. De meest gebruikte methodes hiervoor zijn GPRS en PLC (Power Line Communication). Verder wordt er nog onderzoek gedaan naar eventueel andere vormen van communicatie. De gegevens die verstuurd worden zijn ook zichtbaar voor de consument zelf waardoor deze zelf meer inzicht krijgt in zijn verbruik. Dit is bijvoorbeeld bij Oxxio het geval, deze leverancier geeft iedere klant een login voor een persoonlijke webpagina waarop deze zijn verbruik van de verstreken tijd kan zien. Op deze manier kan de klant zelf zien wanneer hij veel energie verbruikt en hier, indien gewenst, verandering in brengen om zo weer een beetje energie (en dus ook kosten) te besparen.

## 2.4 Voortgang binnen Nederland

De afgelopen jaren zijn er al talloze smart meters verspreid. In 2005 heeft energiemaatschappij Oxxio de eerste smart meters geplaatst. Deze meters waren geschikt voor het meten van zowel elektriciteit en gas. In de brochure van Oxxio over de smart meter staat vermeld dat Oxxio sinds de invoering in 2005 al meer dan 100 duizend meters heeft geplaatst, helaas wil Oxxio geen mededeling over de precieze aantallen doen. De Volkskrant meldde in een artikel van 6 maart 2008, dat Oxxio toen 140 duizend meters in 75 duizend huishoudens had geplaatst. Continuon, het onderdeel van Nuon dat het netbeheer van de energieproducent op zich neemt, zou 75 duizend meters bij 45 duizend huishoudens hebben geplaatst [2]. Op 4 juli 2008 stemde de Tweede Kamer ermee in dat iedereen voor 2015 een dergelijke meter moet hebben [3]. Het uitdelen van de meters is echter tot op heden vertraagd. De belangrijkste oorzaken hiervan zijn de vragen van consumentenbonden wat betreft de privacy. Verder was er niet voldoende rekening gehouden met mensen die energie terugleveren aan het net (bijvoorbeeld door het gebruik

van zonnepanelen). Bij de plaatsing van de eerste meters golden er nog geen technische eisen. Hierdoor verschillen de meters erg van elkaar op het gebied van security en functionaliteit. Op 30 april 2007 zijn er echter wel eisen vastgelegd in de NTA-8130 standaard waaraan de meters moeten voldoen [4]:

- Op afstand kunnen uitlezen van afgenomen en aan het net teruggeleverde energie (meten)
  - Verbetering van de operationele bedrijfsvoering van de leverancier (wisselen van leverancier, verhuizen, facturering)
  - Verbetering inzicht afnemer in actuele energieverbruik en kosten
- Het op afstand kunnen aan- en afschakelen van capaciteit (schakelen)
  - Vergemakkelijken van het netbeheer door de beheerder
  - Preventief uitschakelen tijdens noodsituaties
  - Aan- en uitschakelen bij in- en uithuizen van kleinverbruikers (bij tijdelijke leegstand)
  - Gedeeltelijk uitschakelen bij wanbetaling
  - Ondersteuning van bepaalde betalingsmethoden zoals prepaid
- Het op afstand kunnen meten en signaleren van de kwaliteit van de energieafname (signaleren)
  - Verbetering operationele bedrijfsvoering netbeheerder
  - Maakt de detectie van lekkages of fraude mogelijk
  - Detectie van leveringsfluctuaties (verschillen in levering), onderbrekingen, schakeleffecten
- Online interactie tussen afnemers en leveranciers (communicatie)
  - Online aanbieden van innovatieve producten en diensten (energiebesparingsadviezen, bijzondere tarieven voor bepaalde uren)
  - Afnemer kan realtime reageren op markt/product/prijsontwikkelingen van leveranciers
  - Ondersteuning van bepaalde betalingsmethoden zoals prepaid
- Snelle reactie van regelaars in energie-installaties (regelen)
  - Koppeling met domotica-toepassingen (huisautomatisering)

- Koppeling met decentrale (duurzame) opwekking van energie
- Afstemmen van eigen productie en afname op gunstige tarieven

Hoewel het wel de bedoeling was, zijn smart meters op dit moment nog niet verplicht. Hier zal echter wel verandering in gaan komen als er is aangetoond dat deze naar behoren werken en de privacy en andere security aspecten gegarandeerd kunnen worden. Echter, wie nu al een smart meter wil, kan deze wel aanvragen. Smart meters zijn namelijk wel beschikbaar op vrijwillige basis. Dit is besloten op 8 april 2009 toen een voorstel voor de verplichte invoer van de meter, dat al door de Tweede Kamer was goedgekeurd, werd afgewezen in de Eerste Kamer [3]. De Eerste Kamer vond net als de consumentenbond dat de meters op gespannen voet staan met de privacyregels [5].

# 3 Securityaspecten

Eén van de grootste problemen voor het garanderen van de security, is de levensduur van een smart meter. Een dergelijke meter heeft een gemiddelde levensduur van zo'n vijftien jaar. Als een systeem een korte levensduur heeft, is er ook minder tijd om het systeem te hacken omdat het anders alweer vervangen is tegen de tijd dat de sleutels gekraakt zijn. Bij smart meters daarentegen, kun je er gerust vijf jaar over doen om het systeem te hacken, want vervolgens heb je nog zo'n tien jaar over om van de voordelen te genieten. De situatie waarin dit zou kunnen gebeuren is helaas niet ondenkbaar, de technologische ontwikkelingen gaan immers snel. Om dit te voorkomen moet er dus ook een mogelijkheid worden ingebouwd om de firmware te updaten. Uiteraard brengt dit weer andere securityrisico's met zich mee, je moet er immers voor zorgen dat niemand zonder toestemming de firmware naar eigen inzicht kan aanpassen. Als een systeem wordt ingevoerd is het natuurlijk belangrijk dat het doet wat het moet doen. Naast het goed functioneren van een systeem is het natuurlijk ook belangrijk dat het veilig is zodat er geen misbruik van kan worden gemaakt. Bij het uitvoeren van securitytests op smart meters zijn er drie aspecten van security van belang:

- Confidentiality
- Integrity
- Availability

Ik heb ervoor gekozen om de Engelse termen te gebruiken in deze scriptie. De Nederlandse benamingen zijn respectievelijk confidentialiteit, integriteit en beschikbaarheid.

## 3.1 Confidentiality

Confidentieel is een ander woord voor vertrouwelijk. Confidentiality houdt dus in dat informatie van een bepaald persoon, niet zichtbaar wordt voor andere personen die niet bevoegd zijn om deze informatie te bekijken.
In het geval van smart meters houdt dit in dat de informatie, over onder andere de meterstanden, niet zichtbaar mag zijn voor andere mensen als deze niet gecodeerd is. Als dit wel het geval is kan de privacy ernstig in het geding komen. Zo kan men door middel van het monitoren van de berichten die van en naar de meter gaan, een patroon afleiden. Uit een dergelijk patroon kun je bijvoorbeeld afleiden wanneer iemand wel thuis is, en wanneer iemand

meestal van huis is. Dit is natuurlijk handige informatie voor een inbreker, deze vindt het fijner om in te breken als hij denkt dat er niemand aanwezig is. Deze informatie kan echter ook gebruikt worden bij "legale" activiteiten. Doordat je kunt zien of iemand wel of niet thuis is kan dit bijvoorbeeld ook gebruikt worden bij het oplossen van misdrijven. Om privacy van de klant te kunnen garanderen is het testen van de confidentiality dus van groot belang. De belangrijkste reden dat de smart meter nog niet is ingevoerd, heeft te maken met de confidentiality. Sommige groepen mensen twijfelen er nog aan of het geplande systeem zoals het nu is, de privacy wel kan garanderen. Andere groepen mensen twijfelen zelfs of het systeem ooit zal gaan werken zoals het bedoeld is. Dit is een probleem dat erg speelt onder de consumenten, wat bijvoorbeeld blijkt uit de poster van figuur 1 en de website `http://www.wijvertrouwenslimmemetersniet.nl`.


Figuur 1: Poster waaruit het wantrouwen blijkt

## 3.2 Integrity

Integrity is de eigenschap dat informatie gegarandeerd correct is. Bij het verzenden van informatie is het dus van belang dat de informatie die verstuurd wordt ook daadwerkelijk aankomt. De informatie mag onderweg niet kwijt raken of veranderd worden door mensen die hier geen toestemming voor hebben.

Naast confidentiality is integrity ook een belangrijke eigenschap bij smart meters. Echter, in tegenstelling tot confidentiality, is integrity van belang voor zowel de klant als de leverancier van energie. Beide partijen willen namelijk niet dat de informatie onderweg kwijt raakt of verandert. Als de meterstanden onderweg namelijk verhoogd worden, is de consument hier niet blij mee omdat deze dan een te hoge rekening gepresenteerd krijgt. Als de meterstanden onderweg verlaagd worden is de leverancier hier niet blij mee

omdat deze dan inkomsten misloopt. Om te kunnen garanderen dat een factuur (en overige informatie) klopt, is het van wezenlijk belang dat de integrity behouden blijft.

## 3.3 Availability

Net als integrity is availability ook een belangrijke eigenschap voor zowel de klant als de leverancier. Availability betekent in het geval van de smart meters dat de diensten altijd beschikbaar zijn indien hierom gevraagd wordt. Voor de klant houdt dit in dat deze altijd de beschikking heeft over elektriciteit, water en gas. Aangezien de meters op afstand kunnen worden bediend loop je het risico dat iemand zonder toestemming één of meerdere huishoudens zonder geldige reden afsluit. Voor de leverancier houdt de availability eis in dat hij altijd de meterstanden moet kunnen opvragen. Zou dit niet het geval zijn, dan loopt deze het risico om inkomsten mis te lopen.
Een ander probleem dat er nog niet was bij het "oude" systeem, is het aan- of afsluiten van een heleboel huishoudens tegelijkertijd. Als je namelijk in één keer een hele groep huishoudens afsluit, kan het zijn dat er ineens te veel energie in het netwerk is. Dit kan dan weer leiden tot schade aan het netwerk zelf. Hetzelfde geldt voor het aansluiten van een grote groep. Dit kan tot gevolg hebben dat er op dat moment te weinig energie in het netwerk aanwezig is [20].
Om er zeker van te zijn dat een systeem altijd werkt, moet je ook de availability van het systeem goed implementeren en testen.

## 3.4 Testen

Zoals gezegd zijn confidentiality, integrity en availability, voor zowel de klant als de leverancier, erg belangrijke eigenschappen voor het systeem met de smart meters. Vanwege de belangen van de verschillende partijen moeten deze eigenschappen goed getest worden.
Confidentiality kun je testen door berichten te versturen en vervolgens te controleren of je hier informatie uit kunt halen. Als je zelf niet in staat bent om de informatie uit de berichten te halen kun je ook nog mensen vragen die ervaren zijn met het kraken van systemen. Als ook deze mensen geen informatie uit de berichtenstroom kunnen halen, kun je ervan uitgaan dat het systeem veilig is. Zeker weten doe je dit echter nooit omdat er voor de encryptie van data meestal technieken worden gebruikt waarvan men ervan uit gaat dat niemand ze kan kraken. Als echter blijkt dat een dergelijke techniek toch gekraakt kan worden, valt de hele beveiliging van het systeem in het water.

Naast confidentiality moet dus ook integrity getest worden. Dit kun je doen door berichtjes te sturen, en te kijken of iemand deze kan openen om vervolgens de inhoud te veranderen. Waar men echter ook rekening mee moet houden, is een zogenaamde replay attack. Met een dergelijke aanval probeert iemand een eerder verzonden bericht nog een keer te versturen. Dit kan zeer handig zijn als je bijvoorbeeld de meterstand van de keer daarvoor opstuurt, op deze manier heb je een verbruik van nul. In dit extreme geval zullen er bij de energiemaatschappij natuurlijk wel een paar alarm bellen gaan rinkelen omdat je bijna altijd wel een beetje energie verbruikt, ook al ben je niet thuis. Je kunt je echter ook voorstellen dat je een berichtje stuurt met de meterstand van bijvoorbeeld de buurvrouw, om zo een lager energieverbruik na te bootsen.
Availability kun je testen door het systeem eerst eens een tijd in werking te stellen en te kijken of het bruikbaar blijft. Als dat het geval is kun je kijken of het systeem overeind blijft als je op eens heel veel berichtjes tegelijk wilt versturen. Als het systeem dan plat gaat is de zogenaamde Denial-of-service (DoS) attack succesvol en heeft de aanvaller zijn doel bereikt.
Voor het testen van de confidentiality en integrity, moet je dus eigenlijk alle mogelijke berichten een keer versturen. Je moet immers van alle berichten weten of men hier mogelijk informatie uit kan halen, en of men de inhoud van deze berichten kan veranderen zonder dat iemand anders het door heeft. Om te controleren wat er gebeurt als er in één keer een grote groep wordt aan- of afgesloten zijn de berichten die te maken hebben met het aan- en afsluiten van belang.

# 4 Communicatie met smart meters

## 4.1 Communicatietechnieken

Er zijn verschillende manier om te communiceren met de smart meters. Echter, de manier waarop er gecommuniceerd wordt, kun je niet zelf kiezen. Dit is afhankelijk van het type meter. Iedere fabrikant gebruikt zijn eigen manier van communiceren. Het verschil zit niet alleen in het gebruikte protocol, maar ook in de manier waarop berichtjes verstuurd worden. De NTA staat drie vormen van communicatie met de meters toe [20]:

- Power Line Communication (PLC)
- General Packet Radio Service (GPRS)
- Ethernet

In deze scriptie zal ik niet beschrijven hoe deze verschillende technieken allemaal precies werken omdat dit niet binnen mijn onderzoek valt. Alle protocollen moeten namelijk dezelfde berichten verzenden, onafhankelijk van het medium. Er zijn ook bedrijven die meerdere technieken gebruiken zoals PLC en GPRS. Dit komt omdat sommige technieken niet overal mogelijk zijn. Zo is PLC niet altijd mogelijk in oude gebouwen vanwege de slechte kwaliteit van de bekabeling. GPRS daarentegen, heeft weer te maken met problemen met de dekking van het netwerk.

### 4.1.1 PLC

Power Line Communication, ook wel Power Line Carrier genoemd, is een vorm van het versturen van data via het elektriciteitsnet. Deze vorm van communicatie wordt al een hele tijd gebruikt, bijvoorbeeld voor het aan- en uitzetten van de lantaarnpalen, en voor babyfoons die via het lichtnet werken. Een misschien wel bekendere vorm van PLC is het HomePlug systeem. HomePlug is een technologie die computers in een huis met elkaar kan verbinden via het elektriciteitsnetwerk. Deze techniek wordt vaak gebruikt in huizen waarin het draadloos netwerk geblokkeerd wordt door bijvoorbeeld staal. Hoewel sniffing (het opvangen van verzonden data) nu nog lastig is, is het niet uit te sluiten, zeker niet als we een paar jaar verder zijn. Tegenwoordig zijn er immers al standaard modems om de signalen op te vangen. Het probleem hiermee is alleen dat het nu nog lastig is om de juiste combinatie van hardware en software te vinden. Echter, omdat een smart meter zo'n vijftien jaar mee moet gaan, kun je je voorstellen dat het over een paar jaar wel makkelijk zal worden. Je zult dus security tests moeten doen om de confidentiality en de integrity te kunnen garanderen [6].

### 4.1.2 GPRS

GPRS staat voor General Packet Radio Service en is een techniek die een uitbreiding is op het bestaande gsm-netwerk. Deze techniek maakt het mogelijk om efficiënter, sneller en goedkoper data te kunnen versturen. Men spreekt tegenwoordig vaak over GPRS-netwerken maar dit zijn eigenlijk gewone gsm-netwerken. Je kunt GPRS signalen relatief makkelijk opvangen. Zolang je de data goed versleutelt hoeft dit natuurlijk geen probleem te zijn. Echter, als deze encryptie faalt, is ook GPRS niet meer veilig.

### 4.1.3 Ethernet

Ethernet kan gebruikt worden voor communicatie via het internet. Naast het feit dat dit vandaag de dag al aanwezig is in de meeste huishoudens, is dit ook een aantrekkelijke vorm voor hackers. Er zijn immers genoeg tools om de signalen op te vangen en te manipuleren. Voorbeelden van deze tools zijn:

- Wireshark [7]
- Aircrack-ng [8]
- Tcpdump [9]
- Microsoft Network Monitor [10]
- Netscout [11]

## 4.2 Protocollen

Er zijn verschillende protocollen die worden gebruikt voor communicatie met smart meters. In Nederland worden er momenteel ook meerdere protocollen gebruikt, waarvan er twee bij mij bekend zijn:

- Local Operation Network (LON)
- DLMS/COSEM

Het kan best zijn dat er nog meer protocollen worden gebruikt. Echter, de energiemaatschappijen willen hier geen informatie over vrijgeven dus dit heb ik niet verder kunnen onderzoeken. De maatschappijen (zowel energieleveranciers als netbeheerders) die ik heb benaderd met vragen over de smart meters zijn:

1. Oxxio

2. Eneco
3. Essent
4. Enexis
5. Nuon
6. Alliander
7. Nederlandse Energie Maatschappij

Van deze maatschappijen hebben alleen de eerste drie gereageerd. Het antwoord was in alle gevallen van dezelfde aard, namelijk: "Hierover verstrekken wij geen informatie aan derden". In het geval van Oxxio kwam deze reactie via de telefoon en heeft men vervolgens nog een brochure opgestuurd. Echter, deze brochure stond ook gewoon op de site en bevatte helaas niet de gewenste informatie. Van de overige maatschappijen heb ik nooit een reactie ontvangen, ook niet nadat ik ze nog een tweede en derde keer benaderd had.

### 4.2.1 LON

LON is een protocol dat ontwikkeld is door Echelon. Helaas is dit protocol niet publiekelijk beschikbaar waardoor het lastig is hier informatie over te krijgen. Buiten Echelon heb ik geen andere fabrikant kunnen vinden die dit protocol gebruikt. Een bericht in dit protocol ziet er uit zoals in tabel 1 [19, 18]:

| BitSync | ByteSync | L2Hdr | NPDU | CRC |
|---|---|---|---|---|

Tabel 1: Indeling LON bericht

**BitSync en ByteSync** Het BitSync en ByteSync deel is een soort inleiding die afhankelijk is van het medium.

**L2Hdr** Dit veld is weer opgebouwd uit drie subvelden en ziet eruit zoals weergegeven in tabel 2.

| Prior | AltPath | Delta_BL |
|---|---|---|

Tabel 2: Indeling L2Hdr

- **Prior:** Een veld van één bit dat de prioriteit aangeeft. De waarde 0 geeft een normale prioriteit weer en de waarde 1 geeft een hoge prioriteit weer.
- **AltPath:** Een veld van één bit dat aangeeft welk kanaal moet worden gebruikt. Dit veld is er voor apparaten die kunnen versturen over twee verschillende kanalen en kunnen ontvangen over één van de twee.
- **Delta_BL:** Dit veld bestaat uit zes bits (unsigned). Dit veld specificeert de "channel backlog increment" die gegenereerd wordt als resultaat van het afleveren van dit bericht.

**NPDU** Het NPDU deel ziet er dan weer uit zoals in tabel 3. Dit veld bevat de daadwerkelijke informatie die verzonden wordt en is weer samengesteld uit zeven andere velden.

| Version | PDU Fmt | AddrFmt | Length | Address | Domain | encl.PDU |
|---|---|---|---|---|---|---|

Tabel 3: Indeling NPDU

- **Version:** De versie van het protocol. Dit veld heeft een lengte van twee bits.
- **PDU Fmt:** Dit staat voor PDU Format en dit veld bestaat uit twee bits en specificeert het type PDU (Protocol Data Unit) dat wordt meegestuurd. Op deze manier is er al bekend welk type PDU er in het enclosed PDU veld zit. De lengte van dit veld geeft aan dat er maar vier mogelijke PDU's zijn, omdat je met twee bits maar vier getallen kunt representeren. Dit blijkt ook te kloppen, want er zijn precies vier mogelijke PDU's, namelijk een TPDU, SPDU, AuthPDU of APDU.

  **TPDU:** Dit staat voor Transport Protocol Data Unit. Zoals de naam al zegt zorgt deze PDU ervoor dat de berichten goed verstuurd worden.

  **SPDU:** Dit staat voor Session Protocol Data Unit. Deze PDU zorgt voor het goed verlopen van een communicatie sessie. Zo zorgt dit ervoor dat een vraag en antwoord goed op elkaar aansluiten.

  **AuthPDU:** Dit staat voor Authentication Protocol Data Unit. Zoals de naam al doet vermoeden wordt deze PDU gebruikt voor authenticatie.

**APDU:** Dit staat voor Application Protocol Data Unit. Deze PDU wordt gebruikt voor het verzenden van willekeurige berichten.

- **AddrFmt:** Dit staat voor Address Format en dit veld heeft een lengte van twee bits en geeft aan hoe het adresveld (dat later in het bericht volgt) eruit ziet. Ook hier kunnen er weer maximaal vier mogelijke indelingen zijn voor het adresveld. Ook dit is hier het geval.
- **Length:** Dit veld bestaat uit twee bits en geeft de lengte aan van het domein veld dat nog moet komen.
- **Address:** Het adresveld kan verschillende vormen aannemen. Het veld AdrrFmt geeft aan hoe het adresveld er in een specifiek bericht uit ziet. Hoe de verschillende vormen eruit zien, is te zien in figuur 11 in appendix B.
- **Domain:** Dit veld heeft een lengte van 0 t/m 3 bytes en bevat het domein waarbinnen het bericht verstuurd wordt.
- **Encl.PDU:** Dit staat voor Enclosed PDU en dit veld heeft geen vaste lengte en kan verschillende PDU's bevatten. Zo kan het, zoals eerder vermeld, een TPDU, SPDU, AuthPDU of APDU bevatten. Hoe de verschillende vormen eruit zien, is te zien in figuur 11 in appendix B. De PDU die hierin zit bevat de daadwerkelijke informatie van het bericht.

**CRC** De lengte van dit veld is zestien bits, oftewel twee bytes. Dit veld bevat een check over de Prior, AltPath, Delta_BL en NPDU velden.

### 4.2.2 DLMS/COSEM

DLMS/COSEM is een open protocol voor communicatie met smart meters. DLMS staat voor "Device Language Message specification" en COSEM staat voor "COmpanion Specification for Energy Metering". DLMS is een algemeen model voor het abstract modelleren van communicaties. COSEM stelt de regels voor data-uitwisseling van energiemeters, gebaseerd op bestaande standaarden [12]. Dit protocol wordt gebruikt door, in ieder geval, de volgende fabrikanten:

- Iskraemeco
- Brinck

Een bericht in dit protocol ziet er binnen de MAC sub-layer uit zoals in tabel 4 is weergegeven [17]. De onderlinge samenhang van de componenten is zichtbaar in appendix C.

| Flag | Frame format | Dest. address | Src. address | Control | HCS | Information | FCS | Flag |
|---|---|---|---|---|---|---|---|---|

Tabel 4: HDLC frame format type 3 van een DLMS/COSEM bericht

**Flag field** De lengte van dit veld is één byte. Als er meerdere frames tegelijk worden verstuurd, wordt er maar één flag gebruikt als sluitingsvlag van het huidige frame, en als openingsvlag van het volgende frame. Dit veld heeft altijd de waarde 7E.

**Frame format field** Het frame format veld heeft een lengte van twee bytes en bestaat uit drie subvelden. Het eerste subveld heeft een lengte van vier bits en heet het format type subveld. Het tweede subveld heet het segmentatie subveld en heeft een lengte van één bit en het derde subveld is het frame length subveld en is elf bits lang. De waarde van dit subveld is het aantal bytes in het frame zonder de twee flags.

**Destination and source address fields** Er zijn altijd twee adresvelden in een frame, namelijk één voor de geadresseerde, en één voor de afzender. Het adres van de afzender is altijd 1 byte lang, het adres van de geadresseerde kan één, twee of vier bytes lang zijn.

**Control field** De lengte van dit veld is één byte. Het geeft aan wat voor soort bericht het is.

**Header check sequence (HCS) field** Dit veld is optioneel en heeft een lengte van twee bytes. De check wordt uitgevoerd op de header (de bits tussen de openingsvlag en het HCS-veld). Frames die geen informatieveld of een leeg informatieveld hebben, hebben ook geen HCS-veld.

**Information field** Het informatie veld kan elke lengte hebben. Dit veld bevat de informatie die je met een bericht wilt versturen. Dit veld mag ook leeg zijn, dit kan handig zijn bij het versturen van controle berichten om te kijken of alles nog naar behoren werkt. Als het frame een data-frame is, dan bevat het de MSDU (Mac Service Data Unit). Anders is het veld niet aanwezig. De MSDU is het deel dat de daadwerkelijke informatie van een bericht bevat.

**Frame check sequence (FCS) field** De lengte van het FCS-veld is twee bytes. Deze check wordt, tenzij anders vermeld, berekend voor de hele lengte van het frame, uitgezonderd de openingsvlag en de FCS.

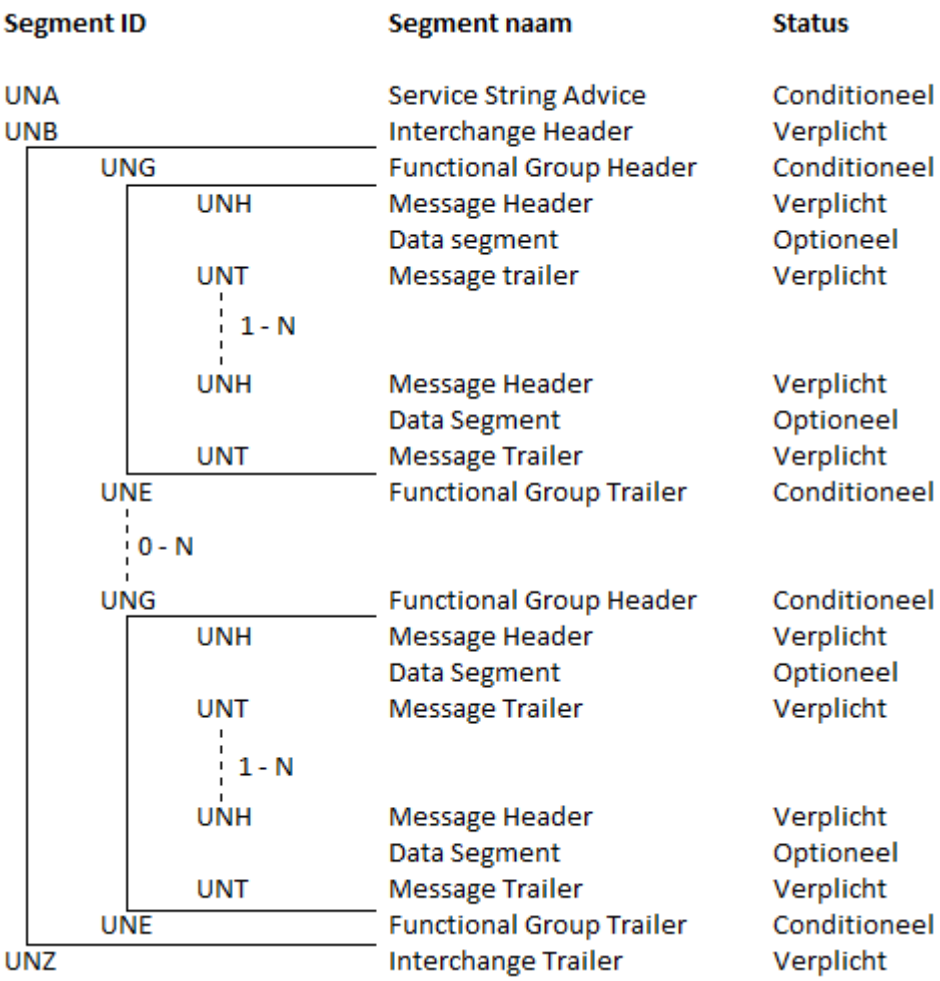
# 5 Berichtenstructuur

Voor het versturen van berichten naar de smart meters, heb je natuurlijk een structuur voor de berichten nodig. Omdat er verschillende protocollen worden gebruikt, zul je een algemene structuur moeten maken als je berichten naar de smart meter wilt sturen zonder te weten welk protocol deze gebruikt. Voor een dergelijke structuur zijn er verschillende mogelijkheden waaruit je kunt kiezen. Ik heb gekeken naar:

- EDIFACT (Electronic Data Interchange For Administration, Commerce and Transport)
- TLV (Type-Length-Value)
- XML (Extensible Markup Language)

## 5.1 EDIFACT

EDIFACT is een standaard voor elektronische gegevensuitwisseling die is opgezet door de Verenigde Naties. Het doel hiervan was de Europese en Amerikaanse EDI standaarden te synchroniseren om tot een universeel geaccepteerde standaard te komen [13]. De standaard beschrijft de opmaak van een bericht dat gebruikt wordt bij gegevensuitwisseling tussen ondernemingen, met als doel het schriftelijke verkeer bij handelstransacties tot het minimum te beperken. Deze standaard wordt voornamelijk binnen Europa gebruikt, maar wordt ook steeds meer als wereldwijde standaard geaccepteerd. De EDIFACT syntax regels bepalen de standaard voor het structureren van data in segmenten, segmenten in berichten en berichten in een interchange. Een interchange heeft een standaard opbouw zoals deze wordt weergegeven in figuur 2. Een interchange is eigenlijk een soort van pakket met daarin verschillende berichten die in één keer worden verstuurd. Zoals in het plaatje te zien is, kunnen berichten gegroepeerd worden in een groep. Binnen deze groep hebben de berichten dan met elkaar te maken. Echter, berichten hoeven niet noodzakelijk in een groep te worden gestopt, dit is optioneel. De status van de UNG is conditioneel omdat deze samen met de UNE moet voorkomen. Als de een voorkomt, moet ook de ander voorkomen. Verder kunnen er binnen de UNH en UNT ook meerdere data segmenten voorkomen.




Figuur 2: Standaard structuur van een interchange

Een voorbeeld van een EDIFACT-document dat een antwoord geeft op een aanvraag van beschikbaarheid van een product binnen de luchtvaartsector:

```
1. UNB+IATB:1+6XPPC+LHPPC+940101:0950+1'
2. UNH+1+PAORES:93:1:IA'
3. MSG+1:45'
4. IFT+3+XYZCOMPANY AVAILABILITY'
5. ERC+A7V:1:AMD'
6. IFT+3+GEEN VLUCHTEN BESCHIKBAAR'
7. ODI'
8. TVL+240493:1000::1220+FRA+JFK+DL+400+C'
```

```
 9.  PDI++C:3+Y::3+F::1'
10.  APD+74C:0:::6++++++6X'
11.  TVL+240493:1740::2030+JFK+MIA+DL+081+C'
12.  PDI++C:4'
13.  APD+EM2:0:1630::6+++++++DA'
14.  UNT+13+1'
15.  UNZ+1+1'
```

Normaliter wordt de tekst zonder regelnummers en direct achter elkaar geschreven, maar in dit voorbeeld is de tekst voor de duidelijkheid in regels verdeeld.
Ik zal het voorbeeld gebruiken om uit te leggen hoe de structuur van het bericht in elkaar zit. Regel 1 t/m 15 zijn als geheel een interchange, regel 2 t/m 14 zijn in zijn geheel een bericht en de regels 3 t/m 13 zijn elk afzonderlijk een segment dat data bevat.
Ik zal nu regel 8 gebruiken om aan te geven hoe een segment is opgebouwd. De data in een segment bestaat uit verschillende elementen en wordt gescheiden door een +. De data in deze regel bestaat dus uit zeven elementen. Een element kan ook weer worden opgesplitst in componenten, hiervoor wordt een : gebruikt. Het tweede element uit regel 8 bestaat dus ook weer uit vier, al dan niet lege, componenten. Uiteindelijk wordt ieder segment door een ' afgesloten.
Hierboven hebben we gekeken naar de opbouw van een segment. Hieronder zal ik nog van een paar voorbeelden bespreken hoe deze regels zijn opgebouwd.
UNH+1+PAORES:93:1:IA'. Dit is een header segment dat verplicht is aan het begin van een bericht. Deze code geeft aan dat de naam van het bericht en de versie PAORES 93 versie 1 is, en dat deze gedefinieerd is door de organisatie IA (IATA).
IFT+3+GEEN VLUCHTEN BESCHIKBAAR'. Dit is een "Interactive Free Text" dat de tekst "GEEN VLUCHTEN BESCHIKBAAR" overdraagt. De 3 is ervoor om aan te geven over welk onderwerp de tekst gaat.
UNT+13+1. Dit is een controle-element dat aangeeft dat het bericht uit dertien segmenten bestaat [14].
Als je echter de specificatie van de TVL tag uit appendix A gebruikt valt er iets op. Als je namelijk de componenten invult, dan zie je dat FRA een "location name code" is, maar JFK is opeens een "party name". Doordat er maar één locatie kan komen te staan, moet men een trucje verzinnen om toch de JFK erin te stoppen.

## 5.2 TLV

TLV is een standaard voor gegevensuitwisseling waarbij een bericht uit drie delen bestaat:

**Type:** een code die aangeeft wat voor soort bericht het is.

**Length:** de grootte van het value deel.

**Value:** het deel dat de gegevens van het bericht bevat (variabel van grootte).

Hieronder volgt een voorbeeld van een TLV bericht voor het voorbeeld zoals gebruikt bij EDIFACT. Een tag bestaat altijd uit 3 tekens en het veld waarin de lengte wordt opgeslagen, bestaat in dit voorbeeld altijd uit 4 decimale karakters, als het getal minder dan 4 karakters bevat, dan wordt het aangevuld met voorloopnullen. Op deze manier weet je altijd waar de data begint, en waar er weer een nieuwe tag moet beginnen. Dit komt doordat je weet hoeveel bytes je moet lezen om een tag te lezen, vervolgens weet je ook nog het aantal bytes dat je moet lezen om de lengte van de data te weten. Doordat je net de lengte van de data hebt gelezen, weet je ook waar deze ophoudt en waar dus weer een nieuwe tag moet beginnen (als die er nog is). De data die ik bij het EDIFACT voorbeeld niet kon herkennen, heb ik bij dit voorbeeld letterlijk overgenomen. Voor dit voorbeeld heb ik een aantal nieuwe tags moeten verzinnen:

**UBD:** bevat de data die in de UNB tag zat.

**UHD:** bevat de data die in de UNH tag zat.

**DAT:** bevat de datum van de vlucht.

**TIV:** bevat de vertrektijd van de vlucht.

**TIA:** bevat de aankomsttijd van de vlucht.

**PLV:** bevat de plaats van waaruit gevlogen wordt.

**PLN:** bevat de plaats waarnaar gevlogen wordt.

**VLU:** bevat het vluchtnummer van de vlucht.

Voor het herkennen van de data uit het EDIFACT bericht heb ik gebruik gemaakt van de specificatie van een TVL tag. Deze specificatie is opgenomen in appendix A.

```
UNB0402UBD0032IATB:1+6XPPC+LHPPC+940101:0950+1
       UNH0356UHD00161+PAORES:93:1:IA
              MSG00041:45
              IFT00253+XYZCOMPANY AVAILABILITY
              ERC0009A7V:1:AMD
              IFT00273+GEEN VLUCHTEN BESCHIKBAAR
              ODI0000
              TVL0068DAT0006240493
                     TIV00041000
                     TIA00041220
                     PLV0003FRA
                     PLN0003JFK
                     VLU0006DL400C
              PDI0013C:3+Y::3+F::1
              APD001774C:0:::6++++++6X
              TVL0068DAT0006240493
                     TIV00041740
                     TIA00042030
                     PLV0003JFK
                     PLN0003MIA
                     VLU0006DL081C
              PDI0003C:4
              APD0022EM2:0:1630::6+++++++DA
```

Hetzelfde voorbeeld ziet er zonder opmaak alsvolgt uit:

```
UNB0402UBD0032IATB:1+6XPPC+LHPPC+940101:0950+1UNH0356UHD00161
+PAORES:93:1:IAMSG00041:45IFT00253+XYZCOMPANY AVAILABILITYERC
0009A7V:1:AMDIFT00273+GEEN VLUCHTENBESCHIKBAARODI0000TVL0068D
AT0006240493TIV00041000TIA00041220PLV0003FRAPLN0003JFKVLU0006
DL400CPDI0013C:3+Y::3+F::1APD001774C:0:::6++++++6XTVL0068DAT0
006240493TIV00041740TIA00042030PLV0003JFGPLN0003MIAVLU0006DL0
81CPDI0003C:4APD0022EM2:0:1630::6+++++++DA
```

## 5.3 XML

XML wordt steeds meer gebruikt als alternatief voor EDIFACT. Hoewel de XML-bestanden groter zijn, is het door de structuur duidelijker om te lezen. EDIFACT is wijd verspreid en wordt dus veel ingezet, waardoor de vervanging van EDIFACT door XML niet of pas op de lange duur zal plaatsvinden. XML is een standaard waarmee je berichten gestructureerd kunt weergeven. Deze berichten zijn leesbaar voor zowel computers als mensen. Je kunt XML

gebruiken voor het opslaan van data en voor het sturen van data over een netwerk. Nu volgt hoe je het eerder gebruikte voorbeeld kunt structureren met XML:

```
<UNB lengte=1>
  <UNH lengte=11>
    <MSG>1:45</MSG>
    <IFT onderwerp=3>XYZCOMPANY AVAILABILITY</IFT>
    <ERC>A7V:1:AMD</ERC>
    <IFT onderwerp=3>GEEN VLUCHTEN BESCHIKBAAR</IFT>
    </ODI>
    <TVL>
      <DATUM>240493</DATUM>
      <VERTREKTIJD>1000</VERTREKTIJD>
      <AANKOMSTTIJD>1220</AANKOMSTTIJD>
      <VAN>FRA</VAN>
      <NAAR>JFK</NAAR>
      <VLUCHTNUMMER>DL400C</VLUCHTNUMMER>
    </TVL>
    <PDI>C:3+Y::3+F::1</PDI>
    <APD>74C:0:::6++++++6X</APD>
    <TVL>
      <DATUM>240493</DATUM>
      <VERTREKTIJD>1740</VERTREKTIJD>
      <AANKOMSTTIJD>2030</AANKOMSTTIJD>
      <VAN>JFK</VAN>
      <NAAR>MIA</NAAR>
      <VLUCHTNUMMER>DL081C</VLUCHTNUMMER>
    </TVL>
    <PDI>C:4</PDI>
    <APD>EM2:0:1630::6+++++++DA</APD>
  </UNH>
</UNB>
```

Zoals je kunt zien is dit een zeer overzichtelijke en leesbare manier van datarepresentatie. Doordat EDIFACT een bepaalde, vaste structuur heeft, kun je de UNT vervangen door een </UNH>. De lengte die in de UNT tag zat, kun je nu als attribuut meegeven aan de UNH tag, hierdoor heb je toch nog een stukje extra controle.

## 5.4 Gekozen structuur

De structuur die ik voor de berichten heb gekozen, is XML. Ik heb hiervoor gekozen omdat dit type naar mijn mening de meest overzichtelijke structuur heeft. Vanwege de minder duidelijke opbouw van EDIFACT heb ik hier niet voor gekozen. Daarbij komt ook nog eens dat XML steeds meer wordt gebruikt als alternatief voor EDIFACT. Verder kun je de "Type", "Length" en "Value" delen, waaruit een TLV bericht bestaat, eenvoudig nabootsen met een XML structuur. Dit ziet er dan ongeveer als volgt uit:

```
<TLV>
 <TYPE>...</TYPE>
 <VALUE>...</VALUE>
</TLV>
```

Merk op dat je hier geen lengte meer hoeft op te geven, dit komt doordat XML gebruik maakt van sluit-tags. Mocht je de lengte echt nodig hebben, dan kun je deze nog altijd als attribuut meegeven in de TLV tag.
Verder biedt XML, in tegenstelling tot EDIFACT, de mogelijkheid om zelf elementen te definiëren. Een ander voordeel van XML is dat je niet gebonden bent aan een vaste lengte van de tags, je kunt deze immers volledig zelf verzinnen. Bij TLV ben je gebonden aan een vaste lengte van de tags, hierdoor ben je een stuk beperkter in het aantal tags dat je kunt gebruiken. Een ander voordeel van het niet gebonden zijn aan een vaste lengte van de tags is dat je beschrijvende namen kunt gebruiken zonder dat je deze moet gaan afkorten.
Een andere reden om voor XML te kiezen is dat ik zelf de berichten moet schrijven. Als je dit doet met een TLV-structuur, moet je bij een wijziging in het bericht, het hele bericht weer doorlopen om de lengtes aan te passen. Bij XML hoeft dit dus niet en dat scheelt weer een beetje extra werk.
De tags die in mijn XML bericht (figuur 9) aanwezig zijn en de samenstelling hiervan wordt uitgewerkt in hoofdstuk 8.

# 6 Standaard software

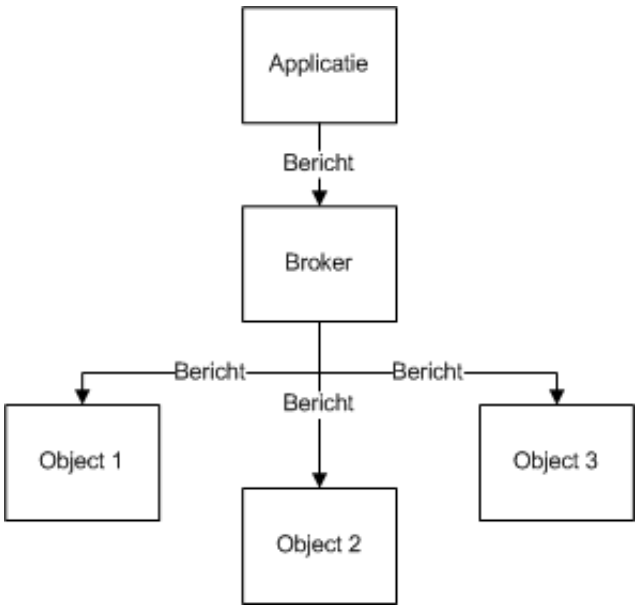
## 6.1 Standaard message brokers

Voor het vertalen van de berichten, vanuit de structuur die ik zelf ontworpen heb, zoals in figuur 9 in hoofdstuk 8, naar een LON of DLMS/COSEM bericht, is er natuurlijk een programma nodig. Nu zijn er echter programma's op de markt die je voor dit werk kunt configureren. Ik hoopte er dus op dat ik zelf geen programma hoefde te schrijven omdat ik hier dan een bestaand pakket voor kon kiezen. Een programma dat dit werk zou kunnen doen, heet een "message broker".
Bij het zoeken naar deze programma's, kijk ik alleen naar de gratis varianten. Het kunnen namelijk behoorlijk prijzige programma's zijn en voor het onderzoek dat ik doe, is immers geen budget beschikbaar. Een andere reden om te kiezen voor een gratis variant is dat andere mensen geen duur programma hoeven aan te schaffen om mijn prototype te kunnen gebruiken. Verder ben ik ook op zoek naar een programma dat onder windows draait aangezien dit vandaag de dag een van de meest gebruikte systemen is, en omdat ik zelf op dit moment geen andere testopstelling heb.
Hieronder volgt een lijstje met door mij gevonden gratis message brokers of message brokers die ik, als student informatica, zonder kosten kon krijgen:

- Apache ActiveMQ
- OpenAMQ
- Fuse Message Broker
- Microsoft BizTalk Server

Er zijn verschillende acties die message brokers kunnen uitvoeren, zo kunnen ze ervoor zorgen dat een bericht op de juiste plek wordt afgeleverd (figuur 3), en ze kunnen ervoor zorgen dat een bericht wordt omgezet naar het juiste formaat voor een bepaald object (figuur 4). Het is uiteraard ook mogelijk dat beide functionaliteiten worden gecombineerd in één message broker (figuur 5 en figuur 6). Er zijn twee mogelijkheden bij het maken van een gecombineerde message broker. Zo kun je ervoor kiezen om de berichten eerst te verdelen en daarna pas te vertalen zoals in figuur 5, of je kunt ervoor kiezen om de berichten eerst te vertalen en vervolgens pas te versturen naar de gewenste objecten zoals zichtbaar in figuur 6. Zoals zichtbaar is in de figuren heeft de volgorde van de verschillende functionaliteiten invloed op het ontwerp van de message broker. De message broker die ik wil gaan gebruiken, zal alleen de berichten kunnen omzetten naar het juiste formaat, en deze moet dus

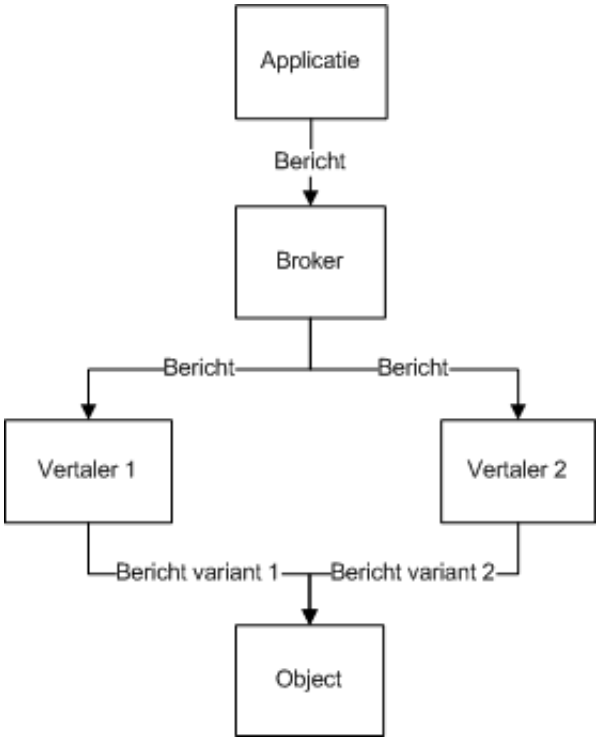
werken zoals zichtbaar is in figuur 4. Dit komt doordat het vertalen van de berichten de belangrijkste functionaliteit van mijn programma is. Daarnaast kan ik het vertalen en verdelen van de berichten niet op een echte meter testen aangezien ik de verbinding met een modem niet ga maken. Daarnaast is er slechts één COSEM meter binnen de universiteit beschikbaar. Het verdelen kan ik dus sowieso niet op een meter testen, dit zou echter nog wel mogelijk zijn door verschillende bestanden te creëren met verschillende adressen. Dit zou je dan, net zoals het vertalen nu, met de hand moeten controleren.
Bij het verdelen van de berichten zoals een message broker uit figuur 3 doet, wordt er een selectie gemaakt van de verschillende objecten die een bericht kunnen ontvangen. Het is dus niet zo dat een bericht altijd naar object 1, object 2 en object 3 wordt verzonden. Het kan dus ook zijn dat er alleen een bericht naar object 1 wordt gestuurd.




Figuur 3: Message broker die berichten verdeelt

### 6.1.1 Apache ActiveMQ

Apache ActiveMQ is een open source pakket. Dit programma ondersteunt verschillende veelgebruikte programmeertalen zoals Java, C, C++, PHP, Python. Verder ondersteunt het ook verschillende protocollen die worden gebruikt voor communicatie. Dit pakket wordt vooral gebruikt voor het verdelen van de berichten naar de juiste plekken en het omzetten naar de juiste




Figuur 4: Message broker die berichten vertaalt

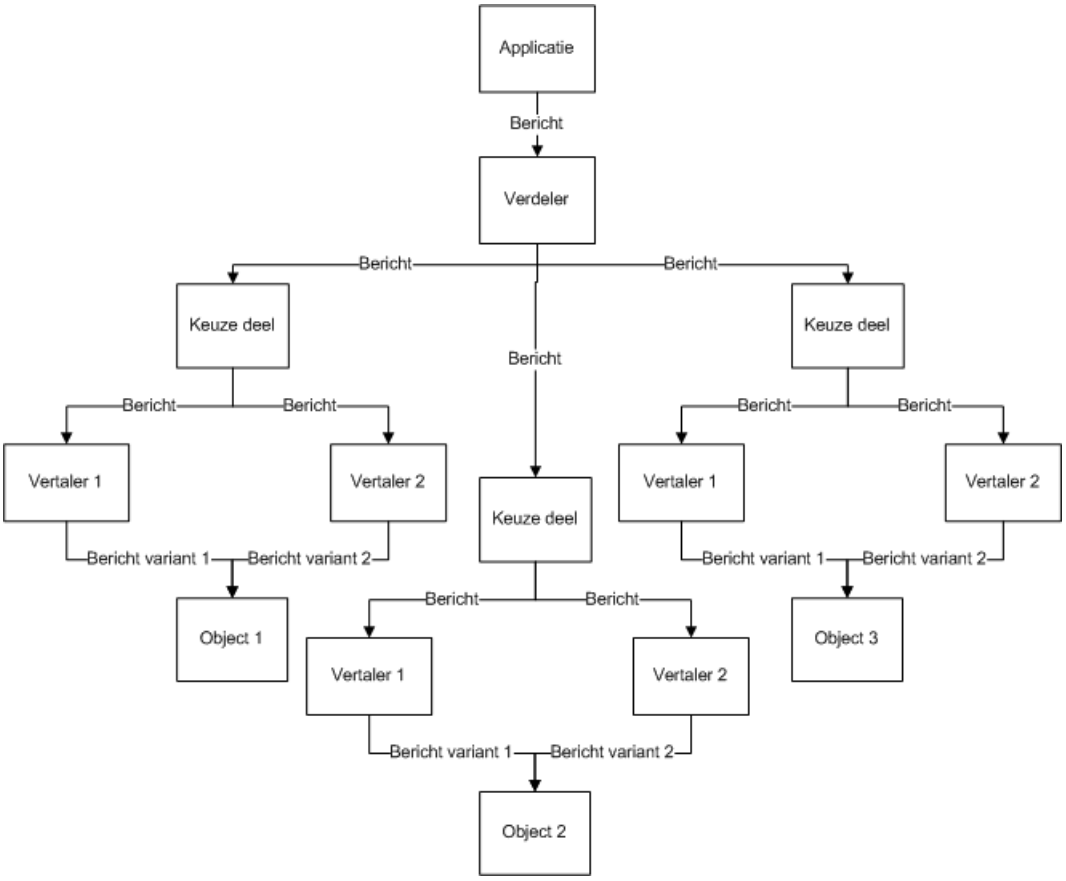
protocollen. In grote lijnen werkt dit pakket dus zoals weergegeven in figuur 3.

### 6.1.2 OpenAMQ

OpenAMQ is ook een gratis message broker. Dit programma maakt het mogelijk om verschillende applicaties met elkaar te laten communiceren via een Local Area Network (LAN). Het zorgt ervoor dat berichten op een juiste en zo snel mogelijke manier worden afgeleverd op de juiste plek. Deze message broker voldoet dus ook aan de functionaliteit voor het verdelen van verschillende berichten zoals weergegeven in figuur 3.

### 6.1.3 Fuse Message Broker

Fuse Message Broker is een pakket dat verder bouwt op Apache ActiveMQ. Zo ondersteunt dit pakket bijvoorbeeld meer standaarden zoals HTTP. Dit programma levert grote hoeveelheden data af op de juiste plaats. Een voordeel van dit programma is dat de mensen die de code hiervoor geschreven hebben, nog steeds ondersteuning geven. Verder zou dit pakket beter moeten

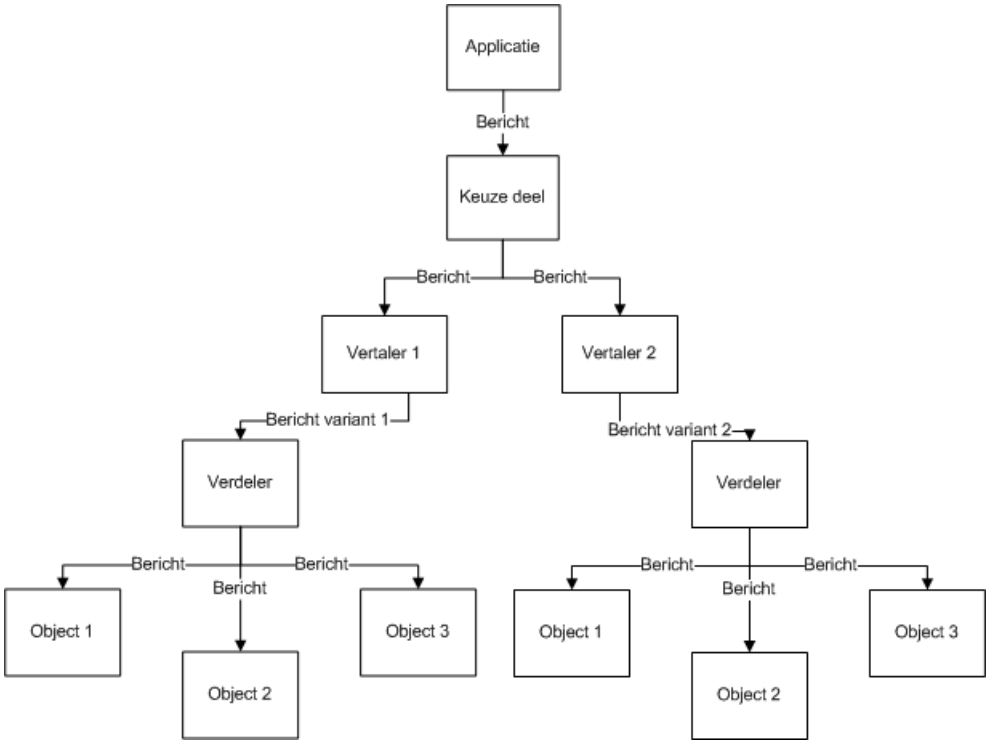



Figuur 5: Message broker die eerst verdeelt

presteren dan alle andere pakketten (waaronder ActiveMQ). Net als Apache ActiveMQ werkt dit pakket dus in grote lijnen zoals weergegeven in figuur 3.

### 6.1.4 Microsoft BizTalk Server

Microsoft BizTalk Server is een programma van Microsoft dat ook een message broker bevat. Dit is echter geen gratis programma maar ik heb het kunnen downloaden via mijn MSDN AA account die ik als student informatica via de universiteit heb gekregen. MSDN AA staat voor MicroSoft Developer Network Academic Alliance. Het is een project dat is opgezet om academische instellingen, medewerkers en studenten te voorzien van tools. Een academische instelling kan lid worden van MSDN AA en verkrijgt hiermee het recht om de software te verspreiden aan bijvoorbeeld studenten. MSDN AA is onderdeel van het grotere MSDN programma dat als doel heeft om ontwikkelaars te voorzien van de nieuwste tools en informatie om zo snelle




Figuur 6: Message broker die eerst vertaalt

ontwikkeling van innovatieve programma's te stimuleren. Het niet gratis zijn van dit programma maakt ook voor eventuele andere gebruikers niet zo veel uit. Mensen binnen de universiteit kunnen het programma immers via MSDN downloaden. Als bedrijven iets met het prototype willen doen is het voor hen, in tegenstelling tot particulieren, een relatief kleine investering om het programma eenmalig aan te schaffen.
Microsoft Biztalk Server zorgt ervoor dat berichten van verschillende programma's worden omgezet zodat deze programma's elkaar kunnen begrijpen. Een programma kan een bericht naar deze applicatie sturen en die zal vervolgens kijken naar welk type bericht het moet worden omgezet, naar welke andere programma's het moet worden gestuurd, en dit vervolgens ook uitvoeren. Neem bijvoorbeeld programma A dat een bericht naar programma B wil sturen. Echter programma A gebruikt protocol 1 en programma B gebruikt protocol 2. We nemen nu even aan dat Biztalk Server de functionaliteit voor het vertalen van protcol 1 naar protocol 2 bevat. Doordat de A en B niet rechtstreeks met elkaar kunnen communiceren, stuur programma A zijn bericht eerst naar Biztalk Server. Biztalk Server zal vervolgens het bericht omzetten naar een bericht dat binnen protocol 2 valt. Als dit gedaan

is, wordt dit bericht doorgestuurd naar programma B. Programma B kan nu het bericht dat A heeft gestuurd, begrijpen en verwerken.
Microsoft BizTalk Server 2009 kan zowel als figuur 3 en figuur 4 fungeren. Er is echter nog wel heel wat extra software nodig voordat dit pakket echt helemaal volledig kan werken.

## 6.2 Toepasbaarheid

Ondanks dat er message brokers bestaan, heb ik ervoor gekozen om zelf een applicatie te schrijven die mijn zelfgedefinieerde berichten omzet naar een LON of COSEM bericht. Ik heb hiervoor gekozen omdat OpenAMQ, Apache ActiveMQ en Fuse Message Broker alleen de functionaliteit uit figuur 3 bieden. Daarnaast kan Apache ActiveMQ het nooit winnen van Fuse Message Broker. Dit komt doordat Fuse is uitbreiding is op ActiveMQ en dus alleen maar meer en betere functionaliteit biedt. Het zou dus niet slim zijn om te kiezen voor Apache ActiveMQ als men ook voor Fuse Message Broker kan kiezen. Aangezien ik minimaal de functionaliteit voor het vertalen van berichten (figuur 4) nodig heb, zijn deze pakketten dus geen optie voor mij. Ondanks dat Microsoft BizTalk Server deze functionaliteit wel biedt, heb ik hier niet voor gekozen omdat het veel aanvullende software vereist, zoals Sharepoint Sever en SQL Server, die ik niet allemaal gratis kan verkrijgen. Een ander praktisch probleem is dat ik niet over een testopstelling beschik, die volledig aan de systeemeisen van dit pakket voldoet. Hierdoor zou ik dus niet weten of het aan het systeem ligt als iets niet werkt, of dat ik zelf iets fout heb gedaan. Verder vermoedde ik dat ik net zo veel tijd kwijt zou zijn met het configureren van deze message broker als met het zelf maken van een dergelijke applicatie. Al met al heb ik dus besloten om zelf een applicatie te gaan schrijven die voor mij een XML bericht kan omzetten naar een type bericht van het gewenste protocol.

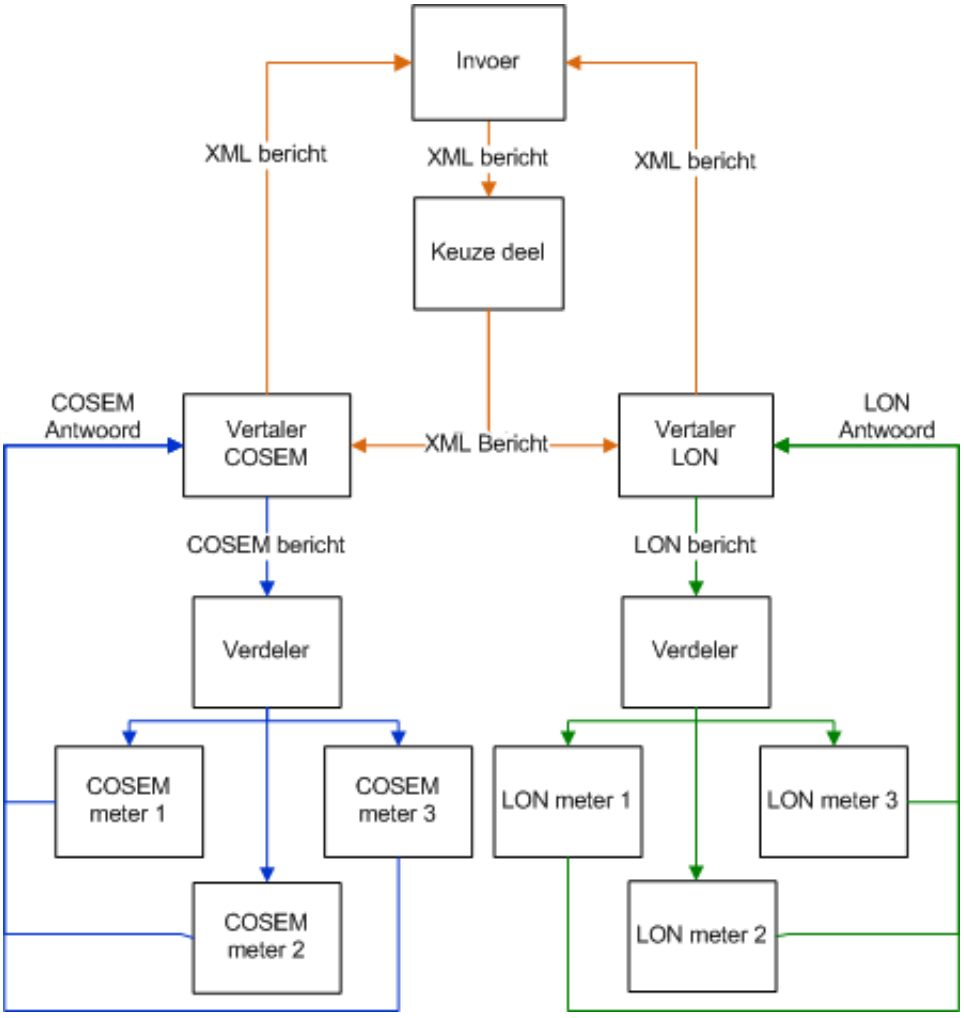
# 7 Eigen ontwerp

## 7.1 Ontwerp

Voordat er een applicatie gemaakt kan worden, moet er natuurlijk eerst een ontwerp gemaakt worden. In figuur 7 wordt het ontwerp van de applicatie getoond zoals dit er in de ideale situatie uitziet (Oranje representeert een XML bericht, blauw een COSEM bericht, groen een LON bericht). Echter zal het prototype dat ik ga maken, niet alle functionaliteiten bevatten zoals weergegeven in het plaatje. Het prototype bevat natuurlijk wel de belangrijkste functionaliteit die de message broker moet bevatten, namelijk het vertalen naar het juiste protocol (figuur 8). De functionaliteit voor het verdelen van berichten, zal dus net als de functionaliteit voor het vertalen van antwoorden niet geïmplementeerd worden. De applicatie begint met de invoer van een XML bestand en heeft vanuit dit punt twee mogelijkheden. Vanaf dit punt moet er namelijk gekozen worden of er een DLMS/COSEM bericht, of een LON bericht moet worden gemaakt. Als dit bekend is, zal het bericht vanaf het laagste niveau worden opgebouwd tot een compleet bericht van het gewenste protocol.
In mijn ontwerp heb ik ook het juist verdelen van de berichten meegenomen omdat dit in de uiteindelijke applicatie ook aan de orde is. Het is immers zo dat niet alle berichten naar alle meters moeten worden gestuurd. Zoals zichtbaar is in figuur 8, zal deze functionaliteit niet aanwezig zijn in het prototype dat ik ga maken.

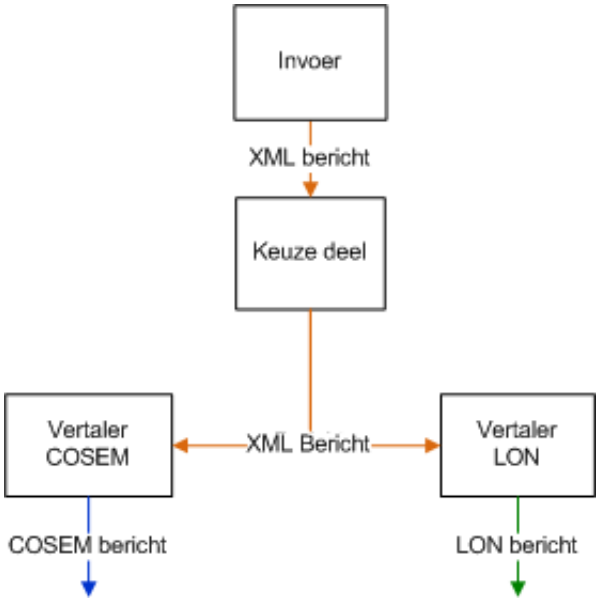
## 7.2 Verantwoording

In dit gedeelte zal ik het ontwerp van de applicatie in de ideale situatie (figuur 7) uitleggen. Zoals zichtbaar is in het plaatje, heb ik ervoor gekozen om het bericht eerst te vertalen en vervolgens pas te verspreiden. Dit heb ik gedaan omdat je dan voor ieder soort maar één vertaler nodig hebt. Zoals in figuur 3 zichtbaar is moet je voor ieder object bepalen naar welk type het XML bericht vertaald moet worden. Dit is in figuur 4 niet het geval. Een andere reden om voor deze volgorde te kiezen, is het feit dat er minder "overbodige" informatie moet worden doorgestuurd. Dit komt doordat het bericht meteen vertaald wordt naar het juiste type. Bij de volgorde zoals in figuur 3, moet informatie voor een specifiek protocol, na het verdelen nog eens naar iedere vertaler worden doorgestuurd. Bij een applicatie zoals in figuur 4 heb je na het vertalen alleen nog de informatie voor dat specifieke protocol.
Nu zal ik uitleggen hoe de applicatie in grote lijnen werkt. De applicatie start




Figuur 7: Ontwerp message broker (ideale situatie)

met de invoer van een XML bestand zoals ik later zal tonen en uitleggen in hoofdstuk 8. Vervolgens zal er worden gekeken naar welk type bericht dit moet worden omgezet. In dit geval zijn er twee mogelijkheden: een COSEM bericht of een LON bericht. Het is natuurlijk mogelijk dat hier in de toekomst nog een ander type aan wordt toegevoegd. Je krijgt dan een extra tak voor elk nieuw type dat men toevoegt. Het keuzedeel is in wezen dus niet meer dan een switch die bepaalt in welke vervolgtak je terecht komt.
Als er voor een type bericht is gekozen zal de vertaler het bericht omzetten naar het gewenste type. Binnen deze vertaler zal er ook weer een keuze gemaakt worden op het gebied van het soort bericht. Er zijn immers verschillende berichten (bijvoorbeeld “geef meterstand” of “stop afname”). Als




Figuur 8: Ontwerp prototype

er dan uiteindelijk een specifiek bericht van het juiste type is, zal dit bericht naar de juiste meters gestuurd moeten worden. Hievoor zorgt de verdeler, deze kijkt naar welke meters het bericht gestuurd moet worden, en zal hier vervolgens ook voor zorgen.
Nadat het bericht is afgeleverd bij een meter, zal de meter in veel gevallen een antwoord terug sturen. Ook dit antwoord is voor ieder protocol anders zoals zichtbaar is in figuur 7. Ook hier is het dus handig als je uiteindelijk van iedere meter een antwoord in hetzelfde formaat terugkrijgt. Om dit voor elkaar te krijgen zal de meter het antwoord weer terugsturen naar de vertaler. Deze zal dit bericht dan weer omzetten naar een vooraf gespecificeerd formaat zodat een (andere) applicatie hier weer gegevens uit kan halen.

# 8 Implementatie

## 8.1 Berichtindeling

Voor het versturen van de berichten moet ik zelf mijn eigen XML structuur indelen. Hiervoor heb ik gekeken naar de indeling van LON en COSEM berichten. Ik heb geprobeerd om de gemeenschappelijk dingen uit deze indeling te halen en deze in de XML structuur te stoppen. Dit heeft als resultaat de XML indeling zoals zichtbaar is in figuur 9. Als je naar dit bericht kijkt valt

```
<MESSAGE>
  <ADDRESS>
    <SOURCE>Source_address</SOURCE>
    <DESTINATION>meter_id</DESTINATION>
  </ADDRESS>
  <TYPE>message_type</TYPE>
  <DATA>value</DATA>
          .
          . 0-n
          .
  <DATA>value</DATA>
</MESSAGE>
```

Figuur 9: Eigen berichtindeling

meteen op dat het erg kort is en er heel simpel uit ziet. Dit komt doordat je op het hoogste niveau alleen het type bericht nodig hebt en de bestemming waar het bericht naar toe moet, maar hierover later meer in het volgende deelhoofdstuk. Als je bijvoorbeeld de meterstand wil opvragen hoef je alleen een bericht te versturen met daarin “geef meterstand” als informatie. De rest van de informatie kun je dan met de applicatie eromheen plaatsen.
Als je op een dieper niveau zou kijken zal dit echter wat anders zijn, je kunt dan extra informatie als attributen toevoegen. Bij het omzetten van deze berichten naar een LON of COSEM bericht, zullen niet altijd alle attributen gebruikt worden omdat ze niet voor beide typen berichten nodig zijn. Op het hoogste niveau wordt zo min mogelijk informatie door middel van optionele attributen meegegeven. Als informatie voor slechts één protocol nodig is, zal alleen informatie die achteraf echt niet kan worden toegevoegd, als optioneel attribuut worden toegevoegd, dit heb ik echter niet nodig gehad bij het maken van mijn prototype. Informatie die voor beide protocollen nodig is, kan altijd als attribuut worden toegevoegd aangezien dit nooit overbodig is. Ech-

ter, als deze informatie heel makkelijk achteraf kan worden toegevoegd, zal dit ook gebeuren. Dit maakt het opstellen van de XML berichten namelijk weer wat makkelijker, aangezien dit met de hand moet gebeuren.
Verder is het ook zo dat je niet alle componenten die in een COSEM bericht zitten, ook in dit bericht moet stoppen. Zo heb je bijvoorbeeld geen flag nodig omdat het begin en einde expliciet aangegeven wordt. Uiteraard zullen tijdens de vertaling deze flags wel worden toegevoegd, maar hier heb je geen informatie voor nodig die je al met het XML bericht zou moeten meesturen. Verder worden de HCS en de FCS berekend aan de hand van de informatie in het bericht, die hoef je dus ook niet in het bericht te stoppen aangezien je die pas bij de vertaling kunt berekenen (het is ook niet mogelijk om dit er in te stoppen). Hetzelfde geldt voor het CRC veld uit een LON bericht. Ook dit veld dient on-the-fly berekend te worden. Verder worden de prioriteit en het te gebruiken kanaal in de code gestopt. De versie van het LON protocol zal ergens hard-coded in de code van de vertaler komen te staan omdat deze toch niet vaak zal veranderen.

### 8.1.1 Verantwoording berichtindeling

Een XML bericht dat door het systeem wordt ingelezen moet altijd beginnen met <MESSAGE> en eindigen met </MESSAGE> Op deze manier weet het programma dat het klaar is. Vervolgens komt er binnen de message-tags een adres. Er kan maar één adresdeel in het bericht zitten. Dit veld moet altijd beginnen met <ADDRESS> en eindigen met </ADDRESS>. Binnen deze tags komt het adres van de bron en het adres van de bestemming. Het deel voor het bronadres moet altijd beginnen met <SOURCE> en eindigen met </SOURCE>. Binnen deze twee tags komt dan het adres van degene die het bericht stuurt of vanuit wie er getest moet worden. Als het sourcedeel is afgesloten komt er een deel voor de bestemming van het bericht. Dit deel moet beginnen met <DESTINATION> en eindigen met </DESTINATION>. Tussen deze twee tags komt het id van de meter waar het bericht naartoe gestuurd moet worden. De waarde tussen deze twee tags is dus niet het daadwerkelijke adres van de meter. Het is een id van een meter. Aan de hand van dit id kunnen de juiste gegevens (protocoltype en adres) uit de database gehaald worden. Nadat het destinationveld is gesloten, wordt ook het adresveld gesloten. Als dit gebeurd is komt er een regel die het type bericht bevat. Dit deel moet altijd beginnen met <TYPE> en eindigen met </TYPE>. Tussen deze twee tags komt een getal dat aangeeft welk type bericht er verstuurd moet worden. De verschillende mogelijkheden staan in tabel 5. Nu komt er een optioneel veld. Dit veld begint met <DATA> en eindigt met </DATA>. Tussen deze tags kan een waarde worden geplaatst

| | |
|---|---|
| 1 | Opvragen van meterstand |
| 2 | Zet het gemiddelde verbruik in een periode op een waarde |

Tabel 5: Geïmplementeerde berichten

die bijvoorbeeld bij een Set bericht nodig is. Als je een veld namelijk een waarde wilt geven. Moet je deze waarde ook kunnen meegeven. Als dit deel is afgesloten wordt het hele bericht afgesloten.
Zoals zichtbaar is zitten er niet veel tags in mijn bericht. Alle andere informatie die ik nodig heb zit in de applicatie, of kan real-time gecreëerd worden. Bij beide berichttypes heb ik ervoor gekozen om de prioriteit van het bericht hard-coded in de applicatie te stoppen omdat deze in mijn ogen niet van belang is. De enige informatie die ik wel nodig heb is uiteraard de meter waar het bericht heen gestuurd moet worden, en wat bericht het is.
Het vlagveld in een COSEM bericht heeft een vaste waarde en kan dus in de applicatie gestopt worden. Omdat het frame formatveld ook een gedeeltelijk vaste waarde heeft kan dit ook in het programma gemaakt worden. Vooral het laatste deel van het veld dat de lengte van het bericht aangeeft kan ook pas echt in het programma berekend worden. De bestemming moet uiteraard wel meegegeven worden. De controlbyte wordt berekend aan de hand van de waarde die tussen de type-tags zit en wordt dus in zekere zin wel meegegeven. Zoals al eerder gezegd, kan de waarde voor de HCS en FCS niet worden meegegeven omdat deze simpelweg nog niet bekend is. Tenslotte kan het informatieveld, net als de control byte, berekend worden aan de hand van het type bericht. Aan de hand hiervan kunnen alle bytes van het veld de juiste waarde krijgen.
Ook bij het LON protocol heb ik niet meer informatie nodig. Binnen de L2Hdr staat het prioriteitsveld. De waarde die dit veld krijgt staat hard-coded in de applicatie omdat het niet helemaal duidelijk is wat het betekent als dit veld de waarde 1 heeft gekregen. Het AltPath-veld gebruik ik eigenlijk niet omdat de berichten gewoon altijd over één kanaal verstuurd worden. Dit is voldoende voor het testen van de securityaspecten. Het resterende veld moet real-time berekend worden omdat de waarde hiervan vooraf niet bekend is. Dan komen we bij de informatie voor de NPDU. Ik heb ervoor gekozen om ook de versie van het protocol in de code te stoppen omdat deze niet vaak zal veranderen. Het heeft in mijn ogen dus weinig zin om deze iedere keer mee te geven. Het PDU Format wordt gecreëerd aan de hand van het berichttype. Hetzelfde geldt voor het Address Format. Het adresveld wordt vervolgens gemaakt aan de hand van het Address Format. Echter, de informatie voor de bestemming kan dan weer uit het XML-bericht worden gehaald. Het type dat in het Enclosed PDU-veld zit, wordt ook weer bepaald

aan de hand van het berichttype. De CRC die als laatste berekend wordt, kan net als de HCS en FCS in een COSEM bericht, niet anders dan real-time berekend worden.

### 8.1.2 Voorbeeld XML-bericht

Hieronder wordt een voorbeeld getoond van een XML dat het mijn applicatie kan inlezen.

```
<MESSAGE>
  <ADDRESS>
    <SOURCE>33</SOURCE>
    <DESTINATION>1</DESTINATION>
  </ADDRESS>
  <TYPE>1</TYPE>
</MESSAGE>
```

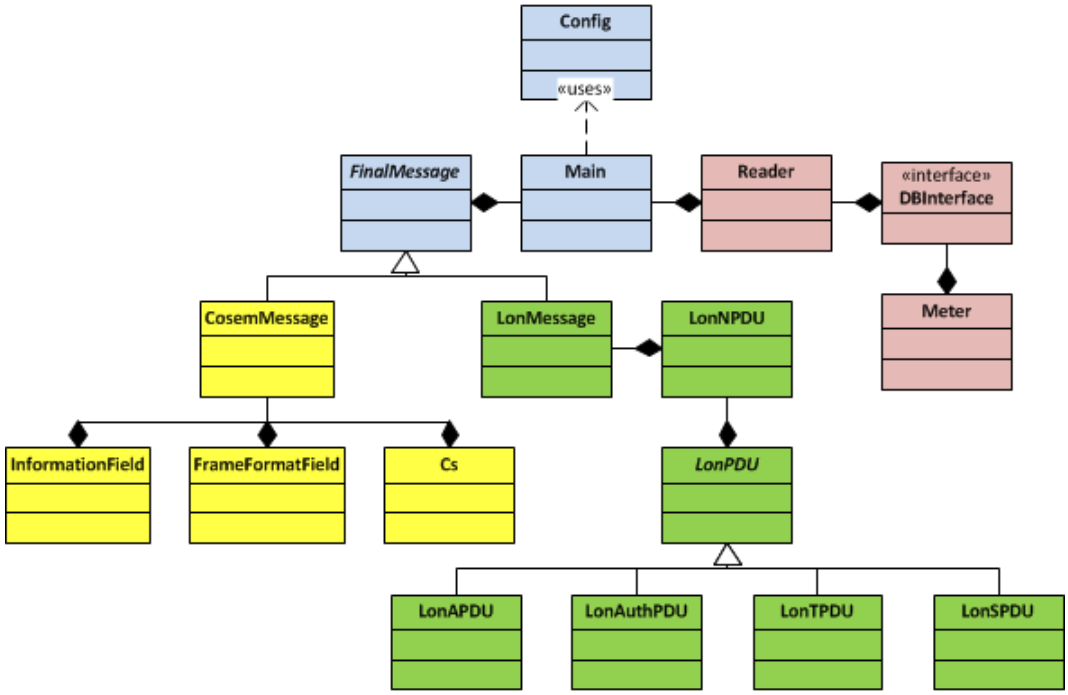
Zoals je kunt zien wordt alleen de bestemming en het type meegegeven. Aan de hand van het meter_id in het destination veld kan het programma de gegevens van de betreffende meter uit de database inlezen. Hierover later meer. Het type geeft aan om wat voor soort bericht het gaat. In mijn applicatie heb ik een lijstje opgenomen met de type berichten. Als je hierin gaat zoeken zie je dat het om een meterreading gaat.

## 8.2 Opbouw

Om de onderlinge samenhang van de verschillende klassen in mijn applicatie te laten zien heb ik hier een klasse-diagram van gemaakt in UML notatie [23, 21]. Dit is te zien in figuur 10.

In figuur 10 is de structuur van de COSEM en LON berichten te zien zoals deze is getoond in tabel 1, 2, 3 en 4. Zo bevat een CosemMessage een InformationField, een FrameFormatField en een CS (kan zowel HCS als FCS zijn). De overige onderdelen van een COSEM bericht zitten als attribuut in de klasse CosemMessage. Hetzelfde geldt voor een LonMessage, deze bevat dan weer een LonNPDU die op zijn beurt weer een LonPDU bevat. Deze LonPDU klasse is dan weer de superklasse van de LonAPDU, LonAuthPDU, LonTPDU en LonSPDU. Ook hier geldt weer dat de overige onderdelen van een LON bericht als attribuut in de LonMessage klasse.

Om ervoor te zorgen dat mijn applicatie gemakkelijk kan worden uitgebreid, heb ik geprobeerd om mijn programma in modules op te bouwen. Dit is overigens altijd verstandig om te doen. Door de applicatie modulair te bouwen is het vrij eenvoudig om hier functionaliteit aan toe te voegen. Zo kan




Figuur 10: Klasse diagram

eenvoudig een protocol (anders dan cosem of lon) worden toegevoegd. Hiervoor moet men een nieuwe subklasse maken van FinalMessage. Doordat het een subklasse is van FinalMessage kan op ieder bericht dezelfde actie worden uitgevoerd. Vervolgens moet er nog een nieuwe case worden toegevoegd in onderstaande functie die te vinden is in Main.java.

```
private static void createFinalMessage()
{
  switch(Config.MESSAGETYPE)
  {
      case Config.COSEM:
              finalMessage = new CosemMessage(...);
              break;
      case Config.LON:
              finalMessage = new LonMessage(...);
              break;
      default: //show error message
  }
}
```

Verder heb ik de applicatie opgedeeld in vier delen zoals zichtbaar is in figuur 10. Zo heb ik een deel voor COSEM berichten (het gele deel), een deel voor LON berichten (het groene deel), een deel voor dat de XML-berichten interpreteert, genaamd reader (het rode deel), en een deel waarin alles aan elkaar gekoppeld wordt: de broker (het blauwe deel).

### 8.2.1 Cosem

Ieder protocol heeft een eigen deel binnen mijn programma. Een COSEM bericht begint bij de klasse CosemMessage, dit is een subklasse van de FinalMessage klasse. Binnen het COSEM bericht moeten er ook keuzes gemaakt worden om een specifiek bericht te maken. Een GetMeterReading ziet er bijvoorbeeld anders uit dan een Set bericht. Daarom zit er bij het maken van de controlbyte een keuzemogelijkheid:

```
private void createControl()
{
    switch(Integer.parseInt(Config.TYPE))
    {
        case Config.GET_METERREADING:
            control = (byte) (((0 | (1 << 5))) | (1<<4));
            break;
        case Config.SET_AVG_BILLING_PERIOD:
            control = (byte) (((0 | (1 << 5))) | (1<<4));
            break;
        default:
            //show error message
            System.exit(1); break;
    }

    frameLength += 1;
}
```

Als men een ander bericht wil versturen dan de GetMeterReading of het Set bericht, moet men een extra case toevoegen in bovenstaande functie. Op deze manier kan de controlbyte worden aangepast aan het daadwerkelijke bericht. In het geval van een meterreading zet deze functie het vijfde en zesde bit van de control byte op 1. Als je in sectie 8.3.4.3.2 van het Green Book [17] kijkt, zie je dat dit klopt. Het is namelijk een I-frame en het eerste bericht (vandaar dat het zesde bitje op 1 wordt gezet, en tevens het laatste deel van een bericht (vandaar dat het vijfde bitje op 1 wordt gezet). Bij een ander bericht kan het dus zijn dat de controlbyte er anders uit komt te zien,

vandaar dat er een keuze gemaakt moet worden.
De echte boodschap van een bericht zit in het informatieveld. Omdat dit een uitgebreid veld is, heb ik hier een aparte klasse voor gemaakt waarvan er uiteindelijk een instantie in een CosemMessage terecht komt. Binnen het informatieveld wordt diverse keren een keuze gemaakt om zo tot het juiste bericht te komen. In de constructor van het veld wordt de keuze gemaakt uit één van de volgende berichten:

- Get.Request
- Get.Confirm
- Set.Request
- Set.Confirm
- Action.Request
- Action.Confirm

Omdat ik slechts een prototype maak en dus niet de tijd heb om alle mogelijke berichten in te bouwen, beperk ik mij slechts tot een klein gedeelte van de Get.Request en de Set.Request. Binnen de Get.Request zijn weer andere keuzemogelijkheden, deze kunnen in sectie 9.4 van het Green Book [17] worden gevonden. Hoe een Get.Request eruit ziet staat op pagina 220 van het boek. Zoals je kunt zien (er staat ::=Choice) is het een keuze tussen een Get.Request.Normal, Get.Request.Next en een Get.Request.With.List. In dit geval kies ik voor de normal variant (in de applicatie wordt deze keuze gemaakt aan de hand van het soort bericht dat verstuurd wordt). Hoe deze variant eruit ziet kun je ook op pagina 220 vinden. Aangezien dit een Sequence is, hoeft hier geen keuze gemaakt te worden. Het laatste veld (access-selection) is optioneel. Doordat ik niet precies weet wat het access-selection veld doet, heb ik besloten om het in mijn prototype weg te laten, het immers slotte optioneel. Verder kun je weer op dezelfde wijze als de Get.Request.Normal opzoeken hoe de eerste twee velden eruit zien.
Voor een volledige applicatie zou men voor ieder blokje uit sectie 9.4 van het Green Book een aparte functie moeten maken (inclusief optionele velden). Door deze functies vervolgens te combineren kun je alle mogelijke berichten versturen. Echter, zoals gezegd, beperk ik mij tot enkele berichten en zal ik dus ook slechts enkele van deze blokjes implementeren.

### 8.2.2 Lon

Net zoals het COSEM deel heeft ook het LON deel een eigen package binnen mijn applicatie. Op het hoogste niveau bestaat het LON bericht uit de L2Hdr, een NPDU en een CRC. Omdat de L2Hdr ook weer opgesplitst is in drie verschillende velden heb ik hiervoor een functie gemaakt die weer drie andere functies, voor het maken van de verschillende velden, aanroept. Helaas was voor het LON protocol niet voldoende informatie beschikbaar om er een echt bericht mee te kunnen maken. Wat ik wel heb kunnen doen is de opbouw van het bericht in mijn code stoppen. Net als bij een COSEM bericht moeten er voor een LON ook weer diverse keuze gemaakt worden binnen de applicatie. Als er dan duidelijk is hoe bijvoorbeeld een meterreading eruit ziet, hoeft men alleen nog de juiste functies aan te roepen en nog enkele cases toe te voegen. Zo staat er in de code (LonAPDU.java) bijvoorbeeld het volgende:

```
private void createData()
{
    switch(Integer.parseInt(Config.TYPE))
    {
        //add case for different types of messages
        default: notImplemented(); break;
    }
}
```

Op de plek waar het commentaar staat moet dan een case worden toegevoegd die voor het type bericht de exacte data toevoegt. Zoals gezegd kan ik dat helaas niet maken met de informatie die ik heb.

### 8.2.3 Reader

In het reader package worden alle dingen ingelezen. Dit houdt in dat het XML bericht dat vertaald moet worden, wordt ingelezen en dat de meters uit de database worden ingelezen. Aan de hand van het id dat in het adresveld van het XML bericht staat wordt dan de juiste meter gekozen. In de database staat één tabel, genaamd meters, met daarin van iedere meter die men wil testen:

- Id
- Protocoltype
- Adres

Op deze manier kan aan de hand van het id dus ook het juiste protocol worden gekozen en het adres worden toegevoegd. Verder is het id van een meter gekoppeld aan het destinationveld uit een XML-bericht. De waarde in het destination veld moet overeen komen met het id van een meter in de database, anders kan er niet vertaald worden omdat er dan ook onbekend is naar welk type er vertaald moet worden.
Als database heb ik gekozen voor een SQLite database, dit omdat deze zeer gemakkelijk verplaatst kan worden omdat alles in één bestand wordt opgeslagen. Verder had ik deze database variant al eens eerder gebruikt waardoor ik een groot deel kon hergebruiken en dat bespaarde mij weer wat tijd.

## 8.3 Uitvoer

### 8.3.1 Cosem

Als ik het bericht uit sectie 8.1.2 (standaard bericht voor opvragen meterstand) invoer krijg ik de volgende hexadecimale bytearray eruit:

```
7E A0 16 20 23 21 30 6F 0E C0 01 00 00 03 01 00 01 08 00 FF 02 C0 1C 7E
```

Als ik deze bytes ga analyseren kom ik tot de volgende verdeling:

```
Flag: 7E
Frame Format: A0 16
Destination: 20 23
Source: 21
Control: 30
Hcs: 6F 0E
Information: C0 01 00 00 03 01 00 01 08 00 FF 02
  GetRequest: C0
    GetRequestNormal: 01
      InvokeAndPriorityId: 00
      AttributeDescriptor: 00 03 01 00 01 08 00 FF 02
        ClassId: 00 03
        InstanceId: 01 00 01 08 00 FF
        AttributeId: 02
Fcs : C0 1C
Flag: 7E
```

Zoals hierboven zichtbaar is, komt er een kloppend bericht uit mijn applicatie. Dit kan gecontroleerd worden aan de hand van de informatie in het Green Book en het Blue Book [17, 16].
Als ik het onderstaande bericht in mijn prototype stop, is duidelijk te zien dat

er binnen het informatieveld waardes veranderen. Buiten het informatieveld blijven vrij veel waardes hetzelfde doordat er uiteraard ook informatie, zoals source en destination, hetzelfde blijft.

```
<MESSAGE>
  <ADDRESS>
    <SOURCE>33</SOURCE>
    <DESTINATION>1</DESTINATION>
  </ADDRESS>
  <TYPE>2</TYPE>
  <DATA>6</DATA>
</MESSAGE>
```

Nadat dit bericht is ingelezen komt er weer een hexadecimale bytearray uit mijn programma die er dit keer als volgt uitziet:

```
7E A0 17 20 23 21 30 64 4A C1 01 00 00 03 01 00 01 00 00 FF 02 06 06 1A 7E
```

Als ik deze dan ga analyseren kom ik tot de volgende indeling:

```
Flag: 7E
Frame Format: A0 17
Destination: 20 23
Source: 21
Control: 30
Hcs: 64 4A
Information: C1 01 00 00 03 01 00 01 00 00 FF 02 06
  SetRequest: C1
    SetRequestNormal: 01
      InvokeAndPriorityId: 00
      AttributeDescriptor: 00 03 01 00 01 00 00 FF 02 06
        ClassId: 00 03
        InstanceId: 01 00 01 00 00 FF
        AttributeId: 02
        Value: 06
Fcs : 06 1A
Flag: 7E
```

Zoals zichtbaar is, verandert het Frame Format field. Dit komt doordat de lengte van het bericht één byte groter is geworden. Hierdoor verandert ook de HCS (en FCS) volledig. Het informatieveld begint nu met C1 in plaats van C0. Dit komt doordat er een Set_Request gestuurd wordt in plaats van een Get_Request. Verder is het meest opvallende aan deze uitvoer dat er

ook een Value is bijgekomen. Dit is logisch aangezien je bij een set operatie natuurlijk ook een waarde nodig hebt die je gaat zetten. Hier wordt het gemiddelde verbruik over een bepaalde periode op 6 gezet.

### 8.3.2 Lon

Voor een LON bericht heb ik helaas alleen een beperkte uitvoer van het programma. Dit komt doordat ik nergens heb kunnen vinden hoe bijvoorbeeld een meterreading er daadwerkelijk uit ziet. Ik heb alleen de informatie over de opbouw van LON berichten kunnen vinden. Echter, informatie over specifieke waardes van de verschillende bytes heb ik niet kunnen vinden. Hierdoor kan ik slechts een paar velden geheel invullen.

```
<MESSAGE>
  <ADDRESS>
    <SOURCE>33</SOURCE>
    <DESTINATION>2</DESTINATION>
  </ADDRESS>
  <TYPE>1</TYPE>
</MESSAGE>
```

Als ik bovenstaand bericht als invoer voor mijn programma gebruik, krijg ik de volgende bytearray als uitvoer:

```
00 B8 00 A1 00 FF 00 F2 6A
```

Als ik deze uitvoer dan analyseer kom ik tot de volgende indeling:

```
L2Hdr: 00
NPDU: B8 00 A1 00 FF
  FirstByte: B8
  Address: 00 A1 00 FF
  Domain: 01
  PDU: ?
CRC: A5 5C
```

Zoals zichtbaar is, ontbreekt er informatie. De belangrijkste informatie die ontbreekt is de enclosed PDU, hier zit namelijk de daadwerkelijke informatie in. Wat betreft het domain en het source- en destinationsubnet is ook meer informatie nodig (dit zijn de twee 00 velden binnen address). Ik heb in het voorbeeld gekozen voor slechts één domein omdat je hier voor het testen van de securityaspecten in principe voldoende aan hebt, hetzelfde zou je kunnen doen voor het source- en destinationsubnet. Als je namelijk maar één meter test, heb je ook maar één netwerk en subnetwerk nodig.

# 9 Conclusie

In dit verslag heb ik beschreven hoe ik tot het maken van de applicatie ben gekomen. Het maken van de applicatie ging echter niet zonder slag of stoot. Doordat er niet veel publiekelijke informatie over het LON protocol te vinden is, kon ik hier geen juiste implementatie van krijgen.
De applicatie die ik gemaakt heb is slechts een prototype. Ik heb geen volledige applicatie gemaakt omdat dit simpelweg te veel tijd zou kosten. Ik heb wel geprobeerd om de applicatie zo te ontwerpen zodat deze eenvoudig kan worden uitgebreid met de resterende dingen. De functionaliteit voor het verdelen van berichten kan worden ingebouwd bij het maken van de adresvelden. Hier moet dan gekeken worden waar het bericht naar toe moet en daar moet het adresveld op worden aangepast. Wat betreft het vertalen van de antwoorden naar een universeel antwoord, hiervoor moet simpelweg een heel nieuw deel aan het programma worden toegevoegd omdat hier binnen het prototype nog niets voor bestaat. Dit deel krijgt dan bijvoorbeeld een bytearray en gaat deze ontleden in alle losse elementen. Als alle losse elementen bekend zijn, kunnen deze in een universeel formaat worden gegoten en in bijvoorbeeld een XML-bestand worden opgeslagen. Ook hier geldt weer dat het voor COSEM geen probleem is maar voor LON wel, vanwege de ontbrekende informatie.
Ik verwacht dat het uitbreiden van de applicatie geen dramatische gevolgen heeft voor de prestatie aangezien het allemaal vrij eenvoudige berekeningen zijn. Het enige probleem met het opschalen zou de database kunnen zijn. Ik weet niet in hoeverre de prestaties verminderen als je hier heel erg veel meters in zet aangezien er gekeken moet worden welke meter je daadwerkelijk wil hebben. Echter, ik zie dit niet echt als een probleem aangezien het een applicatie is voor het testen van smart meters gaat. Het is in mijn ogen dan niet noodzakelijk om miljoenen meters in de database te stoppen.
De uiteindelijke keuze om zelf een applicatie te gaan maken in plaats van bestaande software te gebruiken was in mijn ogen geen slechte keuze. Ik ben hier dan wel flink wat tijd mee bezig geweest, maar de meeste tijd ging zitten in het uitvinden van de COSEM en LON structuur (hoe ziet een bericht er nou precies uit) en dit had ik ook moeten doen als ik een bestaand software pakket had moeten configureren. Een enorme tegenvaller hierbij is dat ik niet heb kunnen vinden hoe een specifiek bericht er in het LON protocol uitziet. Hierdoor heb ik voor een LON bericht dus geen uitvoer kunnen creëeren. Ik heb dit niet kunnen uitzoeken doordat er niet voldoende informatie over beschikbaar is, althans, deze is niet publiekelijk beschikbaar waardoor ik er niet bij kan. Ook het feit dat ik soms moest uitzoeken hoe ik iets in java moest programmeren, zoals specifieke bitoperaties, zie ik niet als een enorme

tegenvaller. Als ik namelijk een bestaand pakket had gebruikt had ik ook bij dit pakket moeten uitzoeken hoe het in elkaar zit en hoe ik het dan zou moeten configureren.
Het resultaat van mijn onderzoek is dus dat het niet mogelijk is om een applicatie te schrijven voor het testen van smart meters, zonder dat men na hoeft te denken over het te gebruiken protocol. Als er meer informatie beschikbaar komt voor het LON protocol zou dit wel mogelijk zijn. Tot die tijd kan men deze meters dus niet snel en automatisch testen, en zal men dit dus met de hand moeten blijven doen.

# 10 Verder onderzoek en verbeteringen

Voor een bruikbare applicatie moet er uiteraard ook een koppeling zijn met de meters. Dit gebeurt door het bericht naar een modem te sturen waarop de betreffende meters zijn aangesloten. Net zoals er verschillende smart meters zijn, zijn er ook verschillende soorten modems. Uiteraard zijn er voor de verschillende communicatietechnieken, zoals beschreven in hoofdstuk 4, verschillende modems beschikbaar. Het verschil houdt echter niet op bij het verschil in communicatietechnieken. Binnen een specifieke communicatietechniek, zoals PLC (hoofdstuk 4), kunnen ook nog verschillende soorten modems gebruikt worden. Zo wordt er bijvoorbeeld gebruik gemaakt van verschillende modulaties.
Om het prototype bruikbaar te maken zullen ook nog de ontbrekende berichten aan de applicatie moeten worden toegevoegd. Als dit gedaan is en de koppeling met het modem is er, zou er met iedere smart meter die het COSEM of LON protocol gebruikt, gecommuniceerd moeten kunnen worden. Om de applicatie nog bruikbaarder te maken kan men ook nog een vertaler voor de antwoorden er tegenaan maken. Dit is op zich een losstaand deel van de applicatie omdat hier nog helemaal niets van geïmplementeerd is. Er kan ook geen van de voor deze applicatie geschreven code gebruikt worden. Er moet namelijk een deelapplicatie gemaakt worden die precies het omgekeerde doet. Voor een COSEM bericht staat in het Green Book precies beschreven hoe het bericht er uit ziet en wat een bepaalde waarde van een bepaalde byte precies betekent. Zoals gezegd, is dit bij LON niet het geval. Dit zal ook hier dus weer een probleem opleveren.
Als blijkt dat er nog meer protocollen gebruikt worden voor communicatie met de meters, of er komt een nieuw protocol bij, zal dit in de uiteindelijke applicatie moeten worden toegevoegd. Anders heeft de applicatie namelijk geen zin omdat men dan nog steeds moet kijken welk protocol gebruikt wordt en dit is nou precies wat er met de applicatie voorkomen moet worden.

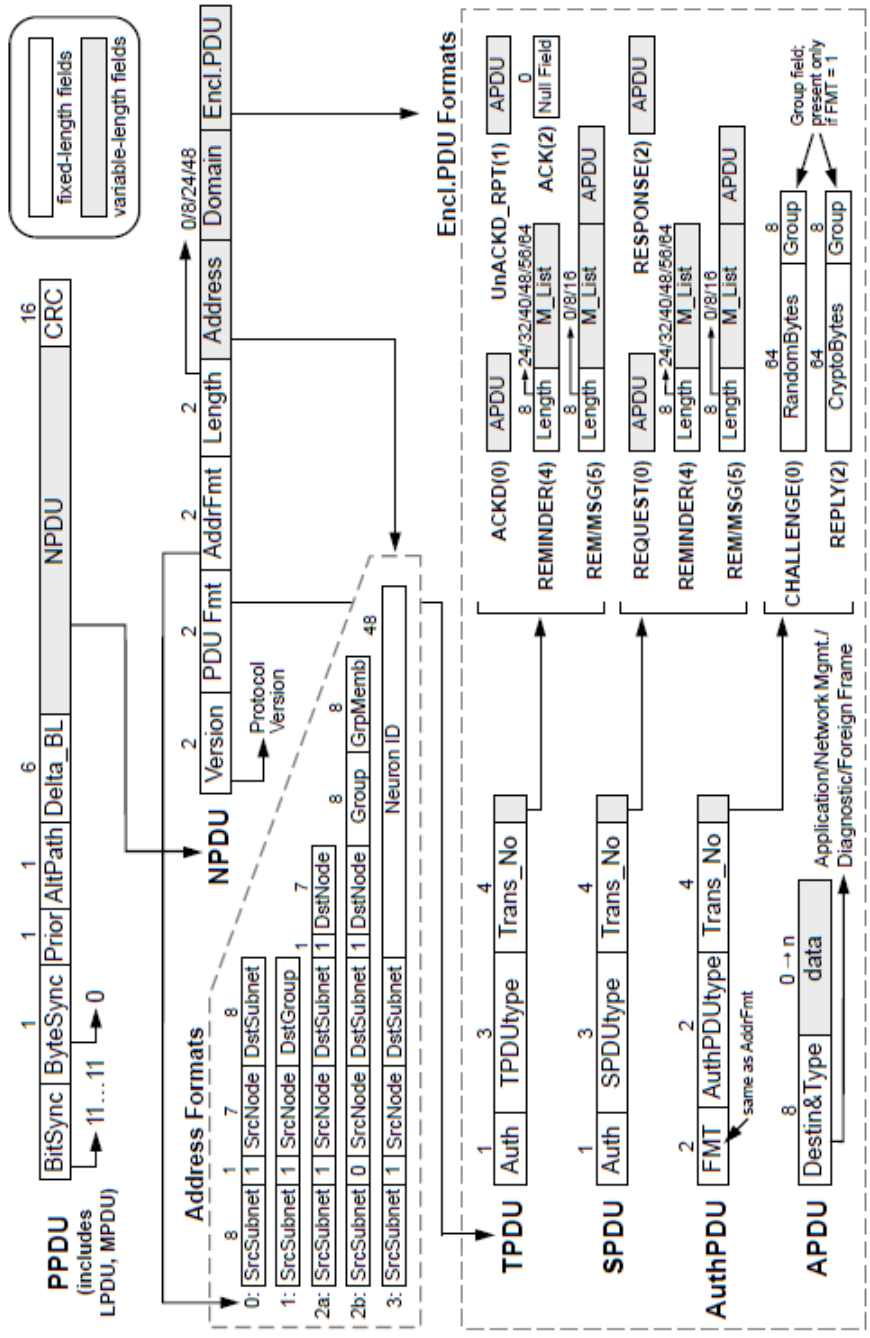
# A TVL: Travel Product Information

In deze appendix staat de specificatie voor een TVL tag zoals deze wordt gebruikt in EDIFACT berichten. De functie van deze tag is het specificeren van details voor een reis. Ik heb deze specificatie overgenomen van een website [15].
De eerste kolom in tabel A geeft het nummer van het element aan. De tweede kolom bevat het nummer van een specifieke component of element. Het nummer van een element begint dan ook met een E. De derde kolom bevat vervolgens de naam van deze componenten/elementen. Tenslotte geeft kolom vier aan of het een conditionele (**C**onditional) of verplichte (**M**andatory) component is. Echter als het conditionele element 030 voorkomt, dan moeten ook element 010 en 020 voorkomen. Een zelfde constructie geldt er voor de verschillende componenten binnen een element. Verder kun je zien dat als element 030 voorkomt, er binnen dit element zeker een "Party name" als eerste component moet staan, deze is immers verplicht. Echter, als element 030 niet voorkomt, dan geldt de verplichting voor de eerste component ook niet meer.

| | | | |
|---|---|---|---|
| 010 | E987 | PRODUCT DATE AND TIME | C |
| | 2000 | Date value | C |
| | 2002 | Time value | C |
| | 2000 | Date value | C |
| | 2002 | Time value | C |
| | 2148 | Date variation number | C |
| 020 | E975 | LOCATION | C |
| | 3225 | Location name code | C |
| | 3224 | Location name | C |
| | 3207 | Country name code | C |
| | 3227 | Location function code qualifier | C |
| 030 | E988 | COMPANY IDENTIFICATION | C |
| | 3036 | Party name | M |
| | 3036 | Party name | C |
| | 3036 | Party name | C |
| 040 | E989 | PRODUCT IDENTIFICATION DETAILS | C |
| | 7135 | Product identifier | C |
| | 7037 | Characteristic description code | C |
| | 7139 | Product characteristic identification code | C |
| | 7009 | Item description code | C |
| | 7009 | Item description code | C |
| | 7009 | Item description code | C |
| | 9608 | Product name | C |
| 050 | E990 | SEQUENCE NUMBER DETAILS | C |
| | 1050 | Sequence position identifier | M |
| | 1050 | Sequence position identifier | C |
| | 1050 | Sequence position identifier | C |
| | 1050 | Sequence position identifier | C |
| | 1050 | Sequence position identifier | C |
| | 1050 | Sequence position identifier | C |
| | 1050 | Sequence position identifier | C |
| | 1050 | Sequence position identifier | C |
| | 1050 | Sequence position identifier | C |
| | 1050 | Sequence position identifier | C |
| 060 | 1082 | LINE ITEM IDENTIFIER | C |
| 070 | 7365 | PROCESSING INDICATOR DESCRIPTION CODE | C |

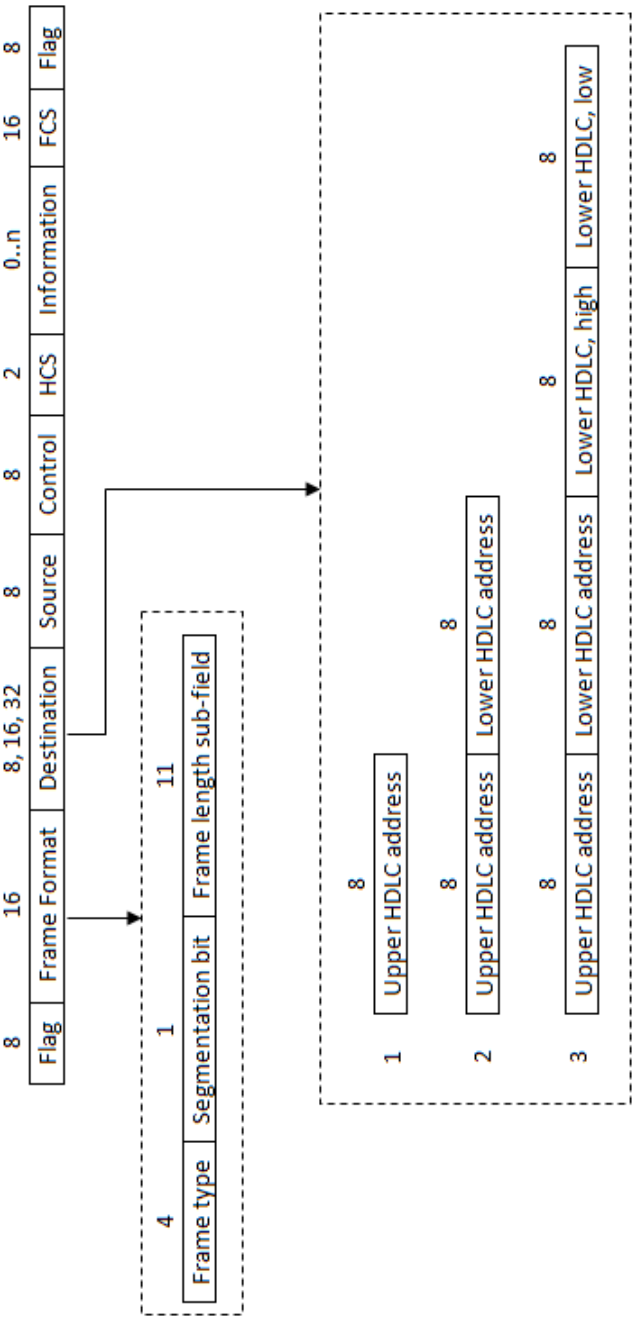
Tabel 6: Specificatie TVL element

# B PDU Samenvatting (LON)




Figuur 11: Overzicht PDU (LON) [19]

# C PDU Samenvatting (COSEM)




Figuur 12: Overzicht PDU (COSEM)

# Referenties

[1] http://www.ez.nl/Onderwerpen/Voldoende_energie/Werking_Kleinverbruikersmarkt/Metermarkt.

[2] http://www.volkskrant.nl/economie/article510259.ece/Slimme_meter_wint_het_van_zijn_domme_neefje.

[3] http://www.mijn-slimme-meter.nl/.

[4] http://www.ez.nl/content.jsp?objectid=150297&rid=150296.

[5] http://www.depers.nl/binnenland/298287/Slimme-meter-wordt-toch-vrijwillig.html.

[6] http://nl.wikipedia.org/wiki/Power_line_communication.

[7] http://www.wireshark.org/.

[8] http://www.aircrack-ng.org/.

[9] http://www.tcpdump.org/.

[10] http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=983b941d-06cb-4658-b7f6-3088333d062f.

[11] http://www.netscout.com/.

[12] http://www.dlms.com/information/whatisdlmscosem/index.html.

[13] http://www.computerwoorden.nl/direct--9766--Edifact.htm.

[14] http://nl.wikipedia.org/wiki/Electronic_Data_Interchange_For_Administration,_Commerce_and_Transport.

[15] http://www.unece.org/trade/untdid/d01a/tisd/tisdtvl.htm.

[16] DLMS User Association. Cosem blue book. Technical report, DLMS User Association, 2002.

[17] DLMS User Association. Cosem green book. Technical report, DLMS User Association, 2004.

[18] Cees-Bart Breunesse. Broncode lonscanner.

[19] Echelon. Lontalk protocol specification. Technical report, Echelon Corporation, 1994.

[20] Sander Keemink and Bart Roos. Security analysis of dutch smart metering systems. Master's thesis, Universiteit van Amsterdam, 2008.

[21] Daryl Kulak and Eamonn Guiney. *Use Cases*. Addison Wesley, second edition, 2004.

[22] G. Lenzini, M. Oostdijk, and W. Teeuw. Trust, security, and privacy for the advanced metering infrastructure. Technical report, Novay, 2009.

[23] Shalloway and Trott. *Design Patterns Explained*. Addison Wesley, second edition, 2007.