AALBORG UNIVERSITET

Det Teknisk-Naturvidenskabelige Basisår
Datalogi og Software
Strandvejen 12-14
Telefon 96 35 97 31
Fax 98 13 63 93
http://tnb.aau.dk

Titel:

Komprimering af kort tekstbesked

Tema:

Fra eksisterende software til modeller

Projektperiode:
P1, 2012

Projektgruppe:
B228

Deltagere:
Anders Trier Olesen
Katrine Sofie Tjell
Jannek Alexander Westerhof Bossen
Kevin Brämer
Frederik Meyer Bønneland
Ólavur Debes Joensen
Rikke Holmberg

Vejledere:
Michael Skøtt Madsen
Amanda Hill

Synopsis:

Denne opgave omhandler problemstillingen omkring det begrænsede antal af tegn man må bruge i en SMS besked, eller andet medie hvor der er en begrænsning på antal af tegn man må bruge pr. besked. Til projektet skal der også udarbejdes et produkt som skal kunne gøre det muligt at komprimere en besked, og derved sende en besked med flere tegn end den er begrænset til. Projektet også vil komme omkring forskellige relaterede områder som for eksempel Unicode, tegnsæt og komprimeringsalgoritmer.

Oplagstal: ??

Sidetal: 31

Bilagsantal og –art: ??

Afsluttet den ??

Rapportens indhold er frit tilgængeligt, men offentliggørelse (med kildeangivelse) må kun ske efter aftale med forfatterne.

P1-Rapport

B228

7. december 2012

Indhold

Kapitel 1

Indledning

I 1992 blev den første SMS afsendt [21]. Dengang kunne denne type tekstmeddelse maksimalt rumme 160 tegn, hvilket er nøjagtig lige så mange tegn, som en SMS kan indeholde i dag. Det var den tyske enginør Friedhelm Hildebrand, der i 1985 så potentialet i muligheden for at sende korte tekst beskeder fra mobiltelefon til mobiltelefon. Han undersøgte hvor mange tegn der normalt blev brugt når man skrev et postkort, ligesom at han selv skrev forskellige beskeder, som han forstillede sig, at folk ville skrive til andre via deres mobiltelefon. Hverken antallet af tegn på postkortene, eller antallet af tegn i de korte beskeder Hillebrand selv skrev oversteg 160 tegn. Derfor blev grænsen for hvor mange tegn en SMS kan indeholde sat ved 160. Gennem tiden har meget teknologi ændret sig, så måske burde man i dag have muligheden for at sende flere end 160 tegn i en SMS. [26]

Datakomprimering

Datakomprimering handler om at gøre en datamængde mindre. Der kan både være tale om filer, billeder, film osv.. Man ønsker ofte at data skal fylde så lidt som muligt, derfor er komprimering et meget centralt emne indenfor datalogi. Når man eksempelvis taler om hukommelseslager, computernetværk, herunder især internettet, er det meget relevant at data fylder så lidt som muligt.

Man kan komprimere enkelte filer såvel som hele samlinger af filer. Mange siger ofte at filer pakkes, når man komprimerer. Der findes allerede flere komprimeringsprogrammer; zip, jpeg, WinZip, SMS ZIP, SMS ZIPPER, for bare at nævne nogle få.

Der findes selvfølgelige mange måder at komprimere på og lige så mange forskellige komprimeringsalgoritmer. Disse algoritmer kan deles op i to kategorier, tabsfri og ikke-tabsfri. Det man ofte udnytter når man komprimere tabsfrit, er at de fleste mængder af data indeholder den samme information flere gange. Eksempelvis indeholder en tekstfil de samme ord gentagne gange. Derfor kan man i stedet for den gentagne information skrive informationer om hvor mange gange den bestemte information er blevet gentaget. På den måde kan dataene genskabes

uden tab. Denne kategori benyttes især til tekstfiler, hvor det er særlig vigtigt at filen kan genskabes 100 procent. Ikke-tabsfri kompression kan eksempelvis benyttes til billedkomprimering, hvor det kan gå an at ikke hver pixel genskabes 100 procent.

1.1 Initierende Problem

Der er forskellige tjenester der giver mulighed for at skrive korte tekstbeskeder. Fælles for dem alle er at det kun er tilladt at skrive et begrænset antal tegn pr. besked, hvis denne grænse overskrides deles beskeden i to. Problemet er, at hvis der eksempelvis er tal om en SMS-besked, kommer man til at betale dobbelt SMS-takst, hvis beskeden bliver delt i to.

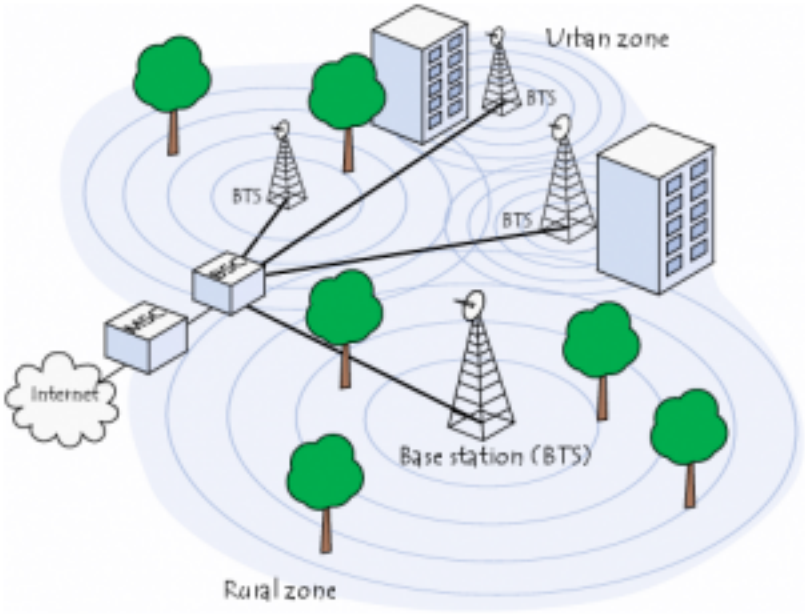
Kapitel 2

Analyse

2.1 GSM

I 1982 begyndte man at arbejde på et "anden generations"(2G) system indenfor mobiltelefoni og kaldte det for "Groupe Spécial Mobile"(GSM). Inden da havde telekommunikation foregået analogt og sådan blev det ved indtil 1991, hvor denne digitale form for mobiltelefoni, GSM, blev sat i gang. Systemet blev hurtigt internationalt udbredt og man besluttede sig for at omdøbe det til "Global System for Mobile communications". Det er dog aldrig blevet til et globalt system idet man i Japan og USA bruger andre systemer [35].

GSM netværket består af forskellige celler, som hver har en basestation, der kan modtage og sende signaler, se figur 2.1. Basestationer dækker, med deres individuelle radius, hver især et geografisk område, altså en celle. Jo mindre radius en basestation har, jo større er dens tilgængelige båndbredde. Stationer som dækker byområder, kan derfor have en radius på helt ned til få hundrede metre mens stationer, som står længere ude på landet kan dække en radius på op til 30 kilometer.[15]



Figur 2.1: Celler, som hver dækker deres eget geografiske område [15]

Når man bruger sin mobiltelefon vil den, gennem luften, skabe kontakt til den nærmeste basestation, se figur 2.2. Hvis man har sendt en SMS-besked, vil denne modtages af basestationen, som sender den videre til SMS-centeret. SMS-centeret er herefter ansvarlig for at viderebringe SMS'en til modtageren. Hvis SMS-centeret ikke kan få forbindelse til modtagertelefonen (hvis denne er slukket eller udenfor signal), gemmes beskeden i centeret og sendes når der er oprettet forbindelse til modtageren. På figuren er der indtegnet et link mellem mobiltelefon og SMS-center, disse kaldes "Mobile switching centre" og er netværksenheder, som tager sig af den trådløse kommunikation. [14]

Figur 2.2: SMS'ens vej fra afsender til modtager [14]

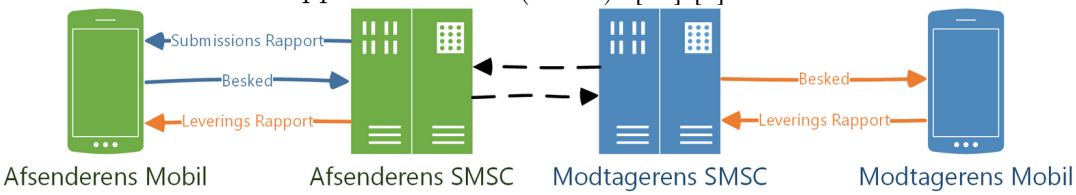
I GSM er der indbygget to muligheder for at sende en flere af SMS-beskeder sammen. Den ene mulighed er SMS-sammenkædning, hvilket vil sige at en række SMS-beskeder bliver kædet sammen og sendes hver for sig, men hos modtageren bliver de sat sammen i samme rækkefølge og læses igen som én besked. Den anden mulighed er komprimering af tekst med huffman coding.[active] Hvis denne mulighed udnyttes kan man sende optil 80 tegn mere pr. SMS besked. Dette kræver dog at telefonen fra fabrikkens side har sprogspecifik kompressions parametre. Dette kræver plads og langt de fleste mobiltelefonproducenter vælger, ifølge en artikel fra University of Teknology Sydney, skrevet af Kuross Amri og Tom Ceglearek, at bruge denne plads på spil og ringetoner [18].

2.2 SMS Teknologi

Langt de fleste mennesker beskæftiger sig dagligt med SMS'er - dog uden at vide hvad der i virkeligheden sker når der sendes en SMS. Selv når en mobiltelefon ikke er i brug, sender og modtager den små datapakker til/fra mobilcentralen. Disse data hjælper blandt andet med at lokalisere de signaltårne mobilen er tilkoblet til. Dette kan især blive nyttigt, idet at mobilcentralen ved hvilke tårne der skal benyttes da telefonen skal modtage opkald eller data, som f.eks. SMS'er.

Når en SMS besked bliver sendt fra en mobil bliver den i første omgang sendt til mobilcentralen via signaltårne. Når mobilcentralen modtager beskeden bliver den overført til et SMS-center (SMSC). SMS-centeret tager sig af at sende beskeden til den rette modtager, ved at overføre den til den ønskede modtagers SMS-center. Når beskeden når frem til den pågældendes center, bliver den, hvis

det er muligt, overført til mobilcentralen der sender beskeden til modtageren. Hvis modtageren af en eller anden årsag ikke er at finde på et mobilt netværk bliver beskeden opbevaret på SMS-centeret i op til flere dage, og vil først blive afsendt når der er mulighed for det. SMS-centeret kan endvidere sende en bekræftelse til afsenderen, når beskeden bliver leveret til modtageren. Alt dette er muligt via Signaling System no. 7 - som er en protokol suite, der indeholder forskellige protokoller. Protokollerne benyttet af SMS systemer befinder sig i mere specifikt i SS7 suitens Mobile Application Part (MAP). [12] [8]



Signalerings systemerne i MAP er efter design begrænset til visse størrelser af data. Opfinderne af SMS systemet prøvede med forskellige beskeder, hvorefter man fandt ud af at langt de fleste var under 160 tegn. Man mente derfor at 160 tegn var rigeligt til at rumme de fleste beskeder, og en SMS beskeds maksimale størrelse blev derfor defineret til 160 tegn. Efter introduktion af udvidede tegnsæt er definitionen præciseret til 140 bytes, eller 1120 bits. [8] [16]

Da begrænsningen er defineret i bits, er beskedens maksimale længde afhængig af det anvendte tegnsæt. Det mest basale er det grundlæggende 7-bit GSM alfabet. Dette alfabet benytter 7-bits til at symbolisere tegn, hvilket udgør 128 forskellige muligheder. 7-bit GSM alfabetet begrænser derfor en SMS beskeds længde til $1120/7$ bits $= 160$ tegn. Når der er brug for mere avancerede specialtegn, bruger SMS systemer UCS-2 tegnsættet. Dette tegnsæt benytter 2 okteter - altså 16 bits - til repræsentation af ét tegn. Ved brug af dette tegnsæt mindskes den maksimale længde derfor ned til $1120/16 = 70$ tegn. [20]

Enhver SMS-besked indeholder også en header[20], som der er afsat plads til udover de 140 okteter. En SMS-header indeholder typisk data som f.eks. afsenderens telefonnummer, længden af beskeden, benyttet tegnsæt og lignende. Hvis en SMS besked bliver længere end grænsen ved det benyttede tegnsæt bliver beskeden delt op i flere beskeder. Når en besked bliver delt op, skrives der information til fletning af beskeden i headeren - og da der ikke er afsat plads til ekstra header information, bliver der brugt 6 okteter af de oprindelige 140 i beskeden. Dette begrænser længden yderligere til 153 ved 7-bit encoding og 67 ved 16-bit encoding.

2.3 Tegnsæt

For at kunne komprimere en besked er det vigtigt at kende til teknologien bag sms'er.

Den teknologi som anvendes i moderne telefoner hedder GSM (Global System

for Mobile Communications), som er en 2G standard. Denne teknologi gør det muligt at benytte sig af sms'er.[19]

For at et computersystem skal have muligheden for at kunne printe tegn til skærmen, er det nødvendig at repræsentere disse tegn med hver sit tal. Disse tal har man bestemt i en standard, som betyder at alle skal benytte de samme tal, for de samme tegn, og derved gøre det lettere for programmørerne af softwaren der benytter disse tegn. Den mest brugte standard indenfor tegnsæt, som dette kaldes, er Unicode.[31]

I mobiler der gør brug af GSM, kan der til sms'er, benyttes et tegnsæt kaldt GSM 03.38. Dette tegnsæt kan kodes i en række alfabeter, hvor standard alfabetet GSM 7 bit er et krav ved skabelse af mobiltelefoner.[1]
Se skema af GSM 7 bit alfabetet i figur4.1
I dette skema symboliserer den vandrette linje, over den stiplede linje, det foerste ciffer, mens den lodrette kolonne er andet ciffer. F.eks hvis man skal printe tegnet A, har den en talvaerdi paa 42. Ved b1 til b7 vises ogsaa den binaere talvaerdi.

2.4 Komprimeringsalgoritmer

2.4.1 Entropikodning

Entropikodning er en lossless/tabsfri datakomprimeringsmetode. Tabsfri, betyder at der ikke går nogen information tabt, ved at komprimere datamængden. Modsat har vi lossy/tabsgivende komprimering, som fx MP3, og JPEG. Entropopikodning går ud på, at få en given datamænde til at benytte et minimalt antal bit. Dette kan opnås ved at kigge på hyppigheden af de forskellige tegn i datamængden der skal komprimeres, og give de oftest fremkommende symboler få bits, og de mere sjældne symboler flere bits. Formålet er, at få det gennemsnitlige antal bits pr. symbol(middelkodelængden) ned. Den teoretiske nedre grænse for middelkodelængden kaldes datamængdens entropi. [10] Der findes flere forskellige entropikodningsmetoder, og et par eksempler er "Huffman kodning"og "arithmetic coding".

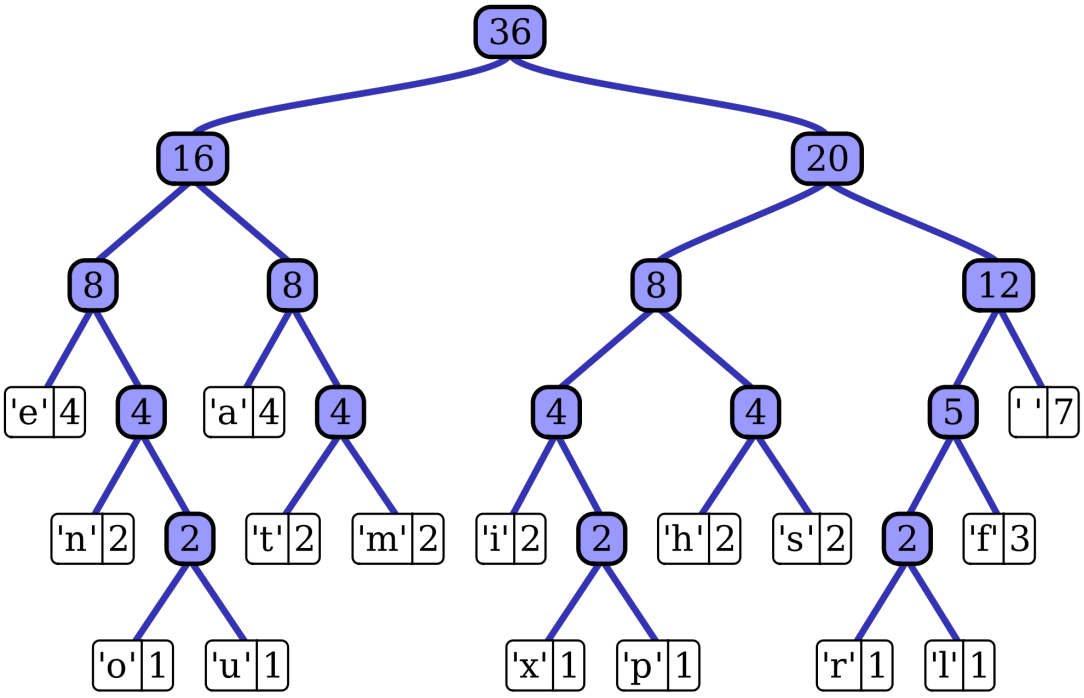
2.4.2 Huffman Coding

Komprimeringsalgoritmen "Huffman coding", er udviklet af David A. Huffman. Huffman udviklede algoritmen mens han var Ph.D studerende på MIT. I 1952 udgav han dokumentet"A Method for the Construction of Minimum-Redundancy Codes"[13]. Her beskrev Huffman hvordan hans komprimerings algoritme fungerede. Hvad han havde udviklet, var en 'lossless' (tabsfri) komprimerings metode, hvilket betyder, at der ikke vil være noget tab af information ved at komprimere. Komprimeringsmetoden er beregnet til binære systemer, og formålet med algoritmen er at få en given datamænde til at benytte et minimalt antal bit.

For så at kunne få de orginale data tilbage fra den komprimerede form, kræver det selvfølgelig, at man har en form for ordbog, der beskriver hvilke tegn, der hører sammen med hvilke bits.

For at Huffman coding effektivt kan fungere, skal algoritmen have adgang til hele datamængden, for at kunne analysere hyppigheden af forskellige tegn. Dette betyder at algoritmen skal løbe i gennem datamængden to gange. Første gang, for at indsamle statistik, og anden gang for så at foretage den reelle komprimering. En eksempel på en algoritme der ikke har den ulempe er komprimeringsmetoden "Lempel-Ziv-Welch".

Figur 2.3 viser et Huffman træ, genereret ud fra sætningen "this is an example of a huffman tree"[33]. Tabel 2.1 viser frekvensen, og en kort kode i bits for de forskellige tegn. Nu fylder sætningen kun 135 bits, mod de 288 bits sætningen originalt fyldte - en besparelse på ca 53%. Denne kræver selvfølgelig, at de binære koder allerede er kendes af algoritmen der skal dekomprimere datamængden.



Figur 2.3: Huffman træ

2.4.3 PPM Komprimering

PPM blev udviklet af John Cleary og Ian Witten. De beskrev metoden i en artikel de udgav i 1984; "Data Compression Using Adaptive Coding and Partial String Matching"[5].

Tegn	Hyppighed	Binær kode
space	7	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010
s	2	1011
t	2	0110
l	1	11001
o	1	00110
p	1	10011
r	1	11000
u	1	00111
x	1	10010

Tabel 2.1: Hyppighed og kode af forskellige tegn

PPM er en tabsfri komprimeringsmetode, der er blandt de bedste til at komprimere tekst. PPM står for "Prediction by partial matching"(Forudsigelse ved delvis matching), hvilket også fortæller lidt om hvordan komprimeringsmetoden fungerer. PPM forsøger at forudsige, hvad det næste tegn i datamængden vil være, ved at kigge på den ikke komprimerede data, og kigge efter om der findes en lignende sammensætning af tegn, og hvor ofte forskellige tegn forekommer.

Algoritme 1 viser en simpel implementering i pseudokode, der bearbejder tegn

for tegn, og forsøger at lave nogle sammenhænge, ud fra om noget går igen. [3]

```
while ikke sidste tegn do
    læsSymbol();
    forkort kontekst;
    while sammenhæng ikke fundet og sammenhæng længde ikke er lig -1
    do
        output(sekvens);
        forkort sammenhæng;
    end
    output(tegn);
    while kontekst længde ikke er -1 do
        tæl tegntæller op;
        forkort kontekst;
    end
end
```

Algorithm 1: Pseudokode af PPM komprimering [3]

2.5 Problemdokumentation

Når det kommer til SMS beskeder, så er der en grænse på hvor mange tegn der kan være i en enkelt besked. For tegn inkluderet i tegnsættet GSM 7-bit ligger begrænsningen på 160 tegn. Begrænsningen ændrer sig fra tegnsæt til tegnsæt. For eksempel har det kinesiske alfabet en tegnbegrænsning på 70 tegn[12]. Normalt vil en besked som fylder mere end sin tegnbegrænsning blive delt op i to separate beskeder, hvis afsenderen af beskeden ikke selv gør det, hvilket kommer til at betyde dobbelt SMS takst. Med denne begrænsning i tankerne kommer spørgsmålet: Hvor betydeligt er dette problem, og er det overhovedet værd at kigge nærmere på? Erhvervs priserne for at sende en SMS inden for Norden og Eurozonen er betydeligt billigere end hvis man sendte til eller fra et Europa land ikke inde under EU, og når man sender til eller fra lande udenfor Europa så bliver det kun dyrere og dyrere. Et internationalt firma som udnytter SMS til intern kommunikation eller andet kan ende med at bruge mange penge på deres telefonregninger. Tilbage i 2009/2010 begyndte de forskellige telefonselskaber at hæve prisen på afsendelse af beskeder til udlandet. TDC's pris, for eksempel, gik fra at være på 2,40 kr. til at koste 3,20 kr. per SMS[4]. Nedenstående tabel viser Telenors SMS takst samt minutpris for erhverv ved at sende beskeder til Danmark, men prisen er stadig den sammen den fra Danmark til udlandet[29].

	Sende/Modtage SMS
Norden	0,66 kr./sms
EU	0,66 kr./sms
Øvrige Europa	3,20 kr./sms
Verden 1	3,20 kr./sms
Verden 2	3,20 kr./sms
Skibe m. MCP-dækning	3,20 kr./sms

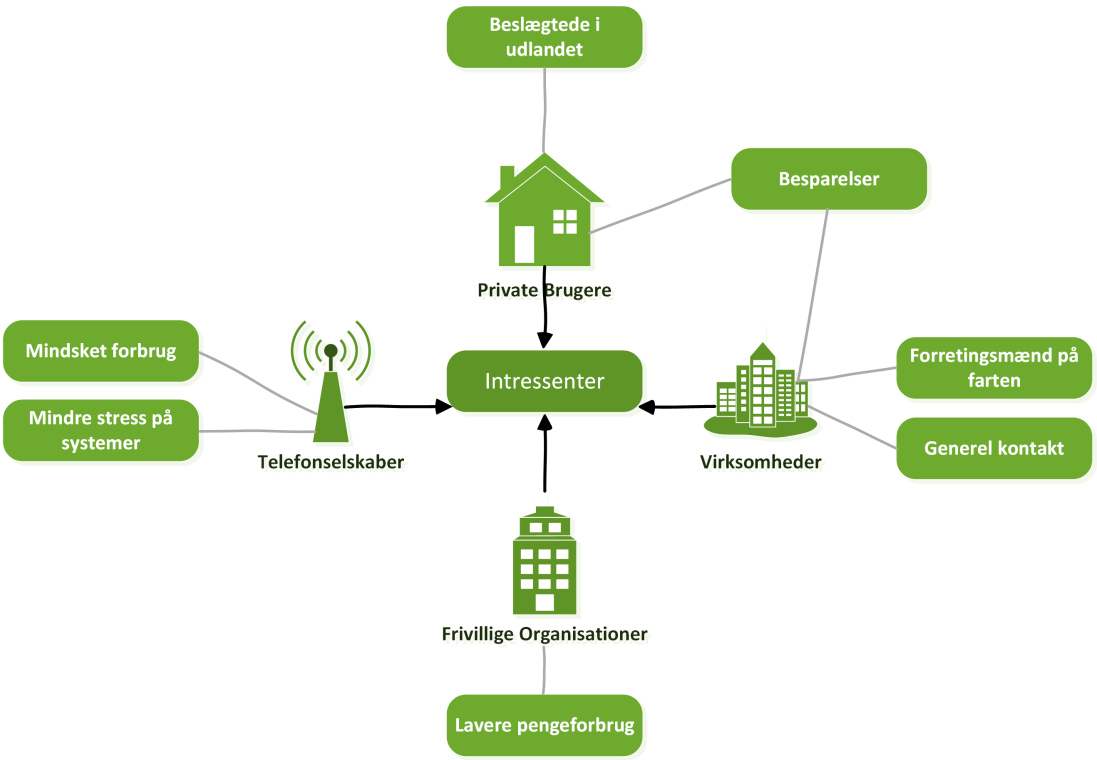
Tabel 2.2: Tabel over SMS-Priser fra Telenor [29]

Ligeledes er priserne for private hen over landegrænserne heller ikke noget at prale af. I det private strækker priserne sig fra 3's pris pr. SMS på 2,50 kr[11] til Telia's pris pr. SMS på 4,00 kr[30]. Uanset om man er privat eller erhvervsdrivende så vil man gerne være sparsomme med antallet af beskeder man sender over landegrænserne, og derudfra gøre god brug af sine 160 tegn sådan at man undgår dobbelt SMS takst ved at beskeden bliver delt, i to. Statistikkerne viser at der i 2011 blev sent omkring 12,3 milliarder SMS?er i Danmark alene, et fald fra forrige år som lå på 13 millarder, men det viser at der stadigvæk er er højt forbrug af SMS beskeder. Derudover så steg den mobile datatrafik fra 15 milliarder MB i 2010 til 26 milliarder MB i 2011[7]. En løsning som komprimer beskeder kunne også hjælpe til at dæmpe belastningen på det mobile datatrafik netværk. Derfor vil en eller anden datalogisk løsning, som gør det lettere at sende beskede, uden at man skal bekymre som om hvorvidt ens besked har mere end de begrænsede 160 tegn, være aktuelt. Sådan en løsning kan både gøre det mere bekvemt for brugeren at bruge SMS'er, og i det lange løb sparer brugeren penge.

2.6 Metodevalg

I forbindelse med projektet er der blevet tænkt over en række metoder, som gør os i stand til at finde ind til vores problems kerne, og understøtte problemet i sin helhed. Ud fra projektforslaget er det blevet fastsat at vores endelige løsning til problemet skal være en prototype af et program. Denne prototype skal være skrevet i C, men da problemstillingen omhandler SMS-beskeder, er det SMS-mediets krav, som vi skal designe vores løsning efter.

Ud fra vores problem findes der en række interessenter, som påvirkes af denne problemstilling. På den følgende brainstorm, ses hvordan disse interessenter fordeler sig, ud fra det initierende problem, og hvordan de forbindes til hinanden.



Figur 2.4: Her ses hvordan interessenterne fordeler sig.

Denne brainstorm identificerer vores primære interessenter, som vil være vores hovedmålgruppe. Man kan dele interessenterne ind i nogle grupper, henholdsvis: Private personer, internationale firmaer, frivillige organisationer og teleselskaber. Hver af disse grupper har sin egen grund til at være interesseret i vores problemstilling, og derfor kan det også betyde, at der skal forskellige løsninger til at kunne løse problemstillingen, for hver forskellig interessent.

Ud fra vores problemstilling findes der en række data som kan være anvendelig i forhold til undersøgelsen af de førnævnte interessenter.

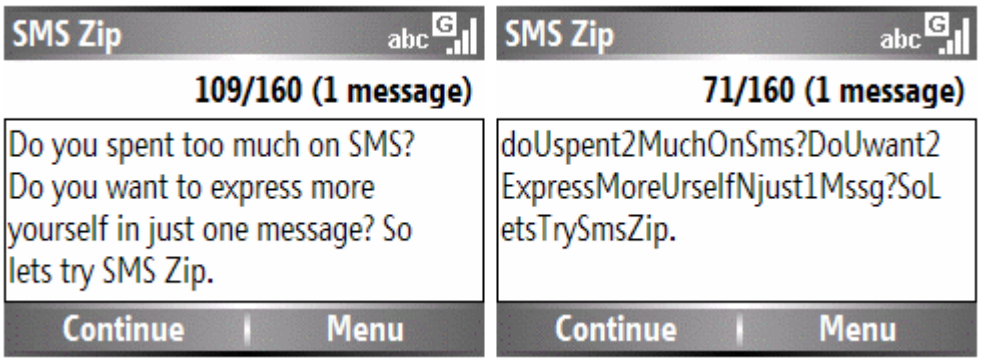
Viden om brugen af SMS'er hos de forskellige interessent grupper. Det er vigtigt at finde ud af hvordan de forskellige interessenter bruger SMS'er som et medie. Med dette menes både hvor tit det bruges, men også i hvilken forbindelse og med hvem kommunikationen foregår.

Til indsamling af data omkring disse interessenter er det nødvendigt at komme i direkte kontakt med den målgruppe vi har med at gøre. Dette betyder at vi bliver nødt til at benytte nogle metoder, som gør det muligt at indsamle eller observere målgruppens forbrug af SMS'er. Til dette vil en spørgeskema undersøgelse være velegnet, da brugen af SMS'er er data velegnet til kvantitative undersøgelser, da det er et spørgsmål om hvor mange SMS'er der sendes.

2.7 Eksisterende Løsninger

Der findes flere forskellige former for programmer der allerede helt eller delvist løser sms- begrænsnings problemet. Vi vil i det følgende afsnit tage udgangspunkt i to eksisterende programmer.

Det første program hedder SMS ZIP og virker kun til smartphones med Windows 5 og 6 som operativsystem. Det er et forældet program, eftersom det kun virker på forældede platforme. Programmet har dog stadig relevans, eftersom det er tanken bag programmet der er interessant. Dette program er af typen ikke tabsfri, da det går ind og fjerner alle unødige mellemrum i teksten, og erstatter første bogstav i følgende ord med et stort bogstav, således at teksten stadig kan læses. Ydermere er det programmeret til at kunne identificere bestemte ord og så erstatte disse med forkortelser. Programmet er indrettet således at brugeren selv skal vælge om hver enkelt besked skal komprimeres. Et eksempel på hvordan programmet vil komprimere en besked: [27]



Figur 2.5: Kort beskrivelse af billedet

Denne konkrete besked bliver altså kortet ned fra 109 til 71 tegn. En af fordelene ved dette program er at modtageren ikke behøver et tilsvarende dekomprimeringsprogram for at kunne læse beskeden. En anden fordel er at beskeder der ikke overgår en begrænsningen, ikke nødvendigvis bliver komprimeret. Denne løsning har dog en del flere ulemper end fordele. Den åbenlyse ulempe er at beskederne bliver en hel del sværere at læse, og kan være en mulig irritation for mange, når de læser beskeden. En anden klar ulempe er at der bliver brugt en del slang for at gøre ordene kortere, slang såsom tallet '2' i stedet for ordet 'to' eller 'too'. Dette kan bevirke at budskabet er sværere at tage seriøst. Yderligere er det et problem at programmet kun virker tilældre windowsphone systemer og at forkortelserne kun er beregnet til engelsktalende beskeder. Det er derfor, på baggrund af ovenstående, vores vurdering at programmet er en ufuldstændig løsning, og er derfor ikke tilstrækkelig.[27]

En anden eksisterende løsning hedder SMS ZIPPER. Det er på mange punkter et modstridende program i forhold til SMS ZIP. Først og fremmest er det

forskelligt da dette er et tabsfrit komprimerings program. Programmet virker på langt de fleste smartphones og er, i modsætning til SMS ZIP, en løsning der komprimerer beskeden hos afsenderen og derefter dekomprimerer beskeden igen hos modtageren. Dette kræver dog at både afsender og modtager har programmet installeret. Programmet starter på modtagerens telefon ligeså snart en komprimeret besked modtages, så beskeden kan læses med det samme uden besvær for læseren. Producenten lover helt op til 480 tegn pr. besked, altså 3 gange så mange tegn som en almindelig sms. Derudover fungerer programmet til flere sprog, heriblandt dansk, engelsk og tysk.

Dette program bruger en fleksibel algoritme til at komprimere beskederne. De har designet algoritmen direkte med henblik på såkaldte korte beskeder, altså beskeder omkring de 160 tegn. Endvidere bruger programmet også andre kodnings modeller, som kan være beregnet specifikt på bestemte sprog eller typer af beskeder. Dette program har til gengæld også en masse features, som ikke er relevante for komprimering, og heller ikke nødvendigvis relevante for brugeren. Dette inkluderer blandt andet muligheden for at sikre sine beskeder med password, en anden feature er selvdestruerende SMS'ere. Hvis brugeren leder efter et program der kun kan komprimere og dekomprimere, så er SMS ZIPPER ikke programmet man leder efter.[28]

I ovenstående afsnit valgt at tage to vidt forskellige programmer under luppen, for at tegne en kontrast mellem en meget simpel og en mere avanceret løsning. Vi ser at de hver især har deres fordele og ulemper, og disse vil vi tage til overvejelse i vores program.

2.8 Afgrænsning

Dette afsnit kommer omkring nogen af de valg der blev lavet for at indskrænke projektets problemfelt. For det første er der forskellige tjenester og teknologier, som giver muglihed for at sende en kort besked, men med et begrænset antal tegn pr. besked. Eksempelvis er der SMS-beskeder med en begrænsning på 160 tegn når man bruger tegn fra tegnsættet GSM 7-bit[12] og der er også internettjenester som for eksempel Twitter, som har en tegnbegrænsning på 140 tegn[32]. Twitters formål har fra starten af, været at give mulighed for at sende korte og smertefrie bidder af information over internettet, og ikke lange blogs og artikler. Denne holdning er folkene bag Twitter meget konsekvente med[17]. Derudover så er det også gratis at gøre brug af Twitter og derfor er det ikke ligeså væsentligt som SMS, som koster penge. Derfor har vi valgt ikke at arbejde med Twitter. Istedet vil projektet blive begrænset til at handle om SMS-beskeder.

Nu hvor at valget om SMS eller Twitter er på plads, så kommer spørgsmålet om hvorvidt der skal arbejdes med smartphones eller almindelige mobiltelefoner. Smartphones har den fordel at de kan implementere applikation uden alt for meget besvær, hvorimod på almindelige telefoner er det meget mere besværligt

at installere programmer. Statistikkerne viser at flere og flere begynder at få smartphones. Derudover så viser de også at brugen af hjemme computere er dallende i det at adgang til internettet også er tilgængeligt gennem smartphones, som derved går det muligt for en person at være på internettet hvor man ellers ikke ville have tilgang til en almindelig computer[9]. Dette kan betyde at flere personer bruger SMS fordi de bruger deres mobile maskiner mere. Dog kan det også betyde at flere mennesker bruger e-mail i stedet for SMS fordi de alligevel har adgang til internettet. Derfor vil projektet yderligere blive begrænset til ikke at prøve at implementere programmet på almindelige mobiltelefoner. Derudover så er det også vanskeligt at implementere en applikation skrevet i C, som er et kræv for dette projekt, på en Smartphone. Derfor vil løsningen være en prototype som kan fungerer på en computer.

Det sidste punkt er hvilke tegnsæt som løsningen skal være i stand til at komprimere og dekomprimere. Skal tegn fra det kyrilliske alfabet eller specialtegn som Æ, Ø, Å, ß være i stand til at blive komprimeret, for eksempel. Det er blevet bestemt at løsningen skal have implementeret GSM 7-bit tegnsættet, som er standard tegnsættet til SMS beskeder på mobiltelefoner og smartphone.

2.9 Problemformulering

Ud fra vores problemanalyse har vi fundet ud af, at der klart er penge at spare, hvis man fra udlandet kan sende en besked i stedet for to. Vi har ligeledes set på eksisterende løsninger som SMS-ZIP, for at undgå at lave de fejl, vi mener der er ved de programmer. Derudfra skal vores løsning være mere implementeret, således at brugeren aldrig kommer til at beskæftige sig med den komprimerede besked, så komprimering og dekomprimering sker automatisk. Vi er på baggrund af dette kommet frem til følgende problemformulering:

> *Hvordan man spare forbrugeren for dobbelt SMS-takst, ved brug af et komprimeringsprogram? Hvordan kan man undgå at programmet bliver en belastning for brugeren?*

Der vil arbejdes frem mod en prototype der kan komprimere en kort besked, for siden at dekomprimere den på en anden enhed.

2.10 Produktkrav

Til dette projekt skal der udarbejdes en løsning i form at et program, som kan komprimere en kort tekstbesked. Komprimeringen vil gøre tekstbeskeden mindre, det vil sige beskeden fylder færre bytes, og skulle gerne både gøre det hurtigere at sende beskeden fordi den er mindre, men også gøre det muligt at sende en besked over en bestemt tegn begrænsning, som f. eks. de begrænsede 160 tegn ved

brug af det latinske alfabet i en SMS. Beskeden skal derefter dekomprimeres hos modtageren, og derefter vise beskeden, som den så ud før den blev komprimeret. Denne process skal ske uden brugeren selv tager en direkte del i processen.

- Funktionelle Krav

  Skal både være i stand til at komprimere og dekomprimere automatisk.

  Programmet skal være i stand til at skelne mellem hvorvidt den pågældende besked skal komprimeres eller dekomprimeres.

  Det er ikke forventet at prototypen skal kunne køre på en mobil enhed som f. eks. en smartphone, men det er forventet at programmet kan bruges på en computer.

- Ikke Funktielle Krav

  Produktet skal afleveres sammen med den tilhørende rapport, og har en fælles deadline den 19 December 2012.

  Programmet skal skrives i programmeringssproget C.

- Løsningsmål

  Brugeren skal kunne gøre brug af programmet uden selv at tage direkte del i komprimerings-processen.

  Programmet skal køre lokalt, og ligeledes skal komprimeringen og dekomprimeringen skal også ske lokalt.

  Programmet skal implementeres og være brugbart.

Kapitel 3

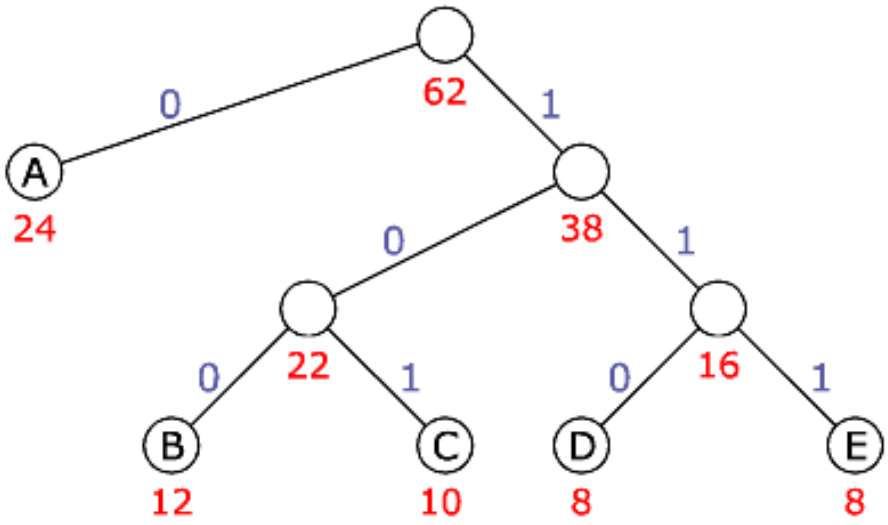
Løsning

3.1 Huffman Træer

Hvis man vil komprimere en datamængde ved brug af huffmancoding, kræver det, at der bliver lavet et huffman træ. Dette træ kan genereres på tre forskellige måder: Statisk, dynamisk eller adaptivt. Principielt ser det resulterende bit-træ ens ud for hver af metoderne. Det handler mere om hvordan træerne bliver generet. I dette afsnit vil der være en gennemgang af hvordan disse metoder virker, samt deres fordele og ulemper i forhold til SMS beskeder. Den følgende tabel samt det følgende billede viser hvordan et potentielt Huffman bit-træ kunne se ud.

Tegn	Forekomster
A	24
B	12
C	10
D	8
E	8

Tabel 3.1: Et sæt tegn og forekomster. Hentet fra binaryessence.com



Figur 3.1: Det samme sæt som før set i et Huffman træ. Hentet fra binaryessence.com

Ud fra billedet kan man se at de tegn som forekommer mindst er placeret nederst i træet, D og E, mens det tegn som forekommer mest er placeret øverst, A. Hver gang man går et niveau ned så går længden på et tegn op. A har en længde på 1 bit mens B, C, D og E har en længde på 3 bit. Tegnet som forekommer mest får altså den mindste længde. Derudover kan man også ud fra billedet se det følgende binære talsystem:

Tegn	Binær Kode	Kode Længde
A	0	1
B	100	3
C	101	3
D	110	3
E	111	3

Tabel 3.2: Binært talsystem fra et Huffman træ. Hentet fra binaryessence.com

Dette sæt af tegn og forekomster fylder i alt 496 bit for sig selv hvis man går ud fra hvert tegn fylder 1 byte, altså 8 bit. I et Huffman træ som dette billedet viser, fylder de samme tegn 138 bit som kan findes ved at lægge tegnene sammen i forhold til deres forekomster og længder. På billedet kan man også se punkter som ikke indeholder nogen tegn. Disse punkter kaldes for knuder og har en størrelse magen til summen af punkterne nedenunder den. Den øverste knude hvor træet

altid starter kaldes for roden. Enderne på træet som indeholder de egentlige tegn kaldes for blade.[24]

3.1.1 Statisk

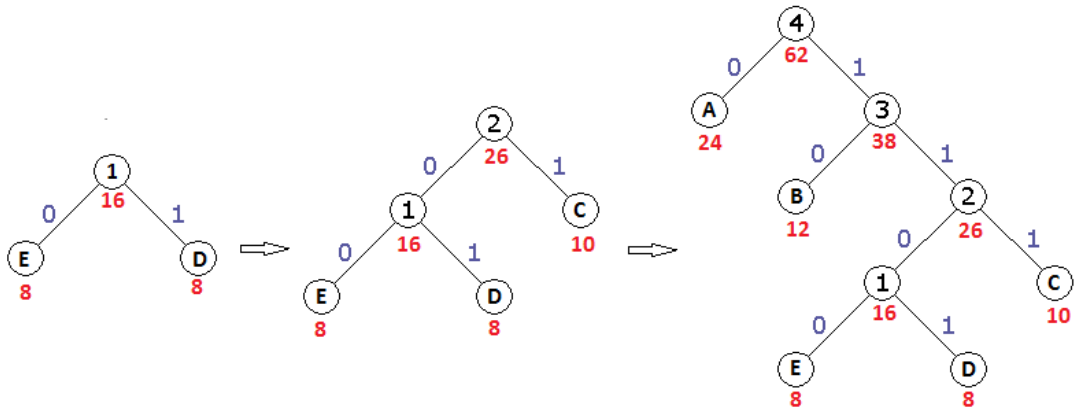
Den første metode man kan bruge til at generere et Huffman træ er den statiske metode. Et statisk Huffman træ bliver lavet ud fra formodede forekomster på i et stykke tekst. For eksempel hvis man kigger generelt på det engelske sprog så forekommer tegnet 'e' mest, og vil derfor blive placeret øverst i det statiske Huffman træ, mens tegn som 'z' og 'x' vil blive placeret nederst[2].

Et statisk Huffman virker på alle stykker tekst, især længere stykker af tekst som for eksempel artikler. Til gengæld kan den ende med at ikke gøre fuld brug af komprimeringskraften ved Huffman coding når det handler om mindre beskeder fordi det ikke altid går op med normalen for tegn forekomster i et sprog. Statiske Huffman træer virker bedre i takt med at den tekst som skal komprimeres bliver større, og bliver mere ineffektiv når teksten bliver mindre. Det er ikke sandsynligt at den komprimerede tekst vil fylde mere end hvis den ikke var komprimeret idet at Huffman komprimering af så effektivt som det er. [25]

Fordele ved et statisk træ er at den kan let give gode resultater for større stykker af tekst og behøver ikke at sende noget yderligt i beskeden udover bitmønstret. Et statisk træ er også hurtigt idet at der ikke er behov for at generer et nyt træ for hvert eneste stykke tekst, som der er behov for med de to kommende metoder. Statisk kan derfor have en fordel fordi den ikke behøver ligeså meget regnekraft på enheder som mobiltelefoner, som muligvis kan være begrænset på det område.

3.1.2 Dynamisk

Den anden metode man kan bruge kaldes for et dynamisk Huffman træ. Et dynamisk træ bliver generet ud fra det reelle data som teksten består af. Træet bliver genereret ud fra den specifikke besked og er ikke en generel liste som ved den statiske metode. Det dynamiske træ er derfor den mest optimale til Huffman coding. Den dynamiske opbygning af et træ starter med at sætte de to tegn med færrest forekomster til en knude. Derefter bliver der bygget ovenpå den knude og tager hele tiden de tegn med færrest forekomster som endnu ikke er blevet sat ind i træet. En ny knude bliver lavet som får sat det næste tegn samt den foregående knude som indeholder alle tidligere tegn og knuder på.



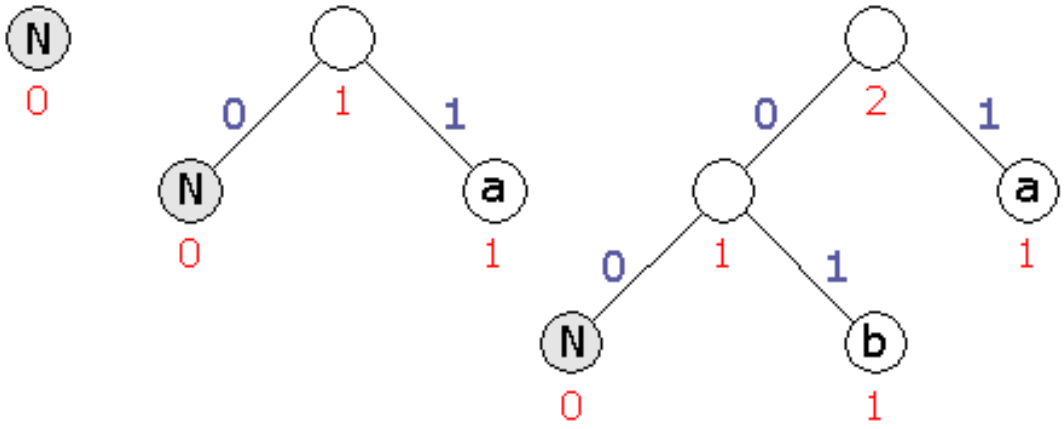
Figur 3.2: Dynamisk opbygning af et Huffman træ udfra det tidligere eksempel. Hentet fra binaryessence.com

En ulempe ved den dynamiske metode er, at for at kunne dekode den komprimeret tekst skal man bruge en kode tabel over træet. Idet at det er et træ skræddersyet til et bestemt stykke tekst, så har dem der skal dekode teksten ingen mulighed til hvordan man oversætter 0 og 1 tallene tilbage til den oprindelige tekst. Derfor er det nødvendigt med den dynamiske metode at medsende en tabel som angiver hvilke tegn har hvilke bitmønstre. Dette betyder selvfølgelig at den komprimeret tekst fylder et stykke mere og gør derfor ikke optimalt brug af komprimeringskraften ved Huffman coding. [23]

Fordelen ved et dynamisk træ i forhold til det statiske er at den mere konstant gør optimal brug af Huffman coding til at komprimere et stykke tekst. Det er selvfølgelig det man gerne vil opnå i forhold til SMS beskeder, men som sagt så er det nødvendigt at sende en tabel med så det er muligt at dekode det komprimeret stykke tekst. Dette betyder at SMS beskeden fylder mere end hvis det bare var det komprimeret stykke tekst, og arbejder derfor imod målet om at formindske størrelsen af den data der bliver sendt.

3.1.3 Adaptivt

Den tredje og sidste metode til at komprimere et stykke tekst med Huffman coding er den adaptive metode. Med den adaptive metode starter træet med at intialisere forekomsten af tegn som de kommer til 1. Når det samme tegn fremkommer igen så vil det blive lagt til tegnets fremkomst værdi, og træet vil derefter også opdatere sig selv sådan hvis et nyt tegn begynder at have flest forekomster så vil den blive placeret øverst i træet. I figur 3.3 kan man se hvordan det adaptive træ initialisere nye tegn træet støder på til en, og har altid en kontrol knude, N, hvis der skulle komme yderligere nye tegn.



Figur 3.3: Initialisering af tegn i et adaptivt træ. Hentet fra binaryessence.com

I forhold til den foregående metode så kan den adaptive metode genere et træ alt efter hvad for noget data som skal komprimeres, ligesom den dynamiske metode, men har til gengæld ikke behov for at videregive en tabel som viser hvordan man oversætter det komprimerede tekst, idet at det adaptive træ på den enhed som dekomprimere vil gøre arbejdet og opdatere sig selv med det data den dekomprimere. Dette kan også betyde at det dynamiske træ muligvis ikke behøver at sende en tabel med over hvis dekomprimeringen sker ved hjælp af det adaptive træ. Ulempen ved den adaptive metode er at det stykke tekst der bliver komprimeret eller dekomprimeret ikke indgår i den nuværende version af træet, idet at det adaptive træ først opdateres med data fra tekststykket efter det er blevet komprimeret eller dekomprimeret. Derudover så kræver den også meget regnekraft fra den enhed som skal komprimere og dekomprimere fordi den konstant skal opdateres af det data den gennemarbejder. [22]

I forhold til SMS beskeder så er den adaptive metoder sikkert bedst når det kommer til at formindske den mængde data der bliver sendt med beskeden. Ulempen er dog at træet har brug for tid til at komme op og køre ved at konstant opdatering af sit eget træ. Den adaptive metode starter derfor ud med at være langsom og ineffektivt idet at den introducere alle tegn fra bunden af og kan først effektivt komprimere efter den har arbejdet sig igennem et antal tekststykker. Kravet for regnekraft på enhederne der komprimere og dekomprimere kan også være en større ulempe på en mobiltelefon.

3.1.4 Sammenligning

3.2 Implementering

3.2.1 Brug af træer

I afsnittet huffman træer, ses der på hvilke metoder, der kan benyttes for at skabe de binære træer, der skal bruges ved dekodning af de sendte komprimerede beskeder.

Der blev nævnt to metoder. Den ene kaldt statisk, hvor der benyttes det samme binære træ for generel komprimering af tekst, mens den anden kaldt dynamisk, bliver lavet i forbindelse med hver enkelt komprimering.

Dette betyder derfor at komprimeringer hvor der bruges dynamiske træer, kræver at det binære træ, sendes sammen med den komprimerede besked. Hvorimod, ved brug af et statisk træ, er muligt at have det binære træ ved modtageren, af den komprimerede besked, på forhånd.

Ved brug af huffman komprimering af korte beskeder, som ved sms'er, er det derfor mest hensigtsmæssigt at bruge statiske træer, da et medsendt dynamisk træ, kan få beskeden til at fylde mere end originalt.

3.2.2 Tabsfri kontra ikke tabsfri

Som nævnt i indledningen er der to former for komprimering, tabsfri og ikke tabsfri metoder.

Tabsfri er, som navnet antyder, en metode hvor ingen data går tabt i komprimeringsprocessen. Den komprimerede og senere dekomprimerede data er altså eksakt magen til den oprindelige data. Alle tabsfri komprimeringsmetoder bryder filen op i mindre sektioner, og udnytter redundans. Redundans er fx når ord eller bogstaver optræder oftere end nødvendigt. Alt dette bevirker at via en algoritme kan dataen genskabes perfekt ved dekomprimeringen. En af ulemperne ved denne komprimeringsmetode er at kompressionsratioen er lavere end en ikke tabsfri metoder[34]. De tabsfri metoder bruges, som i vores tilfælde, til kompression af tekst. Ydermere bruges de til

Ikke tabsfri komprimering genskaber derimod ikke en eksakt kopi af den oprindelige fil. Ved ikke tabsfri komprimering fjernes unødvendig data, fx data der opstår flere gange i filen. Fordelen ved disse metoder er ulempen ved de tabsfri metoder, altså at kompressionsratioen er højere end de tabsfri. Ikke tabsfri metoder bruges som oftest til lydfiler, billed- og videofiler[6].

Kapitel 4

Konklusion

Litteratur

[1] 3GPP. TS 23.038. File;, http://www.3gpp.org/ftp/specs/archive/23_series/23.038/23038-a00.zip, 2011. [Online; tilgået 27-October-2012].

[2] Algoritmy.net. Letter frequency. Algoritmy.net, http://en.algoritmy.net/article/40379/Letter-frequency-English, 2008. [Online; tilgået 04-December-2012].

[3] Ing. Miroslav Balík, PhD.Ing. Marek Hanusdoc., Ing. Jan Holub, and PhD.Ing. Michal Paulícek. PPMC. Stringology.org, http://www.stringology.org/DataCompression/ppmc/index_en.html, 2006. [Online; tilgået 19-November-2012].

[4] Thomas Breinstrup. TDC hæver SMS-pris med 33 procent. Business.dk, http://www.business.dk/digital/tdc-haever-sms-pris-med-33-procent, 2012. [Online; tilgået 28-October-2012].

[5] John G. Cleary, Ian, and Ian H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 31:396–402, 1984.

[6] Maximum Compression. Lossless data compression. MaximumCompression.com,http://www.maximumcompression.com/lossless_vs_lossy.php, 2012. [Online; tilgået 06-December-2012].

[7] Danmarks Statistik/Statistikbanken.dk. Serviceerhverv og Informationssamfundet. http://www.dst.dk/pukora/epub/upload/16252/12ser.pdf, 2012. [Online; tilgået 18-November-2012].

[8] Lee Dryburgh and Jeff Hewett. *Signaling System No. 7*. Networking Technology. Cisco Press, 1 edition, 2004. eBook; http://www.informit.com/library/library.aspx?b=Signaling_System_No_7.

[9] Google. Think with Google - Our Mobile Planet. http://services.google.com/fh/files/blogs/our_mobile_planet_denmark_en.pdf, 2012. [Online; tilgået 19-November-2012].

[10] Henrik Graversen and Casper Thomsen. Entropikodning. Version2.dk, http://www.version2.dk/leksikon/Entropikodning, 2012. [Online; tilgået 27-October-2012].

[11] Hi3G Denmark. Ring fra Danmark. 3.dk, http://www.3.dk/Privat/Kundeservice/Hjaelp_til_mobilen/Udland/Ring-fra-Danmark/, 2012. [Online; tilgået 28-October-2012].

[12] Jennifer Hord. How SMS Works. HowStuffWorks.com, http://computer.howstuffworks.com/e-mail-messaging/sms.htm, 2009. [Online; tilgået 1-November-2012].

[13] David Albert Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(9), September 1952.

[14] Informationsteknologi. SMS - Netværksarkitektur. iftek.dk,http://iftek.dk/sms-netvaerksarkitektur. [Online; tilgået 02-December-2012].

[15] Mubashar Islam. How GSM Network Works. techviral.com,http://www.techviral.com/networks/gsm-network-works/, 2012. [Online; tilgået 04-December-2012].

[16] Mark Milian. Why text messages are limited to 160 characters. *Los Angeles Times*, 2009. http://latimesblogs.latimes.com/technology/2009/05/invented-text-messaging.html [Online; tilgået 1-November-2012].

[17] Chelsi Nakano. Twitter Stories: Expanding Beyond the 140 Character Limit. CMSWire.com, http://www.cmswire.com/cms/customer-experience/twitter-stories-expanding-beyond-the-140-character-limit-013346.php, 2011. [Online; tilgået 27-October-2012].

[18] Kuross Amri og Tom Ceglarek. SMS: How Does It Work? feit.uts.edu.au/,http://services.eng.uts.edu.au/userpages/kumbes/public_html/ra/sms/. [Online; tilgået 04-December-2012].

[19] PCMag. Definition of: GSM. PCMAG.COM, http://www.pcmag.com/encyclopedia_term/0,1237,t=GSM&i=43985,00.asp, 2012. [Online; tilgået 27-October-2012].

[20] Lars Pettersson. SMS and the PDU format. Dreamfabric.com, http://www.dreamfabric.com/sms/, 2005. [Online; tilgået 4-November-2012].

[21] Post og Tele Museum Denmark. Mobiltelefoner. PTT-Museum.dk, http://www.ptt-museum.dk/faq/mobiltelefoner/, 2012. [Online; tilgået 7-November-2012].

[22] Roger Seeck. Adaptive Distribution. Binaryessence.com, http://www.binaryessence.com/dct/en000085.htm, 2011. [Online; tilgået 04-December-2012].

[23] Roger Seeck. Dynamic Distribution. Binaryessence.com, http://www.binaryessence.com/dct/en000084.htm, 2011. [Online; tilgået 04-December-2012].

[24] Roger Seeck. Huffman Code: Example. Binaryessence.com, http://www.binaryessence.com/dct/en000080.htm, 2011. [Online; tilgået 04-December-2012].

[25] Roger Seeck. Static Distribution. Binaryessence.com, http://www.binaryessence.com/dct/en000083.htm, 2011. [Online; tilgået 04-December-2012].

[26] Patrick Seitz. Friedhelm Hillebrand hit send on Text Messaging Reboot: The German Engineer changed the Way We Chat. *Investor's Business Daily*, 2012. http://search.proquest.com/docview/1032814751?accountid=8144 [Online; tilgået 6-November-2012].

[27] Smartphone Freeware. SMS Zip. Smartphone-Freeware.com, http://www.smartphone-freeware.com/download-sms-zip.html, 2009. [Online; tilgået 7-November-2012].

[28] SMSzipper. What is SMSzipper? SMSzipper.com, http://smszipper.com/en/about/, 2012. [Online; tilgået 7-November-2012].

[29] Telenor Danmark. Udlandspriser Erhverv. Telenor.dk, http://www.telenor.dk/erhverv/kundeservice/mobil/udlandspriser/, 2012. [Online; tilgået 28-October-2012].

[30] Telia Denmark. Priser og Zoner. Telia.dk, http://telia.dk/mobil/kundeservice/udland/priser/zoner, 2012. [Online; tilgået 28-October-2012].

[31] The Unicode Consortium. What is Unicode? Unicode.org, http://www.unicode.org/standard/WhatIsUnicode.html, 2011. [Online; tilgået 27-October-2012].

[32] Twitter. New User FAQ. Twitter.com, https://support.twitter.com/articles/13920-get-to-know-twitter-new-user-faq, 2012. [Online; tilgået 27-October-2012].

[33] Wikipedia. Huffman coding. Wikipedia.org, http://en.wikipedia.org/w/index.php?title=Huffman_coding&oldid=522385524, 2012. [Online; tilgået 14-November-2012].

[34] Wisegeek. What is Lossless Compression? wiseGEEK.com,http://www.wisegeek.com/what-is-lossless-compression.htm, 2012. [Online; tilgået 06-December-2012].

[35] Gyldendals åbne encyklopædi. Mobilkommunikation. denstoredanske.dk,http://www.denstoredanske.dk/It%2c_teknik_og_naturvidenskab/Elektronik%2c_teletrafik_og_kommunikation/Antenner_og_b%C3%B8lger/mobilkommunikation. [Online; tilgået 06-December-2012].

Bilag

				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5	6	7
0	0	0	0	0	@	Δ	SP	0	¡	P	¿	p
0	0	0	1	1	£	_	!	1	A	Q	a	q
0	0	1	0	2	$	Φ	"	2	B	R	b	r
0	0	1	1	3	¥	Γ	#	3	C	S	c	s
0	1	0	0	4	è	Λ	¤	4	D	T	d	t
0	1	0	1	5	é	Ω	%	5	E	U	e	u
0	1	1	0	6	ù	Π	&	6	F	V	f	v
0	1	1	1	7	ì	Ψ	'	7	G	W	g	w
1	0	0	0	8	ò	Σ	(	8	H	X	h	x
1	0	0	1	9	Ç	Θ	)	9	I	Y	i	y
1	0	1	0	10	LF	Ξ	*	:	J	Z	j	z
1	0	1	1	11	Ø	1)	+	;	K	Ä	k	ä
1	1	0	0	12	ø	Æ	,	<	L	Ö	l	ö
1	1	0	1	13	CR	æ	-	=	M	Ñ	m	ñ
1	1	1	0	14	Å	ß	.	>	N	Ü	n	ü
1	1	1	1	15	å	É	/	?	O	§	o	à

Figur 4.1: Her ses skemaet for bit 7 tegnsættet. Ved 0x12 ses tegnet 1), der er et escape tegn til den ekstra tabel, som ses på figur 4.2

				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5	6	7
0	0	0	0	0					|			
0	0	0	1	1								
0	0	1	0	2								
0	0	1	1	3								
0	1	0	0	4		^						
0	1	0	1	5							€	
0	1	1	0	6								
0	1	1	1	7								
1	0	0	0	8			{					
1	0	0	1	9			}					
1	0	1	0	10	3)							
1	0	1	1	11		1)						
1	1	0	0	12				[				
1	1	0	1	13				~				
1	1	1	0	14				]				
1	1	1	1	15			\					

Figur 4.2: Den ekstra tabel for bit 7 tegnsaettet