Error correctie in VoIP

Frank van der Loo
0314005

20 juni 2007

Inhoudsopgave

Hoofdstuk 1

Inleiding

VoIP staat voor Voice over IP. Het is de techniek om spraakgegevens te versturen via het Internet Protocol. Met andere woorden: bellen via het internet. De afgelopen jaren is het gebruik hiervan toegenomen.

Dit komt voornamelijk, omdat bellen via VoIP een stuk goedkoper is dan bellen via een gewone telefoon. Het internet is echter geen geschikt medium voor real-time communicatie: pakketten raken verloren, komen pas later aan, komen beschadigd aan en meer problemen. Toch kan men gewoon bellen met VoIP zonder veel last te hebben van deze problemen. Een van de redenen hiervoor is het gebruik van error correctie.

1.1 Probleemstelling

Nu kan men niet zomaar error correctie toevoegen aan een toepassing. De onderzoeksvraag die hier beantwoord gaat worden is dan ook "Op welke manier wordt error correctie gebruikt bij telefoneren via internet (VoIP)?".

Deze vraag zal beantwoord worden door de volgende deelvragen te beantwoorden:

Met welke protocollen werkt VoIP?

Waar in deze protocollen wordt gebruik gemaakt van error correctie?

Welke error-correcting codes worden gebruikt?

Hoe werken deze error-correcting codes?

Waarom worden juist deze codes gebruikt?

Hoofdstuk 2

VoIP

2.1 VoIP protocollen

Er zijn verschillende protocollen die gebruikt worden voor VoIP. Enkele hiervan, zoals gebruikt door Skype, zijn gesloten en geheim. Maar de meest gebruikte protocollen zijn open. Deze zijn H.323 en SIP (Session Initiation Protocol) voor het opzetten van de verbinding en RTP (Real-time Transport Protocol) en RTCP (Real-Time Control Protocol) voor het versturen van het geluid.

2.1.1 H.323

Het H.323 protocol is bedacht door de ITU (International Telecommunication Union) in 1996. Het is een zogenaamde "paraplu" specificatie, omdat het protocol alleen maar beschrijft hoe andere protocollen gebruikt moeten worden. De belangrijkste andere protocollen die gebruikt worden zijn H.225.0, RTP, RTCP en H.245. H.225.0 wordt gebruikt voor "call signaling". Dat wil zeggen het opzetten, beheren en beëindigen van een gesprek. RTP en RTCP worden gebruikt voor het versturen van de gesprekken. H.245 wordt gebruikt voor het openen van een verbinding (kanaal) tussen het begin- en eindpunt en de onderhandeling over de mogelijkheden van beide partijen en dus de gebruikte codecs. H.323 maakt, in tegenstelling tot andere protocollen zoals HTTP, geen gebruik van textberichten, maar van een efficiënte binaire codering (PER (Packed Encoding Rules))[PACKE98].

H.323 specificeert vier componenten[DATAB99]: terminals, gateways, gatekeepers en Multipoint Control Units.

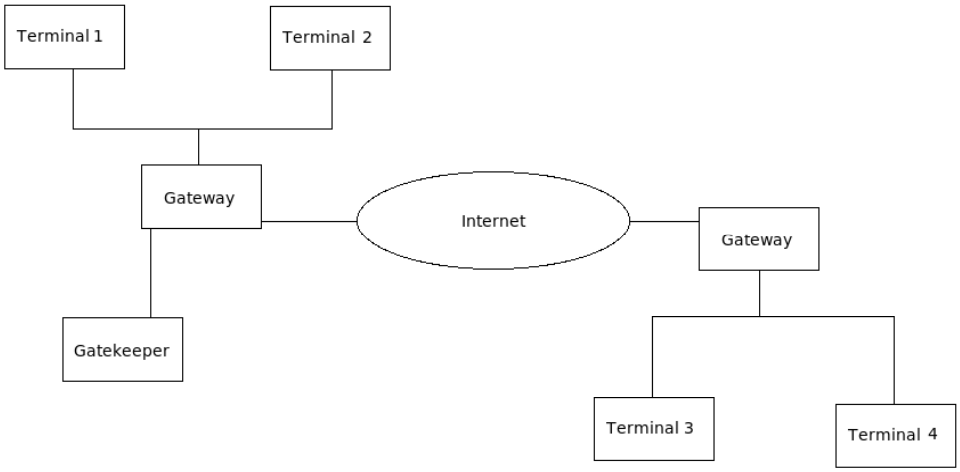
Terminals zijn de clients die met elkaar communiceren via het H.323 protocol.

Gateways zijn optioneel. Ze verzorgen de verbinding met andere H.323 netwerken en zorgen voor de eventueel benodigde vertaling voor de verbinding. Een gateway is dus niet nodig indien er een verbinding wordt opgezet tussen twee clients in hetzelfde netwerk.

Een gatekeeper is ook optioneel, maar als er een aanwezig is moet hij ook gebruikt worden door de terminals. Een gatekeeper voert twee functies uit. De

eerste is het vertalen van aliassen naar netwerkadressen. De tweede functie die hij uitvoert is het beheer van de bandbreedte. De gatekeeper kan de hoeveelheid bandbreedte gebruikt door verbindingen via H.323 beperken. Een derde, optionele, functie van de gatekeeper is om verbindingen via een bepaalde route te sturen. Hierdoor is er meer controle over de gesprekken. Dit kan bijvoorbeeld gebruikt worden om gebruiker te laten betalen voor gesprekken.

Een Multipoint Control Unit is alleen nodig voor gesprekken tussen meer dan twee terminals.



Twee H.323 netwerken met elk twee terminals. Terminal 1 kan direct een gesprek opzetten met terminal 2 en andersom. Hetzelfde geldt voor terminal 3 en 4. Wil terminal 1 een gesprek opzetten met terminal 3 wordt de gatekeeper gebruikt om het adres van terminal 3 op te vragen en wordt via de gateways en het internet een gesprek opgezet.

Bij het opzetten van een gesprek tussen twee terminals moet eerst, zoals in H.225.0 beschreven, een gesprek opgezet worden. Daarna wordt, zoals in H.245 beschreven, door de terminals besproken welke mogelijkheden ze hebben en dus welke codecs worden gebruikt voor de data overdracht. Ook opent H.245 een logisch kanaal voor de overdracht van het geluid.

<table>
<tr><th>Layer</th><th>Signaling</th><th>Registration</th><th>Audio</th><th>Video</th><th>Data</th><th>Security</th></tr>
<tr><td rowspan="2">5</td><td rowspan="2">H.225.0-Q.931
H.250-Annex G
H.245
H.250</td><td rowspan="2">H.225.9-RAS</td><td>G.711
H.263
G.722
G.723
G.728</td><td>H.261
H.323</td><td rowspan="2">T.120</td><td rowspan="2">H.235</td></tr>
<tr><td colspan="2">RTP, RTCP</td></tr>
<tr><td>4</td><td>TCP, UDP</td><td colspan="3">UDP</td><td>TCP</td><td>TCP, UDP</td></tr>
<tr><td>3</td><td colspan="6">IP, RSVP, and IGMP</td></tr>
</table>

Een overzicht van de gelaagdheid van de H.323 protocollen[COMER04].

2.1.2 SIP

Het SIP (Session Initiation Protocol) is een protocol bedacht door de IETF (Internet Engineering TaskForce). In tegenstelling tot H.323 maakt SIP, net zoals andere veelgebruikte protocollen zoals HTTP, gebruik van textberichten.

SIP specificeert vijf componenten[LIESE00, HANDL99]: User Agents, Proxy Server, Redirect Server, Registrar Server en Location Server. Een User Agent is een van de partijen die met een andere communiceert via het SIP protocol. Een User Agents kan zowel een Client (als hij iemand belt) als een Server (als hij gebeld wordt) zijn.

Een Proxy Server wordt alleen gebruikt bij het opzetten en verbreken van een verbinding. Zijn taak is om berichten (zoals een binnenkomend gesprek) naar de goede User Agents te sturen. Om dit te doen kan het nodig zijn dat de proxy server een deel van het bericht herschrijft voor het door te sturen.

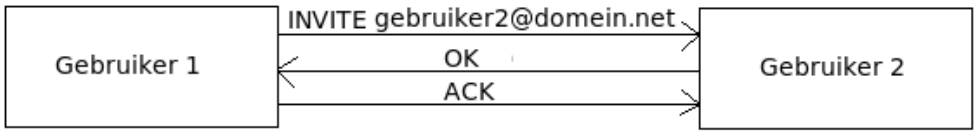
Een Redirect Server levert de benodigde informatie om een gesprek met een User Agent op te zetten als antwoord op een uitgaand gesprek naar die User Agent. In tegenstelling tot een proxy server stuurt een Redirect Server het bericht niet door, maar levert enkel de informatie nodig om de User Agent te bereiken.

Een Registrar Server krijgt informatie over de locaties van de User Agents van een bepaald domein van de User Agents. Deze informatie wordt door de Registrar Server in een database opgeslagen. Deze informatie wordt gebruikt om een gesprek op te zetten met een User Agent.

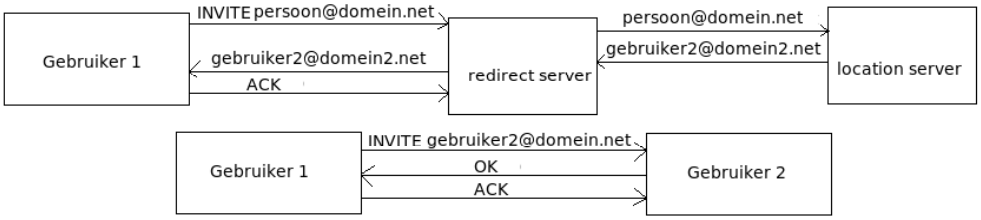
De Location Server gebruikt de informatie in de database van de Registrar Server om een Redirect Server de locatie van een User Agent te sturen.

Bij SIP netwerken worden gebruikers aangeduid met een een SIP-URL[LIESE00]. Zo'n URL ziet er uit als "sip:persoon@domein.net". Om iemand te bellen wordt een uitnodiging gestuurd naar het domein in de SIP-URL. Als de gebruiker daar

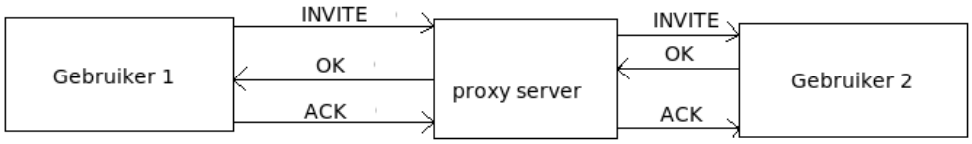
aanwezig is kan hij een antwoord sturen en zo laten weten of hij het gesprek accepteert of niet. De beller stuurt na ontvangst van het antwoord nog een antwoord om aan te geven dat hij het antwoord ontvangen heeft.



De uitnodiging kan ook verstuurd worden naar een Redirect Server. Deze server zoekt dan de mogelijke locaties van de gebruiker op en levert de bijbehorende SIP-URLs. Hierna kan de beller deze URLs gebruiken om de gebruiker te bellen.



De laatste mogelijkheid is om de uitnodiging naar de Proxy Server te sturen. De proxy server zal dan (evt. door gebruik te maken van een Location Server) de locaties van de gebruiker opzoeken en uitnodigen. De proxy server zal dan het antwoord van de gebruiker sturen naar de beller.



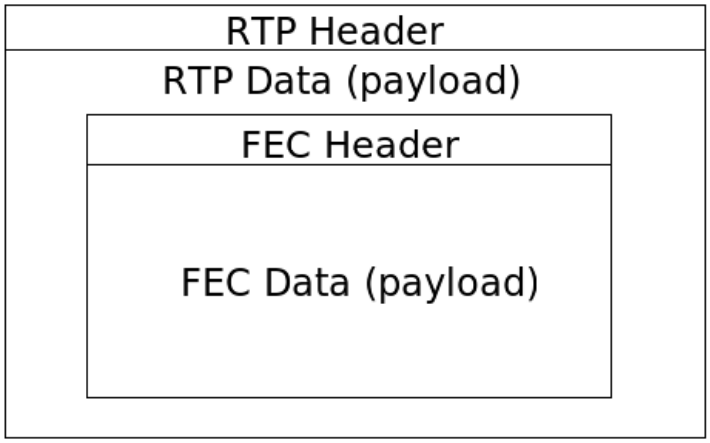
2.1.3 RTP

De SIP en H.323 protocollen worden, zoals eerder al vermeld, enkel gebruikt om een verbinding op te zetten en te beheren. Voor het verzenden van het geluid (het telefoongesprek) zelf wordt het RTP (Real-time Transport Protocol) en RTCP (Real-Time Control Protocol) protocol gebruikt.

De gebruikte protocollen op het internet (TCP en UDP) zijn niet betrouwbaar. TCP pakketten kunnen later aankomen en UDP pakketten kunnen helemaal niet aankomen. Hierdoor is het internet eigenlijk niet geschikt voor real-time communicatie. Om applicaties toch toe te staan gebruik te maken van real-time gegevens is het RTP protocol bedacht[LIU98].

Vaak wordt UDP gebruikt als onderliggend protocol, maar RTP is zo opgezet dat hij werkt met elk protocol. Om een RTP pakket te versturen wordt een RTP pakket (header en payload) als payload in een ander pakket gebruikt (meestal UDP). Dit pakket wordt dan verzonden. Omdat het RTP protocol de

data die verstuurd moet worden bevat, bevat dit protocol ook de error correctie[ROSEN99]. Dit wordt bereikt door een FEC pakket (met de gegevens nodig voor error correctie) als payload van een RTP pakket te versturen.



Een RTP pakket, met FEC gegevens als payload.

2.1.4 RTCP

Het RTCP protocol dat vaak samen met RTP gebuikt wordt heeft vier functies[LIU98].

1. Informatie over de kwaliteit van de verbinding leveren aan het programma dat gebruik maakt van RTP. Zo bevat het gegevens als aantal pakketten verzonden, aantal pakketten verloren gegaan enz. Deze gegevens kunnen gebruikt worden om een andere manier te kiezen om data versturen en om problemen in het netwerk te ontdekken.

2. Identificeren van de bron. RTCP bevat een zgn "canonical name" van de bron. Dit wordt gebruikt om bij te houden wie er allemaal deelnemen in een RTP sessie.

3. Controlen van het RTCP verzendinterval. Om te voorkomen dat het netwerk overbelast raakt en om veel personen deel te laten nemen aan een RTP sessie wordt de hoeveelheid controle verkeer beperkt tot 5 procent van het totale verkeer. Omdat iedereen RTCP pakketten stuurt naar anderen kan iedereen berekenen welk percentage van het verkeer bestaat uit controle pakketten en indien nodig de hoeveelheid controle pakketten aanpassen.

4. De laatste, en optionele functie, is het sturen van informatie over de deelnemers (bijv. de naam) naar anderen.

RTCP lijkt qua doel enigzins op TCP. Beide protocollen zijn "Control Protocollen". Toch zijn er een aantal grote verschillen:

TCP wordt gebruikt om gegevens via het internet te versturen. RTCP wordt alleen gebruikt om gegevens over de kwaliteit van het netwerk te versturen en niet de gegevens zelf.

TCP maakt gebruik van verbindingen. Voor er gegevens verstuurd kunnen worden moet er eerst een verbinding worden opgezet en na het versturen weer worden gesloten. RTCP doet dit niet per se. RTCP kan de gegevens versturen met een TCP pakket, maar andere protocollen (zoals UDP) kunnen ook gebruikt worden.

Nog een verschil is dat TCP paketten die verloren zijn gegaan opnieuw stuurt. Bij RTCP wordt dit alleen gedaan als TCP gebruikt wordt als onderliggend protocol.

2.2 VoIP Problemen

Ook al wordt VoIP steeds meer gebruikt dat wil niet zeggen dat VoIP zonder problemen werkt. Om gebruik te kunnen maken van VoIP moeten er een aantal problemen opgelost worden.

2.2.1 Firewalls

VoIP moet verbindingen opzetten en ontvangen. Dit wordt echter vaak geblokkeerd door middel van een firewall. Ook verbinden veel mensen tegenwoordig via een router. Om gebruik te kunne maken van VoIP moet zowel de firewall als de router zo ingesteld worden dat de computer verbindingen kan openen en ontvangen.

2.2.2 Packet loss

Een groot probleem van het verzenden van gegevens via het internet is het feit dat het internet geen zekerheid biedt dat de gegevens ook daadwerkelijk aankomen. Zelfs als de pakketten aankomen is het niet zeker dat ze onderweg niet veranderd zijn door ruis.

Normaal gesproken wordt hier rekening mee gehouden en wordt er gebruik gemaakt van het TCP protocol. Bij dit protocol moet er een bevestiging worden gestuurd voor elk ontvangen pakket. Alle pakketten waarvoor geen bevestiging wordt ontvangen worden opnieuw verstuurd.

VoIP protocollen maken hier geen gebruik van. Dit is omdat het versturen van bevestigingen en pakketten opnieuw versturen veel vertraging en overhead oplevert.

Het optreden van packet loss kan een gesprek via VoIP verstoren. Zeker omdat packet loss vaak achter elkaar optreedt[VOIPT06]. Vaak raken een heleboel pakketten die achter elkaar verzonden worden verloren. Zelfs als het gemiddeld aantal verloren gegane pakketten laag is kunnen dergelijke periodes met veel packet loss de kwaliteit van VoIP behoorlijk negatief beïnvloeden.

2.3 Oplossingen

Voor deze problemen zijn verschillende oplossingen bedacht. Voor het probleem van de firewalls/routers zijn weinig technische oplossingen mogelijk, omdat dit gaat om de instellingen van de gebruiker. Voor het optreden van packet loss zijn wel technische oplossingen mogelijk.

2.3.1 Opnieuw sturen

Een oplossing voor het probleem van packet loss is om, net zoals het TCP protocol, gebruik te maken van Automatic Repeat Request (ARQ). Hierbij moet er voor elk ontvangen pakket een bevestiging worden gestuurd. Pakketten waarvoor geen bevestiging wordt gestuurd worden opnieuw verzonden.

Het grote nadeel van deze oplossing is dat het een vertraging oplevert en overhead. Voor elk ontvangen pakket moet namelijk een pakket verstuurd worden. Ook moet er gewacht worden totdat elke verloren gegane pakket opnieuw is verstuurd. Dit vertraagt een telefoongesprek.

2.3.2 Packet Loss concealment

Een veelgebruikt methode is packet loss concealment. Dit is een methode om de effecten van pakketten die niet zijn aangekomen te verbergen. Het is natuurlijk niet mogelijk om alle effecten van packet loss te verbergen, maar zo lang maximaal 20-30 milliseconden spraak verloren gaat is het mogelijk[VOIPT06].

Er zijn verschillende manieren om packet loss te verbergen, maar ze komen allemaal neer op het vervangen van ontbrekende pakketten. Deze pakketten kunnen vervangen worden door de laatst ontvangen pakketten, door het gemiddelde van de pakketten voor en na het verloren pakket, door stilte, door Gaussiaanse ruis of door eerdere pakketten die goed passen[MEHTA05].

De mate waarin packet loss concealment effectief is, is afhankelijk van de hoeveelheid pakketten die verloren zijn gegaan en de kwaliteit van de gebruikte packet loss concealment algoritmen.

2.3.3 Forward Error Correction

Forward Error Correction (FEC) is een manier om fouten tijdens dataoverdracht te repareren. Hierbij worden extra (redundante) bits toegevoegd aan de gegevens die verzonden te worden. De ontvanger kan hier gebruik van maken om fouten te ontdekken en (tot op zekere hoogte) repareren.

Er bestaan twee verschillende soorten FEC: block codes en convolutional codes.

> Block codes werken met hele blokken van een bepaalde grootte. Deze blokken van een aantal bits worden omgezet in een ander blok van een bepaald aantal symbolen.

Convolutional codes daarentegen werken met bitreeksen met een willekeurige lengte. Ook hierbij wordt een aantal bits omgezet in een bepaald aantal symbolen. Maar nu wordt er ook gebruik gemaakt van symbolen die al eerder zijn gegenereerd.

De error correcting code bestaat uit de symbolen (het codeword) en de mogelijke mappings. De decoder dient bij het decoderen het codeword te vinden dat het meeste lijkt op de gegevens die het ontvangt.

2.3.4 Audio CD

De door VoIP gebruikte technieken om de effecten van packet loss tegen te gaan lijken erg op de technieken gebruikt door de audio CD om de effecten van krassen en andere beschadigingen op een audio CD tegen te gaan. Audio CD's gebruiken namelijk ook concealment en error correction bij het afspelen van beschadigde CD's.

Bij het schrijven van de gegevens naar de CD wordt gebruik gemaakt van Cross-Interleaved Reed-Solomon Code (CIRC)[HANLE02]. Hierbij worden de bits die het geluid representeren gesplitst in 24 blokken van 8 bits (1 byte). Deze 24 blokken worden met Reed-Solomon code gecodeerd tot 28 blokken van 8 bits. 4 van deze blokken bestaan uit pariteitsbits. Hiermee kunnen fouten in 2 blokken (16 bits) gerepareerd worden. De gegevens van deze 28 blokken worden geïnterleaved (in een andere volgorde gezet). Zo wordt voorkomen dat een compleet blok beschadigd raakt door bijvoorbeeld een kras. Deze code wordt de C2 codering genoemd. De gegevens van deze codering worden nogmaals met een Reed-Solomon code gecodeerd. Nu tot 32 blokken. Ook deze code kan fouten in 2 blokken repareren en weer worden de gegevens geïnterleaved (ditmaal met een ander pattroon dan de eerste keer). Deze code wordt de C1 codering genoemd. Hiermee kunnen tot 3874 opeenvolgende foute bits worden gerepareerd. Dit komt overeen met een beschadiging van ongeveer 2,6 mm op de CD.[MCLEA06].

Als er meer foute bits op de CD staan wordt er gebruik gemaakt van concealment. Hierbij wordt door middel van interpolatie een stuk geluid gegenereerd dat wordt afgespeeld in plaats van de beschadiging. Voor deze interpolatie wordt gebruik gemaakt van de bits voor en na het beschadigde deel. Op deze manier kunnen tot 13.282 opeenvolgende foute bits worden verborgen. Dit komt overeen met een beschadiging van ongeveer 8,9 mm op de CD.

Hoofdstuk 3

Error correction

3.1 Geschiedenis

In 1948 was Claude Shannon aan het werken aan de capaciteit van communicatiekanalen[BURR01]. Hij heeft aangetoond dat een communicatiekanaal in principe informatie kan versturen met een bepaald aantal fouten, zelfs als er veel ruis of inferentie op het kanaal zit, zo lang de capaciteit van het kanaal niet overschreden wordt. Deze capaciteit is afhankelijk van de signaal-ruis verhouding. Om deze capaciteit zo groot mogelijk te maken kan er gebruik gemaakt worden van technieken om fouten te herstellen. De theoretische maximumsnelheid van het kanaal bij een bepaalde hoeveelheid ruis wordt de Shannon limiet genoemd.

Er zijn twee manieren om fouten bij het versturen van gegevens te herstellen:

Automatic Repeat Request (ARQ)

Forward Error Correction (FEC)

Bij ARQ wordt er na de gegevens een controlecode verstuurd. Met deze code kan de ontvanger controleren of er fouten zijn opgetreden tijdens het versturen van de gegevens. Als er geen fouten zijn opgetreden stuurt hij een bericht terug naar de verzenden om te laten weten dat de gegevens goed zijn ontvangen. Alle gegevens waarvoor geen bericht terug wordt gestuurd worden opnieuw verstuurd door de afzender.

Bij FEC wordt door de verzender bij het versturen extra informatie toegevoegd aan de spraakgegevens. Deze extra gegevens stellen de ontvanger in staat om eventuele corrupte of verloren gegane pakketten te herstellen.

3.2 Soorten forward error correction

Er bestaan twee soorten forward error correction[CHEN03][ALTMA01][PERKI98]:

Media-specifieke codes

Media-onafhankelijke codes

3.2.1 Media-specifieke codes

Bij media-specifieke codes wordt elk pakket twee of meer keer verstuurd. Deze dubbele pakketten zijn niet helemaal gelijk aan het oorspronkelijke pakket. Ze hebben een lagere kwaliteit, om zodoende de extra bandbreedte die hiervoor nodig is te beperken. Als een pakket nu verloren gaat kan dit pakket vervangen worden door zo'n extra pakket, waardoor de gebruiker nauwelijks iets merkt.

De naam media specifiek is gebaseerd op het feit dat er gebruik wordt gemaakt van de eigenschappen van de manier waarop de data (de media) verstuurd wordt. Als de data bestaat uit pakketten worden deze dubbel verstuurd. Als de data bestaat uit meerdere pakketten (dus niet in één pakket past) worden alleen de belangrijkste delen vaker verstuurd.

Bij VoIP passen de gegevens in enkele pakketten en worden, als media-specifieke FEC wordt gebruikt, alle pakketten vaker verstuurd.

Deze vorm van FEC heeft als voordeel dat het nauwelijks voor vertraging zorgt, omdat er op maar één pakket gewacht hoeft te worden. Het nadeel is wel dat een extra pakket groter is dan een recovery pakket en dus meer bandbreedte verbruikt.

3.2.2 Media-onafhankelijke codes

Media-onafhankelijke codes zijn de meest bekende error-correcting codes. Hierbij worden voor blokken pakketten die verstuurd gaan worden eerst met een error-correcting code extra pakketten gegenereerd. Deze pakketten worden ook verstuurd en kunnen door de ontvanger gebruikt worden om verloren pakketten te herstellen, mits er voldoende pakketten onbeschadigd aangekomen zijn.

De voor- en nadelen van deze vorm zijn het tegenovergestelde van de media-specifieke codes. Media-onafhankelijke codes verbruiken minder bandbreedte, maar leveren wel meer vertraging op. Er moet namelijk gewacht worden op een compleet blok pakketten.

3.3 Bestaande error correcting codes

Sinds Shannons ontdekking zijn er verschillende error-correction codes ontwikkeld. De bekendste zijn:

Hamming code

Reed-Solomon code

Reed-Muller code

Convolutional code

Turbo code

De Hamming code werd uitgevonden door Richard Hamming in 1948. Het is gebaseerd op het opsplitsen van de gegevens in blokken bits. In deze blokken worden de bits genummerd[TERVO05]. Op bepaalde plaatsen (alle machten van 2) worden controlebits toegevoegd aan de gegevens. Elke controlebit controleert een aantal gegevensbits. Zo'n controlebit is een 0 als het aantal gegevensbits die hij controleert even is en een 1 als het aantal gegevensbits die hij controleert oneven is.

Helaas kunnen Hamming codes alleen maar fouten repareren als er slechts één bit anders aankomt dan het verzonden is. Dit is een groot nadeel waardoor Hamming codes vandaag de dag nauwelijks meer gebruikt worden.

Reed-Muller codes zijn ontdekt door Irving Reed en D.E. Muller in 1954[RAAPH03]. Hiermee zijn ze een van de oudste error correcting codes. Een groot nadeel van Reed-Muller codes is echter dat ze zwakker worden naarmate hun lengte groter wordt. Desondanks worden ze nog steeds gebruikt als basis blokken voor andere codes. Het grootste voordeel van Reed-Muller codes is dat ze relatief simpel te coderen en te decoderen zijn.

Reed-Solomon codes, Convolutional codes en Turbo codes worden verderop uitvoeriger besproken.

3.4 Gebruikte error correcting codes voor VoIP

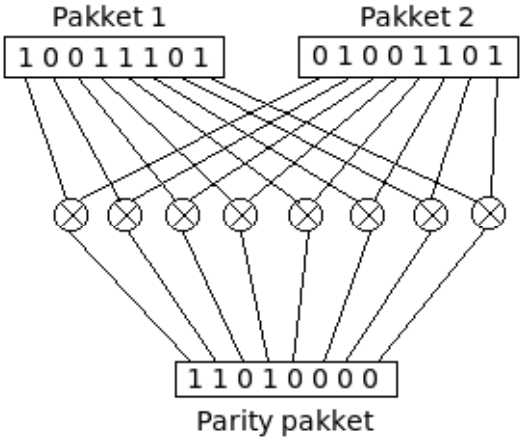
Niet elke error correcting code is geschikt voor gebruik in VoIP. Dit komt voornamelijk omdat het bij VoIP belangrijk is dat de error correction snel werkt. Als de error correction traag werkt merkt de gebruiker dit doordat er dan "gaten" vallen in het gesprek.

Error correcting codes die veel gebruikt worden in VoIP zijn Reed-Solomon codes en Parity codes. Sinds kort worden ook Turbo codes geïmplementeerd in VoIP programma's.

3.5 Parity codes

Parity codes zijn een hele simpele vorm van error correctie. Dit is dan ook één van de redenen dat ze gebruikt worden in VoIP. Door de simpelheid is de vertraging minimaal[ROSEN99].

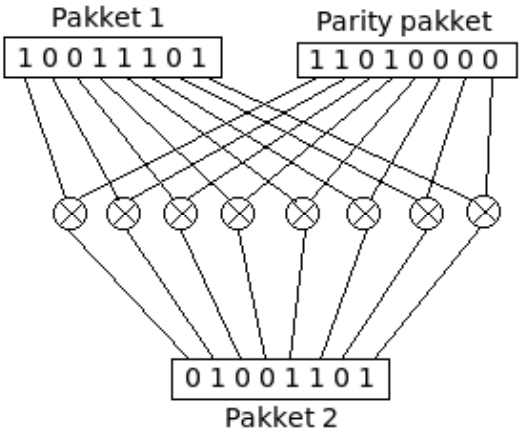
Bij parity codes wordt de XOR operator bit voor bit toegepast op een aantal pakketten. Dit levert een nieuw pakket op, het zgn. parity pakket.



Het genereren van een parity pakket uit pakket 1 en pakket 2.

Als nu een pakket verloren gaat of beschadigd aankomt kan dit parity pakket gebruikt worden om dat pakket te reconstrueren. Hiervoor is het wel nodig dat de andere pakketten die gebruikt zijn om het parity pakket te maken en het parity pakket zelf goed aankomen.

Het reconstrueren van het pakket gaat op dezelfde manier als het maken van het parity pakket. Weer wordt de XOR operator bit voor bit toegepast. Ditmaal echter op het parity pakket en de goed aangekomen pakketten. Dit levert dan het pakket op dat beschadigd of helemaal niet is aangekomen.



Pakket 1 en het parity pakket worden gebruikt om pakket 2 te reconstrueren.

Het grote voordeel van het gebruik van parity codes is dat het een hele simpele vorm van error correctie is. Hierdoor kost het reconstrueren van een pakket nauwelijks tijd. Ook is het door deze eenvoud eenvoudig voor ontwikkelaars om

deze codes te implementeren in een VoIP programma.

Deze simpelheid is ook meteen het grootste nadeel van parity codes. Omdat ze zo simpel zijn is de hoeveelheid fouten die ze kunnen herstellen klein. Parity codes kunnen slechts één pakket uit elk blok herstellen. Ook zijn parity codes hierdoor heel gevoelig voor packet loss en packet corruption. Als het parity pakket niet (onbeschadigd) aankomt is correctie niet meer mogelijk. Bovendien, omdat parity codes maar één pakket uit een blok pakketten kan herstellen is het voor een goede error correctie nodig om het aantal pakketten per blok klein te houden. Dit zorgt voor veel overhead.

3.6 Turbo codes

3.6.1 Geschiedenis

Turbo codes zijn ontdekt door Claude Berrou, Alain Glavieux en Punya Thitimajshima[SAYIR05]. In 1993 hebben ze het voor elkaar gekregen om een error correcting code te bedenken die de Shannon limit benaderde. Hierop publiceerden ze hierover hun paper "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo codes".

De resultaten waren echter zo goed dat de resultaten in eerste instantie met argwaan bekeken werden. Hier kwam nog bij dat de onderzoeksgroep en de universiteit waar ze voor werken onbekend waren in de wereld van de informatie theorie. Ook zag hun paper en onproffessioneel uit en bevatte spelfouten. Veel onderzoekers waren er van overtuigd dat ze een fout hadden gemaakt en begonnen de resultaten te controleren met het doel om aan te tonen dat de resultaten niet kloppen.

Dit was nog een vrij moeilijke klus, omdat veel details van de turbo codes onduidelijk waren en er een aantal foutjes in de paper stonden. Toen het gelukt was om de turbo codes te implementeren bleek echter dat ze het bij het rechte eind hadden. De resultaten werden door verschillende onderzoeksgroepen bevestigd.

3.6.2 Werking

Forward Error Correction (FEC) werkt door de gegevens op te splitsen in blokken met k bits. Elk blok wordt dan gemapt op een ander blok van n symbolen (het codeword). Omdat de decoder het codeword moet vinden dat het meeste lijkt op de bits die het ontvangt werkt FEC het beste als k en n heel groot zijn. Hierbij is het verschil tussen de codewords namelijk groter en zal de decoder dus minder verward worden door codewords die op elkaar lijken. Helaas heeft dit een negatief effect op de snelheid ervan.

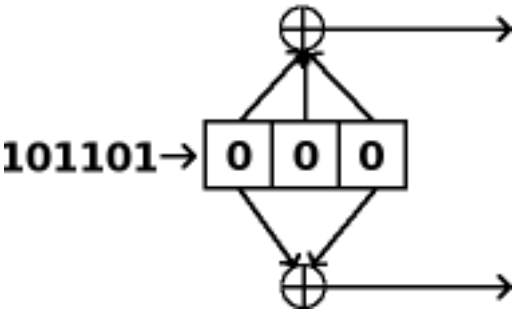
Turbo encoder

Bij Turbo codes wordt om deze reden gebruik gemaakt van lange complexe codes die opgebouwd worden uit kortere codes[BURR01]. Deze codes worden

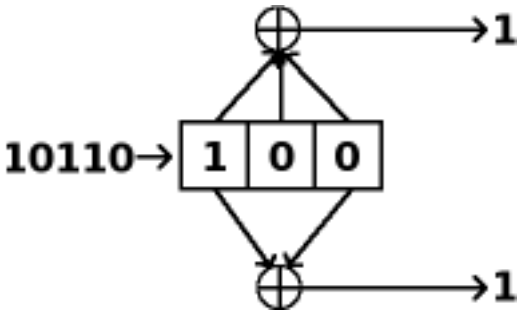
verkregen door de data parallel door twee convolutional encoders te coderen. De ene encoder codeert de data zoals hij binnenkomt en de andere encoder codeert de data nadat die door een interleaver in een andere volgorde is gezet.

Convolutional encoder

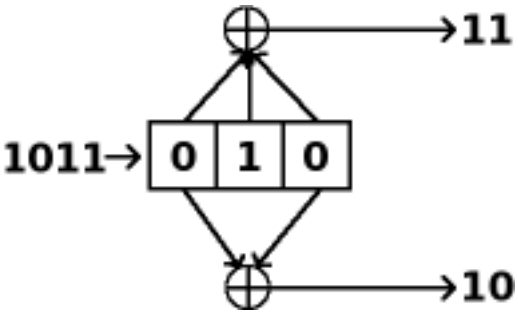
Een convolutional encoder bestaat uit een aantal registers, die elk één bit kunnen bevatten, een aantal XOR poorten en evenveel "generator polynomials". In het begin bevat elk register een nul. De XOR poorten leveren de uitvoer, dus het aantal XOR poorten (en het aantal "generator polynomials") bepaalt het aantal codes als uitvoer. Een generator polynomial is een vector van enen en nullen.



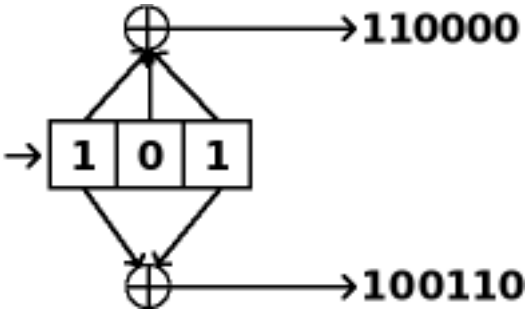
Om een reeks bits te coderen wordt het eerste bit in het meest linkse register geplaatst. Aan de hand van de "generator polynomials" en de resterende waarden van de registers levert elke XOR poort één bit uitvoer op. De "generator polynomials" bepalen welke registers gebruikt worden om via een XOR poort een bit op te leveren. Een generator polynomial bestaat uit evenveel getallen (een één of een nul) als het aantal registers. Elke één in een generator polynomial wil zeggen dat de waarde in het register dat overeenkomt met de positie van de één in de generator polynomials wordt gebruikt bij het berekenen van de uitvoer. Het aantal enen en de posities daarvan in een generator polynomails bepalen dus hoeveel en welke registers worden ge-XORed.



Hierna worden de waarden van de registers naar rechts geshift en gewacht op het tweede bit. Als het tweede bit in het meest linkse register is geplaatst worden er weer bits gegenereerd.



Dit gaat zo door tot alle invoerbits geweest zijn. In dat geval gaat de encoder door met bits uitvoeren en shiften tot alle registers weer de waarde nul bevatten. De uitgevoerde bits worden vervolgens met een multiplexer samengevoegd tot een reeks bits.

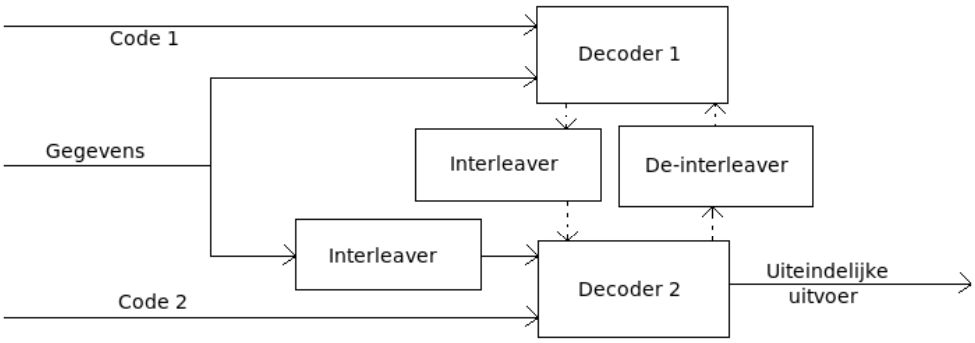


Dergelijke encoders bestonden al langer. Dit is dan ook niet de reden van de grote kracht van turbo codes. Die zit in de decoder.

Turbo Decoder

Bij het decoderen van turbo codes worden de codes iteratief gedecodeerd. De codes worden dus meerdere keren door de decoders gedecodeerd. Voor elke bit in de code wordt een getal tussen de -127 en 127 geproduceerd dat aangeeft hoe waarschijnlijk het is dat die bit nul of één is. -127 betekent dat de bit zeker een nul is, een nul betekent dat niet bekend is of de bit een nul of een één is. 127 betekent dat de bit zeker een één is. Alle negatieve getallen betekenen dat de bit waarschijnlijk een nul is en alle positieve getallen dat de bit waarschijnlijk een een is.

Om een een blok bits te decoderen wordt er een blok waarschijnlijkheidswaarden gemaakt waarbij er voor elke bit een waarschijnlijkheidswaarde is. Hierna zijn er twee parallelle decoders die van deze waarden gebruik maken om de code te decoderen. Hiervoor worden convolutional decoders gebruikt. Elke decoder decodeert de bits en berekent een waarschijnlijkheid voor de bits. De decoders vergelijken deze decoderingen. Als ze verschillen wisselen ze de waarschijnlijkheidswaarden uit en genereren ze beiden een nieuwe decodering met deze nieuwe waarschijnlijkheidswaarden. Dit wordt zo een vast aantal keer gedaan of totdat beide decoders hetzelfde opleveren.



Elke decoder krijgt de ontvangen gegevens en een code binnen om te decoderen (doorgetrokken lijnen). Om deze te decoderen wisselen de decoders informatie uit (stippellijnen).

Convolutional Decoder

Er zijn verschillende algoritmes die gebruikt kunnen worden voor het decoderen van convolutional codes. Het meeste wordt echter het Viterbi algoritme gebruikt.

3.6.3 Voor- en nadelen

Turbo codes worden tegenwoordig veel gebruikt, maar net als alle andere error correcting codes hebben Turbo codes voor- en nadelen die Turbo codes geschikter maken voor bepaalde toepassingen dan voor andere.

Voordelen

Het grote voordeel van Turbo codes ten opzichte van alle andere error correcting codes is natuurlijk de enorme hoeveelheid fouten die ermee gerepareerd kunnen worden, waardoor de Shannon limit benaderd wordt[BURR01]. Naast andere codes die ook een goede error correctie halen hebben Turbo codes ook een structuur[HUANG97]. Hierdoor kunnen Turbo codes efficiënt gedecodeerd worden.

Nadelen

Een groot nadeel van Turbo codes is dat ze vrij langzaam zijn in het gebruik. Dit heeft twee redenen. Ten eerste maken Turbo codes gebruik van een interleaver. Deze interleaver moet een blok ter grootte van de interleaver bevatten. Hoe groter deze interleaver hoe groter de code die in de interleaver zit en hoe groter de vertraging die dit oplevert[BURR01]. Een bijkomend probleem is dat de error correcting kwaliteit van turbo codes onder andere afhankelijk is van de grootte van de interleaver. Hoe groter de interleaver hoe beter de kwaliteit[BARBU99]. Maar dit levert weer veel vertraging op. Ten tweede moeten de twee decoders tijdens het decoderingsproces met elkaar overleggen en opnieuw decoderen. Deze

stappen kosten ook veel tijd. Een ander nadeel van Turbo codes is dat de kwaliteit afneemt als het aantal fouten per verzonden bits (Bit Error Rate) laag is.

3.7 Reed-Solomon codes

Turbo codes worden pas sinds kort gebruikt voor VoIP. Hiervoor werden voornamelijk Reed-Solomon codes gebruikt.

Reed-Solomon codes zijn bedacht door Irving Reed en Gustave Solomon in 1960[CIPRA93]. Ondanks het feit dat ze meer dan 45 jaar geleden zijn bedacht worden ze nog steeds veel gebruikt. Enkele bekende toepassingen van Reed-Solomon codes zijn in CDs, DVDs en DSL.

Om door middel van Reed-Solomon codes een stroom van spraakgegevens te kunnen coderen, is het als eerste nodig om de gegevens op te splitsen in blokken. Omdat de gegevens via het internet worden verstuurd is dat al gebeurd. De gegevens worden opgesplitst in pakketten die verstuurd worden. Met Reed-Solomon en deze pakketten worden recovery pakketten gemaakt, die later gebruikt kunnen worden om een bepaalde hoeveelheid verloren gegane pakketten te herstellen. Deze, en de pakketten met de echte gegevens, worden verzonden. Voor het versturen van beide soorten pakketten wordt gebruik gemaakt van het RTP protocol[ROSEN99].

Reed-Solomon codes worden genoteerd als RS(n,k). n is het totaal aantal symbolen (pakketten) per blok. k is het aantal data symbolen (pakketten) per blok. n-k is dan het aantal parity symbolen (pakketten) per blok. Een Reed-Solomon code kan (n-k)/2 foute symbolen (pakketten) herstellen.

Een simpele vorm van error correctie met Reed-Solomon is de (3,2) Reed-Solomon Block code[RAAKE06]. Dit wordt genoteerd als RS(3,2). Dit wil zeggen dat deze code 2 symbolen (pakketten) gebruikt. Hier worden parity symbolen (pakketten) voor gemaakt zodat er een codewoord van 3 symbolen (pakketten) uit ontstaat. De parity pakketten kunnen gebruikt worden om fouten te herstellen. Het aantal parity symbolen is 3 - 2 = 1. Dit codewoord bestaat dus uit 3 pakketten. Hiervan zijn er 2 data pakketten en 1 parity pakket.

Dit parity pakket wordt op dezelfde manier gemaakt als bij de parity codes: door de bits paarsgewijs op te tellen modulo 2 (de bits te XOR'en). Ook het herstellen van een pakket gaat op dezelfde manier: door de bits van het parity pakket paarsgewijs te XOR'en met de bits van een pakket dat wel goed is aangkomen.

3.7.1 Werking

De werking van Reed-Solomon codes is gebaseerd op een specialistisch wiskundig gebied. Het gebied van de eindige lichamen[MACWI77].

Eindig lichaam

Reed-Solomon codes zijn gebaseerd op eindige lichamen. Een eindig lichaam (vaak afgekort tot GF: Galois Field) is een verzameling met een eindig aantal elementen. Voor deze getallenverzameling geldt:

De optelling is commutatief. Dat wil zeggen $a + b = b + a$, met $a, b \epsilon$ GF(q).

De optelling is associatief. Dat wil zeggen $(a + b) + c = a + (b + c)$, met a, b en $c \epsilon$ GF(q).

Er bestaat een nulelement, genoteerd als 0. Hiervoor geldt $a + 0 = a$, met $a \epsilon$ GF(q).

Voor elk element is er een tegengesteld element. Dat wil zeggen $a + (-a) = 0$, met $a \epsilon$ GF(q).

De vermenigvuldiging is commutatief. Dat wil zeggen $a * b = b * a$, met $a, b \epsilon$ GF(q).

De vermenigvuldiging is associatief. Dat wil zeggen $(a * b) * c = a * (b * c)$, met a, b en $c \epsilon$ GF(q).

Er bestaat een eenelement, genoteerd als 1. Hiervoor geldt $a * 1 = a$, met $a \epsilon$ GF(q).

Voor elk element is er een inverse. Dat wil zeggen $a * (a^{-1}) = 1$, met $a \epsilon$ GF(q).

De vermenigvuldiging is distributief over de optelling. Dat wil zeggen $a * (b + c) = a * b + a * c$, met a, b en $c \epsilon$ GF(q).

Voor Reed-Solomon codes is het aantal elementen van zo'n lichaam altijd een macht van een priemgetal. Een eindig lichaam met q elementen wordt genoteerd als GF(q). q is altijd een priemgetal tot een bepaalde macht. Dit wordt vaak genoteerd als GF(p^{λ}). Omdat VoIP werkt met binaire pakketten (rijen van bits) wordt er gebruik gemaakt van GF(2^m) Hierbij is m de grootte van een pakket (aantal bits per pakket).

Constructie

Een eindig lichaam wordt geconstrueerd volgens de methode van Kronecker. Hierbij wordt gebruik gemaakt van een irreducibel polynoom van graad λ. Een polynoom f(x) heet irreducibel wanneer uit f(x) = g(x) * h(x) volgt: g(x) of h(x) heeft graad 0 (is constant). Met andere woorden als f(x) niet bestaat uit het product van twee andere polynomen met een lagere graad dan f(x), maar wel groter dan 0.

Allereerst wordt voor elke x in het irreducibele polynoom een primitief element (vaak α) ingevuld. Vervolgens wordt dit gelijk gesteld aan 0. Dit levert

een vergelijking op van de vorm $\alpha^\lambda = a * \alpha^{\lambda-1} + b * \alpha^{\lambda-2} + ... + c$ met a, b en c elementen van GF(p) berekend modulo p.

Nu is een element van het eindige lichaam bekend (α^λ). De andere elementen kunnen nu berekend worden door dit element te vermenigvuldigen met α en te herschrijven zodat geen enkel coëfficient een graad hoger dan λ heeft.

Voorbeeld

De constructie van GF(8). GF(8) = GF(2^3). Een irreducibel polynoom van graad 3 is

$$f(x) = x^3 + x + 1$$

Neem α als primitief element en stel dit polynoom gelijk aan 0. Dit levert dan de volgende vergelijking op:

$$\alpha^3 + \alpha + 1 = 0$$

$$\alpha^3 = \alpha + 1$$

Nu bestaat GF(8) uit de volgende elementen:

$$\alpha^1 = \alpha$$

$$\alpha^2 = \alpha^2$$

$$\alpha^3 = \alpha + 1$$

$$\alpha^4 = \alpha^2 + \alpha$$

$$\alpha^5 = \alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1$$

$$\alpha^6 = \alpha^3 + \alpha^2 + \alpha = \alpha^2 + 1$$

$$\alpha^7 = \alpha^3 + \alpha = 1$$

Generalized Reed-Solomon

Er bestaan verschillende soorten Reed-Solomon codes. Een veel gebruikte en redelijk eenvoudige Reed-Solomon code is de Generalized Reed-Solomon code (afgekort tot GRS). Dat is de code die ik hier ga bepreken.

Een Generalized Reed-Solomon code bestaat uit een $\alpha = (\alpha_1, ..., \alpha_N)$ waarbij elke α_i een apart element van GF(q^m) is. Verder is er een $v = (v_1, ..., v_N)$ waarbij elke v_i een element van GF(q^m) ongelijk aan nul is. De code is dan:

$$GRS_K(\alpha, v) = (v_1 F(\alpha_1), v_2 F(\alpha_2), ..., v_N F(\alpha_N))$$

Hierbij is F(z) een functie van alle polynomen van graad < K met coëfficienten uit GF(q^m). Oftewel F(z) ϵ F[x]$_K$.

Encoder

Bij het coderen van gegevens met Reed-Solomon wordte gebruik gemaakt van een vector $u = (u_0, u_1, ..., u_{K-1})$. Elke u_i is een element van GF(q). Deze u_i's zijn de gegevens die gecodeerd gaan worden.

Met deze vector wordt vervolgens een vector met het codewoord berekend. De elementen van deze vector worden berekend met de formule

$$u(z) = \sum_{i=0}^{K-1} u_i z^i$$

De vector met het codewoord wordt dan gevormd door

$$c = (v_1 u(1), v_2 u(\alpha), ..., v_N u(\alpha^{N-1}))$$

Hierbij is elke v_i een element uit GF(q) ongelijk aan nul.

Decoder

Het codewoord $c = (v_1 u(1), v_2 u(\alpha), ..., v_N u(\alpha^{N-1}))$ wordt verzonden via het netwerk. Hierbij kan een fout vector $e = (e_0, ..., e_{N-1})$ optreden. De ontvanger ontvangt dan $y = (y_0, ..., y_{N-1})$. Dit moet door de ontvanger gedecodeerd worden.

$$y_0 = e_0 + v1 * (u_0 + u_1 + .. + u_{K-1})$$
$$y_1 = e_1 + v2 * (u_0 + \alpha u_1 + ... + \alpha^{K-1} u_{K-1})$$
$$...$$
$$y_{N-1} = e_{N-1} + v_N * (u_0 + \alpha^{N-1} u_1 + ... + \alpha^{(K-1)(N-1)} u_{K-1})$$

Er zijn twee mogelijkheden: het codewoord komt zonder fouten aan of het codewoord komt met fouten aan.

Als het codewoord zonder fouten aankomt bij de ontvanger, als $e = 0$, kan de ontvanger $u = (u_0, ..., u_N)$ berekenen door K vergelijkingen op te lossen. Hiervoor kan gebruik gemaakt worden van interpolatie. Als er geen fouten zijn zal het oplossen van elke K vergelijkingen dezelfde u opleveren. Elke oplossing voor zo'n u is een stem. Als er geen fouten zijn zijn er $\binom{N}{K}$ mogelijke verzamelingen van K vergelijkingen in de N vergelijkingen en dus $\binom{N}{K}$ stemmen voor de juiste u.

Als er wel fouten in het codewoord zitten zullen bepaalde verzamelingen van K vergelijkingen een foute u berekenen. Andere K vergelijkingen zullen wel een juiste u opleveren. De foute u's zullen echter niet veel stemmen krijgen. De u met de meeste stemmen zal gekozen worden en dat zal de correcte u zijn.

Deze decodeer methode is de originele methode bedacht door Reed en Solomon. Deze methode is bij grote waarden van $\binom{N}{K}$ echter erg onpraktisch. Daarom zijn er ingewikkeldere en praktischere methoden bedacht.

Voordelen van Reed-Solomon

Het grootste voordeel van Reed-Solomon codes is dat het burst errors kan repareren. Dit komt omdat Reed-Solomon werkt met complete symolen. Bij het herstellen van een symbool maakt het niet uit of één bit per symbool fout is of alle bits. Een burst error is de situatie wanneer er heel veel fouten achter elkaar optreden. Zeker bij het versturen van gegevens via internet is dat een groot voordeel. Op internet treden namelijk voornamelijk burst errors op en slechts af en toe een enkele error.
Een ander voordeel van Reed-Solomon codes is dat het een vrij simpele code is. Het coderen en decoderen kost dus weinig tijd. Ook is het eenvoudig in hardware te bouwen[CLARK02].

Hoofdstuk 4

Conclusie

Het was niet makkelijk om informatie te vinden over het gebruik van error correction in VoIP. Bedrijven geven hier weinig informatie over vrij. Met name over nieuwe technieken, zoals Turbo codes, maar ook over oudere technieken wordt niet veel informatie gegeven.

Desondanks is het toch gelukt om de deelvragen en onderzoeksvraag te beantwoorden.

4.1 Deelvragen

Met welke protocollen werkt VoIP?
VoIP werkt met de protocollen H.323, SIP, RTP en RTCP. Het H.323- en het SIP protocol wordt gebruikt om een gesprek op te zetten en te beëindigen. Het RTP protocol wordt gebruikt om de daadwerkelijke gegevens (de spraak) te versturen. Het RTCP protocol wordt gebruikt om extra informatie over de communicatie op te leveren.
Waar in deze protocollen wordt gebruik gemaakt van error correctie?
Alleen het RTP protocol wordt gebruikt om de spraakgegevens te versturen. Daarom wordt alleen in dit protocol error correctie toegepast. Dit wordt gedaan door een FEC pakket te maken met de FEC gegevens hierin. Dit pakket wordt vervolgens als payload van een RTP pakket gebruikt en verstuurd.
Welke error-correcting codes worden gebruikt?
Bij VoIP worden een drietal error-correcting codes toegepast:

Parity codes

Reed-Solomon codes

Turbo codes

Hoe werken deze error-correcting codes?
Parity codes zijn erg simpel. Hierbij worden twee of meer pakketten bitsgewijs ge-XORed. Dit levert een nieuw, parity, pakket op. Dit pakket wordt samen

met de pakketten waaruit het is gemaakt verstuurd. Als een van de pakketten verloren raakt kan het parity pakket en de overige pakketten gebruikt worden om dat pakket, weer door middel van XOR, opnieuw te berekenen.

Reed-Solomon codes werken door een polynoom te construeren uit de data. Deze polynoom bestaat uit elementen van een eindig lichaam. Deze polynoom wordt vervolgens op veel verschillende punten bekeken (gesampled). Omdat de polynoom op meer punten dan nodig wordt gesampled kunnen een paar punten gecorrigeerd worden als er genoeg punten correct ontvangen zijn.

Turbo codes werken door met convolutional codes de gegevens te coderen. Deze werken door bepaalde bits uit een stroom van bits te XOR'en. Voor het decoderen van deze codes wordt gebruik gemaakt van iteratieve decodering. Dat wil zeggen dat de codes meerdere keren gedecodeerd worden. Voor elke bit wordt door de decoders een waarschijnlijkheiswaarde gegenereerd. De twee decoders decoderen vervolgens de gegevens en vergelijking deze decodering. Als ze verschillen wisselen ze de waarschijnlijkheidswaarden uit en doen ze het opnieuw.

Waarom worden juist deze codes gebruikt?
Parity codes worden gebruikt, omdat ze zo simpel zijn. Hierdoor kunnen ze met weinig overhead een pakket herstellen.

Reed-Solomon codes worden gebruikt, omdat ze relatief simpel zijn en omdat ze heel geschikt zijn om burst errors te herstellen. Op internet zijn burst errors heel gewoon, dus is het erg handig als de code hier goed mee om kan gaan.

Turbo codes worden gebruikt, omdat ze veel fouten kunnen herstellen. Hoe meer fouten een code kan herstellen, hoe minder herstelpakketten nodig zijn.

4.2 Onderzoeksvraag

Dit beantwoord ten slotte de hoofdvraag. Bij VoIP zijn er drie mogelijke error-correcting codes: Parity codes, Reed-Solomon codes en Turbo codes. Deze codes worden in een RTP pakket gestopt en net als de data verstuurd. De verschillende codes hebben allemaal andere voor- en nadelen, maar ze zijn allemaal geschikt voor het gebruik in VoIP door de lage overhead of door de hoge foutcorrectie.

Hoofdstuk 5

Referenties

5.1 Literatuur

[LIESE00] Jori Liesenborgs, *Voice over IP in networked virtual environments*, 2000, http://research.edm.uhasselt.be/jori/thesis/onlinethesis/contents.html

[DATAB99] *A Primer on the H.323 Series Standard.* DataBeam Corporation, 1999
http://www.packetizer.com/voip/h323/papers/primer/

[LIU98] Chunlei Liu, *Multimedia Over IP: RSVP, RTP, RTCP, RTSP*, 1998
http://www.netlab.ohio-state.edu/ jain/cis788-97/ip_multimedia/index.htm

[PACKE98] *H.323 versus SIP: A Comparison*, Packetizer Inc., 1998-2007
http://www.packetizer.com/voip/h323_vs_sip/

[BURR01] A.Burr, *Turbo codes the ultimate error correcting codes?*, ELECTRONICS & COMMUNICATION ENGINEERING JOURNAL, 2001

[SAYIR05] Jossy Sayir & Gottfried Lechner, *Theory and Design of Turbo and Related Codes, Introduction and Historical Notes*, 2005,
http://userver.ftw.at/ jossy/turbo/intro.pdf

[TERVO05] Richard Tervo, *Error Correction and the Hamming Code*, EE4253 Digital Communications, 2005,
http://www.ece.unb.ca/tervo/ee4253/hamming.htm

[RAAPH03] Sebastian Raaphorst, *Reed-Muller Codes*, Carleton University, 2003
http://www.sebandthecity.com/academic/mat5127paper.pdf

[CIPRA93] Barry A. Cipra, *The Ubiquitous Reed-Solomon Codes*, SIAM News,

Volume 26-1, 1993,
http://www.eccpage.com/reed_solomon_codes.html

[VOIPT06] *VoIP Troubleshooter — Packet Loss Concealment*, VoIP Troubleshooter LLC, 2006,
http://www.voiptroubleshooter.com/problems/plc.html

[HUANG97] Fu-hua Huang, *Evaluation of Soft Output Decoding for Turbo Codes*, 1997,
http://scholar.lib.vt.edu/theses/available/etd-71897-15815/

[MEHTA05] Shveni P Mehta, *Comparative Study of Techniques to minimize packet loss in VoIP*, 21st Computer Science Seminar, 2005

[BARBU99] S. Adrian Barbulescu en Steven S. Pietrobon, *TURBO CODES: a tutorial on a new class of powerful error correcting coding schemes*, Part II: Decoder Design and Performance, Journal of Electrical and Electronics Engineering, 1998

[VALEN98] Matthew C. Valenti, *Turbo Codes and Iterative Processing*, Proc. IEEE New Zealand Wireless Communications Symposium, 1998

[HANLE02] Stan Hanley, *Reed-Solomon Codes and CD Encoding*, 2002,
http://web.usna.navy.mil/ wdj/reed-sol.htm

[MCLEA06] Terry McLean, *Handling CD Read Errors*, 2006,
http://www.e-articles.info/e/a/title/Handling-CD-Read-Errors/

[HANDL99] M. Handley, H. Schulzrinne, E. Schooler en J. Rosenberg, *SIP: Session Initiation Protocol*, RFC 2543, 1999

[ROSEN99] J.Rosenberg and H.Schulzrinne, *An RTP Payload Format for Reed Solomon Codes*, RFC 2733, 1999,

[CHEN03] Xiuzhong Chen, Chunfeng Wang, Dong Xuan, Zhongcheng Li, Yinghua Min en Wei Zhao, *Survey on QoS Management of VoIP*, Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing, 2003

[ALTMA01] Eitan Altman, Chadi Barakat en Victor Manuel Ramos, *Queueing analysis of simple FEC schemes for IP Telephony*, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, 2001

[PERKI98] C. Perkins en O. Hodson, *Options for Repair of Streaming Media*, RFC 2354, 1998

[RAAKE06] Alexander Raake, *Speech Quality of VoIP; Assessment and Prediction*, Wiley, 2006

[CLARK02] C.K.P. Clarke, *R&D White Paper, Reed-Solomon error correction*, Research & Development BRITISH BROADCASTING CORPORATION, 2002

[MACWI77] F.J. MacWilliams and N.J.A. SLoane, *The Theory of Error-Correcting Codes*, North-Holland Publishing Co, 1977

5.2 Afbeeldingen

[COMER04] Douglas E. Comer, *Computer Networks and Internets*, Prentice Hall, 2004