Et studie i applikasjonsutvikling for ulike kommunikasjonsteknologier og UI på mobilterminaler

av

Kim Tommy Humborstad
Knut-Sølve Furset Urke

Hovedoppgave til mastergraden i informasjons- og kommunikasjonsteknologi

Høgskolen i Agder
Fakultet for teknologi
Grimstad, juni 2004

Sammendrag

Den siste tiden er mobiltelefoners bruksområder blitt kraftig utvidet i form av støtte for tredjepartsprogramvare. SEA (Smart Energy Applications AS) har utviklet et trådløst styringssystem for smarthus basert på Bluetooth. I denne sammenhengen ønsker SEA å undersøke mulighetene for å benytte mobilterminaler til styring av Bluetooth nettverket.

I denne oppgaven evalueres applikasjonsplattformer og kommunikasjonsteknologier, spesielt J2ME (Java 2 Micro Edition), Bluetooth og GPRS, med fokus på brukergrensesnitt. Videre diskuteres løsninger for trådløs styring av SEAs Bluetooth nettverk, for å avgjøre om dette er mulig å gjennomføre med dagens mobiltelefoner.

Basert på egenskapene til de forskjellige kommunikasjonsmediene, kunne flere mulige løsninger på problemstillingen presenteres. En undersøkelse av brukergrensesnitt på mobiltelefoner, avgjorde om løsningene kunne presenteres på en intuitiv og brukervennlig måte. En prototyp er utviklet, som demonstrerer deler av disse temaene.

I oppgaven presenterer vi krav og problemstillinger ved lokal- og fjernstyring av SEAs Bluetooth nettverk, og har skissert flere forslag til arkitekturer som gir løsninger på problemstillingene. Av disse løsningene for fjernstyring, involverer den enkleste løsningen SMS basert kommunikasjon. En mer fullstendig løsning vil benytte GPRS, men dette medfører økt implementasjonsmessig kompleksitet. For lokalstyring representerer Bluetooth den beste løsningen. Vi har, for disse arkitekturene, kommet frem til at J2ME applikasjoner vil kunne tilby intuitive, plattformuavhengige og funksjonelle løsninger for lokal- og fjernstyring av SEAs Bluetooth nettverk.

Det gjenstår å implementere en mer fullstendig demonstrator for styring av SEAs Bluetooth nettverk. Det bør også kartlegges hvor mange mobiltelefoner som for fremtiden vil støtte aktuelle J2ME API'er, slik at det senere vil kunne tas et korrekt valg av løsning basert på dette. Samtidig bør det undersøkes hvordan den fremtidige introduksjonen av 3G-terminaler og -mobilnett vil påvirke mulighetene for å kunne fjernstyre et smarthusnett over IP.

Forord

Denne oppgaven er den siste i Mastergradutdannelsen i Informasjons- og kommunikasjonsteknologi ved Høgskolen i Agder (HiA), Grimstad.

Oppgaven er skrevet for Smart Energy Applications AS (SEA), og deler av prosjektet ble utført i SEA sine lokaler ved Teknologiparken i Grimstad. Vi ønsker å rette en takk til SEA, som gav oss muligheten til å skrive om et emne vi finner svært interessant og fremtidsrettet. Vi ønsker også å takke for lokaler og utstyr som ble stilt til disposisjon for oss, noe som gjorde at vi kunne basere forskningen på relevant materiale. Hvilket førte til en aktuell og adekvat løsning.

I tillegg vil vi takke høgskolelektor Magne Arild Haglund som også sto som formell veileder for prosjektet. Hans tekniske kunnskaper om kommunikasjonssystemer, og hjelp til å styre prosjektet i riktig retning var svært viktig. Hos SEA ønsker vi å takke sivilingeniør Hans Klouman, som var kontaktperson for oss i bedriften, for hans ideer og kommentarer om arbeidet.

Vi er begge svært tilfreds med sluttresultatet av oppgaven, og vi mener den har gitt oss kunnskap som vil være svært aktuell når vi nå har fullført vår 5-årige utdannelse på HiA, Grimstad.

Kim Tommy Humborstad

Knut-Sølve Furset Urke

Grimstad, 2004

Figurliste

Tabelliste

Innholdsfortegnelse

Akronymer og forkortelser

2.5G	Generasjon 2.5 (GPRS)
3G	Tredje Generasjon
A3	Authentication algorithm
A5	Signalling data and user data encryption algorithm
A8	Ciphering key generating algorithm
ADSL	Asymmetric Digital Subscriber Line
API	Application Programming Interface
APN	Access Point Name
AuC	Authentication Center
BCC	Bluetooth Control Center
BDS	Brew Distribution System
BTS	Base Transceiver Station (basestasjon)
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
EDGE	Enhanced Data GSM Environment
EIR	Equipment Identity Register
EMS	Enhanced Messaging Service
FHSS	Frequency-Hopping Spread-Spectrum
GAPI	Gaming API
GCF	General Connection Framework
GGSN	Gatway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GUI	Graphical User Interface (grafisk brukergrensesnitt)
GW	Gateway
HCI	Host Controller Interface
HSCSD	High Speed Circuit Switched Data
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
IMEI	International Mobile Equipment Identity
IP	Internet Protocol
IR	Infrared
IrDA	Infrared Data Association
J2ME	Java 2 Micro Edition
JAD	Java Application Descriptor file
JAR	Java Archive file
JCP	Java Community Process
JSR	Java Specification Request
Kc	Ciphering key
Ki	Individual subscriber authentication key
ME	Mobile Equipment (mobil enhet)
MS	Mobile Station (mobiltelefon)
KVM	Kilobytes Virtual Machine aka. Kuaui Virtual Machine
L2CAP	Logical Link Control and Adaptation Layer Protocol
LAN	Local Area Network
MAC	Media Access Control
MIDP	Mobile Information Device Profile
MIEP	Mobile Internet Enabling Proxy
MMS	Multimedia Messaging Service

MS	Mobile Station (mobilstasjon)
NAT	Network Address Translation
OBEX	Object Exchange
OPL	Open Programming Language
OS	Operating System (operativsystem)
OTA	Over The Air
PAN	Personal Area Network
PAP	Push Access Protocol
PC	Personal Computer (datamaskin)
PDA	Personal Digital Assistant
PDP	Packet Data Protocol
QoS	Quality of Service
RAND	Random number
RFCOMM	Radio Frequency Communications
RPC	Remote Procedure Call
RTT	Round Trip Time
SEA	Smart Energy Applications AS
SIG	Special Interest Group
SIM	Subscriber Identity Module
SiS	Symbian installation System
SMPP	Short Message Peer to Peer Protocol
SMS	Short Message Service
SMSC	SMS Centre
SPP	Serial Port Profile
SS7	Signalling System number 7
SSL	Secure Socket Layer
TCP	Transport Control Protocol
TLS	Transport Layer Security
TDMA	Time Division Multiple Access
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
UI	User Interface (brukergrensesnitt)
UIQ	User Interface for Quartz
UMTS	Universal Mobile Telecommunication System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
VB	Visual Basic
VFB	Video Frame Buffer
VM	Virtual Machine (virtuell maskin)
WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WEP	Wired Equivalent Privacy
W-HTTP	Wireless Profiled HTTP
WLAN	Wireless LAN
WMA	Wireless Messaging API
WML	Wireless Markup Language
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
XML	Extensible Markup Language

1 Innledning

1.1 Bakgrunn

Smart Energy Applications AS (SEA) ble startet i november 2002 etter et prosjekt om Bluetooth rutingalgoritmer, og ideer rundt strømsparingssystemer for bolighus. Ved å benytte Bluetooth som kommunikasjonsmedie, jobber bedriften med å utvikle et trådløst styringssystem for lys, ovner, alarmer og lignende utstyr.

SEA har kunnet opparbeide stor kunnskap om trådløs kommunikasjon fra dette systemet, og ser at dagens mobiltelefoner som støtter Bluetooth, kan være et svært aktuelt redskap å benytte i lokal- og fjernstyring av smarthusløsninger. Spesielt etter at personsøkertjenesten ble lagt ned i Norge i 2003, har behovet for nye løsninger innen fjernstyring av boliger og hytter vært etterspurt.

De siste årene har mobiltelefonene som utvikles fått langt flere muligheter for utvidelser i form av tredjepartsprogramvare. Dette, kombinert med at flere og flere telefoner støtter kommunikasjonsteknologier som GPRS og Bluetooth, har åpnet for nye bruksområder for mobiltelefoner. Denne oppgaven har som utgangspunkt å undersøke dagens status vedrørende muligheter for å utvikle programvare for mobiltelefoner, og hvordan de ulike egenskaper og ressurser kan utnyttes.

Oppgaven er blitt spesifisert i samarbeid mellom SEA og studentene. Den er likevel objektiv, og tar for seg et svært nytt felt i et marked med tilnærmet ubegrensede muligheter. Oppgaven er derfor også aktuell for andre enn SEA, og gir kunnskaper om muligheter og begrensninger rundt dette feltet.

1.2 Oppgavedefinisjon

I første omgang skal det undersøkes hvilke muligheter som finnes for applikasjonsutvikling for mobiltelefoner. Dette innebærer å evaluere de ulike OS og programmeringsspråk, og undersøke hvordan disse kan utnytte funksjonaliteten i en mobiltelefon. Spesielt for mobilapplikasjoner som avhenger av kommunikasjonsteknologier, skal det vurderes hvilke utviklingsmuligheter som gis. For å kunne løse SEA's behov er det også ønskelig å finne ut hvilke begrensninger de ulike systemene har med tanke på minne, datalagring, brukergrensesnitt og innlastingsmuligheter.

SEA jobber spesielt mot ”smarthus-markedet” og styring av små, intelligente enheter basert på Bluetooth kommunikasjon. SEA er også interessert i å benytte mobiltelefoner til styring av dette nettet. Derfor skal paperet beskrive hvilke krav og problemstillinger som forbindes med å integrere mobiltelefoner i slike nett. Det må anbefales en løsning på styringseksempler ved hjelp av kommunikasjonsteknologier som Bluetooth, GPRS eller eventuelt andre, på mobilterminaler. Samtidig er det ønskelig å undersøke hvordan brukergrensesnitt og innlasting av programvare kan gjøres mest mulig brukervennlig for brukeren, og at vesentlige sikkerhetsaspekter ivaretas på en rimelig måte. Uavhengig av hvilken kommunikasjonsteknologi som benyttes, bør brukeren kunne presenteres for et tilsvarende brukergrensesnitt.

Dersom tiden tillater det kan deler av de resultater som er funnet demonstreres i form av en prototype. Dette er et svært nytt felt, og testing må kanskje gjøres ved hjelp av emulatorer eller eventuelle terminaler som støtter de nyeste spesifikasjonene. Resultatene vil likevel være av stor interesse for SEA i sammenheng med fremtidige løsninger.

1.3 Begrensninger

Selv om det arbeides hardt med å utvikle globale standarder innenfor mobilindustrien, har dette sjelden vært hundre prosent vellykket. Ofte har regioner som Europa, Amerika og Asia bygd ut ulike varianter av de samme teknologiene. Dette har naturligvis resultert i at mobiltelefonprodusentene har vært nødt til å tilpasse sine produkter til de ulike markedene. Det er derfor ulogisk å tilpasse produkter fra én region til produkter som er beregnet for en annen region. Av den grunn har vi valgt å studere det nordiske markedet for mobilterminaler, og har sett bort fra andre regioner.

Det skjer stadig en rivende utvikling innen trådløs teknologi for mobilkommunikasjon. Vi vil i dette studiet naturligvis undersøke teknologiene som benyttes på de nyeste mobiltelefonmodellene, men fordi utviklingen på dette feltet går så raskt, er nye teknologier stadig på fremmarsj. Bluetooth versjon 2.0 er en videreutvikling av den velkjente Bluetooth teknologien som benyttes i mange av dagens mobiltelefoner, men vi vil ikke se nærmere på denne versjonen av Bluetooth siden den ennå ikke er ferdig spesifisert. I tillegg har vi valgt å se bort fra ZigBee, som i likhet med Bluetooth er en trådløs kommunikasjonsteknologi for små enheter over korte avstander. Årsaken til dette ligger i at ingen mobiltelefonprodusenter i nær fremtid ser ut til å velge løsninger basert på en implementasjon av ZigBee.

Et svært viktig felt innen trådløs teknologi, er løsninger som hindrer at sensitive data ikke kan avlyttes av andre. Dette, kombinert med løsninger for å identifisere hvem en kommuniserer med, og hvilke rettigheter disse har, er et svært stort felt. I denne oppaven er det ikke rom for å gå i dybden på sikkerhet og kryptering av datatrafikk. Vi vil likevel påpeke hvilken sikkerhetsproblematikk som eksisterer i de ulike løsningene vi presenterer, uten at vi vil gå veldig i dybden på hvordan denne problematikken kan løses ved en eventuell implementasjon av disse.

Innen løsninger for kort rekkevidde for mobilterminaler, har vi valgt å ikke vurdere IrDA IR, som også levers på en rekke mobilterminaler i dag. Denne teknologien baserer seg på meget begrenset rekkevidde og retningsbestemt en til en kommunikasjon, hvor enhetene har fri sikt mellom hverandre. For SEA sitt nett basert på radiokommunikasjon og ruting, ville en IR løsning ikke være aktuell.

1.4 Tidligere forskning og status på området

Det finnes i dag en rekke forskningsartikler og systemer som er basert på fjernovervåking over mobilnettet ved bruk av mobilterminaler. En felles linje for disse artiklene, er at noen tar for seg systemer der en sentral server samler inn informasjon fra mobilterminaler, og andre tar for seg hvilke alternativer en server har for å tilby informasjon til mobilterminaler. En fjernstyringsløsning for SEA krever derimot raskere toveis kommunikasjon mellom to mobilnoder som er koblet til samme IP-nettverk. I denne sammenhengen er det andre problemstillinger og utfordringer som må håndteres, enn de som er omtalt i de aktuelle artiklene. I tillegg finnes det få artikler som omhandler de nyeste applikasjonsutviklingsmulighetene for mobiltelefoner, og på hvilke måter disse er i stand til å tilby mer intuitive og brukervennlige løsninger for styring via mobilnettet. Vi har likevel funnet følgende artikler, som har vært av interesse under arbeidet med oppgaven:

Hiroshi Kanma, Noboru Wakabayashi, Ritsuko Kanazawa, Hiromichi Ito: Home Appliance Control System over Bluetooth with a Cellular Phone [13]: Uten å gå inn på brukergrensesnitt, og muligheter for Bluetooth støtte i J2ME, ser paperet på Bluetooth styring av smarthusnett via mobiltelefoner med Java støtte.

KK Tan, CY Soh, KN Wang: Development of an Internet Home Control System [12]: I tillegg til å ta for seg Bluetooth kommunikasjon i smarthusnett, ser oppgaven på hvordan slike nett kan kontrolleres via WAP-nettleseren på mobiltelefoner.

Tomasz Keller, Rajmund Paczkowski, Józef Modelski: Using Bluetooth in a System for Integrated Control of Home Digital Network Devices [14]: Oppgaven ser på en løsning der en Bluetooth/HTTP bro benyttes for å rute trafikk mellom et hjemmenettverk basert på Bluetooth, og fjernstyringsløsninger basert på HTTP. Ved å benytte XML koding over HTTP, kan styringsapplikasjoner utvikles for J2ME mobiltelefoner og stasjonære PC'er.

Innen mobiltelefoner og applikasjonsutvikling, er det gjort lite åpen forskning. Primært er dette et resultat av at de enkelte selskapene har utviklet sine egne proprietære OS for sine produkter. Dersom det er gitt muligheter for tredjepartsutvikling, har dette blitt spesifisert fra leverandøren, og har gitt et fast sett med muligheter for utvikleren. Eksempler på dette er rene spill-API'er, og andre høynivå språk, som gir få utviklingsmuligheter ut over det lille feltet de er ment for. Det er bare de 2-3 siste årene at utviklingsmulighetene for PDA'er og mobiltelefoner gått i retning av utnytting av nyere kommunikasjonsteknologier som Bluetooth og GPRS.

Rococo Software [37] er et selskap som er svært langt fremme innen forskjellige kommunikasjonsteknologier og applikasjonsutvikling for mobiltelefoner. De tilbyr i dag spesialisert programvare for større bedrifter som ønsker å gjøre lokale databaser tilgjengelige for ansatte via en rekke mobile plattformer. Rococo Software har i stor grad gjort egne API'er tilgjengelige for andre utviklere, og de deltar aktivt i spesifiseringen av ulike mobile kommunikasjonsteknologier, da spesielt med fokus på programvare. For eksempel var de først ute med en simulator for testing av utviklingsmuligheter for Bluetooth kommunikasjon mellom mobiltelefoner med J2ME, lenge før dette ble tilgjengelig på markedet.

Innen GW og utveksling av data mellom ulike kommunikasjonsteknologier, er Possio [38] et av få firmaer som har spesialisert seg på dette. Ved å benytte Open Services Gateway initiative (OSGi) løsningen, utvikler de broer mellom ulike kommunikasjonsteknologier, som Bluetooth, GPRS og WLAN, i blant annet Java, for å benytte disse mot forskjellige mobile enheter.

1.5 Litteraturstudie

Dette studiet er knyttet til svært ny teknologi som er i fortløpende forandring. På enkelte områder har det derfor vært vanskelig å finne tilgjengelig litteratur av god kvalitet. I arbeidet med å kartlegge ulike applikasjonssystemer, ble det hovedsakelig benyttet SDK dokumentasjon. Denne dokumentasjonen følger ofte kun med som en del av installasjonen til en SDK.

I sammenheng med J2ME og Bluetooth API'ene, er det i dag utgitt to bøker [3][4]. Begge disse bøkene har bred dekning av teknologien. Førstnevnte er derimot utgitt etter at kompatible enheter ble tilgjengelige på markedet, og inneholder derfor informasjon som er mer relevant i denne sammenhengen.

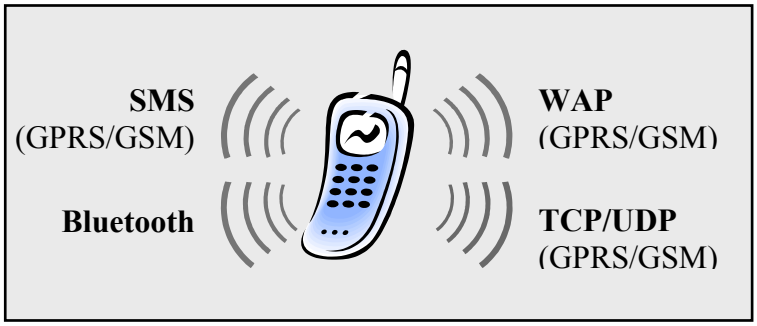
For informasjon om kommunikasjon over lang rekkevidde, i sammenheng med J2ME, anbefaler vi [5], som gir en omfattende dekning av kommunikasjonsmedier og -protokoller for J2ME på mobilterminaler. Den tar for seg en rekke problemer og løsninger. Selv om ingen av disse gir direkte svar på utfordringene som ligger i fjernstyring som definert i denne oppgaven, gir eksemplene relevant informasjon om hvilke muligheter og begrensninger som finnes i J2ME.

Publikasjonen "User Interface Design Guidelines for J2ME MIDP 2.0" [6], som er tilgjengelig i bokform fra Little Springs Design, har vist seg å inneholde mange gode tips for utvikling av brukergrensesnitt for mobilterminaler. Siden det finnes svært sparsommelig med litteratur innenfor dette området i dag, har denne publikasjonen vist seg å være nyttig for oppgaven.

2 Evaluering av mobilteknologier

2.1 Kommunikasjonsteknologier

Dagens mobilterminaler er klargjort for mange ulike kommunikasjonsmedier. Disse utfyller hverandre ved å dekke ulike kommunikasjonsformål. For tale og datakommunikasjon over lengre avstander, benyttes GSM og GPRS som bærere. For kort rekkevidde leveres en del mobilterminaler med Bluetooth, og noen få også med WLAN. På høyere protokollag er det spesifisert en rekke tjenester, som meldingstjenestene SMS og MMS, og WAP for nettverkstilgang. Støtten for ulike kommunikasjonsmedier varierer, men så å si alle dagens mobiltelefoner leveres som et minimum med GSM, GPRS, SMS og WAP.



Figur 2.1: Aktuelle kommunikasjonsteknologier for kommunikasjon mot et hjemmenett.

I dette studiet ser vi for oss to forskjellige scenarier. Ett hvor enheten befinner seg i nærheten av et hjemmenett og kan kommunisere direkte med dette, og ett hvor mobilterminalen er utenfor rekkevidde av hjemmenettet og må benytte andre nettverk for å nå det. Uavhengig av hvor mobiltelefonen befinner seg, bør nettet kunne styres på samme måte, og scenariene deler derfor de samme kravene.

En av de viktigere egenskapene for styring av et slikt hjemmenett, er at brukeren gis tilbakemelding fra nettverket. Dette innebærer at det settes opp en toveiskommunikasjon mot enheter i hjemmenettet, samtidig med at ikke forsinkelsen blir for stor. Dette gjelder både for en eventuell oppkoblingsfase, og idet meldinger sendes gjennom nettet. Ettersom det også er aktuelt å benytte systemet til alarmer og lignende, er det også viktig at det ikke er bare den mobile terminalen som kan initialisere kommunikasjon. Lokalnettet må også være i stand til å ta kontakt med mobiltelefonen.

Ved fjernstyring av hjemmenettet, hvor kommunikasjonen går via andre nettverk, er det som regel en kostnad knyttet til overføring av data. Stort sett snakker vi da om en operatør som, avhengig av kommunikasjonsmediet, tar betalt for nettilgang, overført datamengde, eller antall meldinger.

Sikkerhet bør også ivaretaes, spesielt når data sendes via andre nettverk, men og ved lokal kommunikasjon. Autentisering og autorisering bør anvendes for å gi riktig bruker tilgang til korrekte data, og kryptering for å sikre data under overføring. Som et minimum bør et trådløst medie tilby samme sikkerhet som et trådbasert system.

Avhengig av hva som skal overføres, er også overføringshastighet av betydning for valget av kommunikasjonsmedie. Stort sett er det ved bruk av sanntidsapplikasjoner, og ved overføring av lyd og bilde, at dette er avgjørende. En grundigere gjennomgang av hvilke egenskaper som kreves i forhold til SEA sitt nettverk finnes i Kapittel 3.

2.1.1 Bluetooth

I 1994 startet Ericsson et prosjekt for å finne et alternativ til kabler for tilknytning av datautstyr, mobiltelefoner og PDA'er. Ericsson var allerede sterkt inne i markedet for trådløse løsninger, og brukte sine erfaringer til å utvikle det som skulle bli starten på Bluetooth.

En ny gruppe kalt Bluetooth SIG[1], tok i 1998 over spesifiseringen av denne trådløse teknologien. Den ble dermed åpen for alle produsenter som ønsket å benytte og utvikle Bluetooth. Så å si alle de store produsentene av mobiltelefoner er nå medlemmer av Bluetooth SIG, eller støtter Bluetooth teknologien som fremtidig trådløs teknologi for kort rekkevidde og mobiltelefoner.

Bluetooth er spesifisert ut fra prinsipper om lav kostnad, lavt strømforbruk og liten størrelse. Teknologien er ment som en trådløs versjon av trådbaserte forbindelser, og baserer seg på én til én kommunikasjon uten et sentralt aksesspunkt. Et Bluetooth nettverk vil likevel kunne bestå av en master og opptil 7 slaver i et såkalt piconett. En Bluetooth enhet vil ha muligheten til å være knyttet til flere piconett samtidig, og vil kunne være master i ett, og slave i et annet. Dette kalles et scatternett.

Hastigheten mellom to Bluetooth noder er i Bluetooth versjon 1.x spesifisert til 1 Mbit/s [10]. Grunnet overhead og feiloppretting ligger den praktiske hastigheten en del lavere enn dette, og ved synkron overføring ligger hastigheten rundt 432,6 kbit/s i hver retning [44]. Rekkevidden på Bluetooth enheter varierer mellom 10, 30 og 100 meter, men de fleste mobiltelefoner og PDA'er benytter seg av den korteste rekkevidden da den har et langt lavere strømforbruk.

Ettersom Bluetooth er ment for mindre enheter, og gjerne enheter uten større muligheter for interaksjon med brukerne, er Bluetooth spesifisert til å kunne sette opp en tilkobling nesten uten konfigurering. Enheter vil kunne søke etter andre Bluetooth enheter i nærheten, og finne enhetenes profiler og funksjonalitet. Selve søkefasen kan ta opptil ti sekunder i Bluetooth versjon 1.1, og opptil fem sekunder i versjon 1.2. Selve tilkoblingen fra en enhet til en annen vil også kunne ta i underkant av ett sekund.

[1] Bluetooth Special Interest Group består av en rekke mobiltelefonprodusenter.

Sikkerheten i Bluetooth regnes for å være bedre enn for tidlige 802.11[1] nettverk. Autentisering basert på en kombinasjon av adresse og pin-kode, kombinert med 128 bit krypteringsnøkler, sikrer dataoverføringer mellom to enheter. I tillegg benytter radioteknologien i Bluetooth Frequency-Hopping Spread-Spectrum (FHSS) modulasjon, som bytter frekvens 1600 ganger per sekund i et synkronisert nettverk. Dette gjør nettverket vanskeligere å avlytte, og i tillegg blir det mer robust mot forstyrrelser. [22]

For at enheter automatisk skal kunne opprette en kobling, er det spesifisert en rekke ulike profiler i Bluetooth. Hvis en enhet støtter en bestemt profil, vet andre enheter hvilke meldinger og muligheter denne støtter. På enheter som mobiltelefoner og PDA'er, er det derimot vanlig å støtte en rekke profiler. I API'ene er det likevel vanligst å gi støtte for de mest grunnleggende profilene, og så la utviklerne implementere andre standardiserte eller proprietære profiler og protokoller over dette.

2.1.2 802.11a/b/g

Enkelte av de mer avanserte PDA baserte mobiltelefonene leveres med WLAN støtte. Det er en kommunikasjonsteknologi for kort rekkevidde som blir mest brukt for å gi trådløs LAN støtte. Flere versjoner med ulike modulasjonsmetoder og hastigheter fra 11 til 54 Mbps er spesifisert. En oppnår ikke maks hastighet i hele dekningsområdet, ettersom denne senkes når som distansen og forstyrrelsene mellom nodene øker. Den maksimale rekkevidden for 802.11 standarder er stort sett satt til 100-meter. Ved mye forstyrrelser eller hindringer kan derimot denne avstanden senkes betraktelig. Spesielt de nyere standardene a og g, regnes for å ha lettere for å miste kontakten ved lang rekkevidde. Prisen, strømforbruket og det fysiske volumet til 802.11a/b/g moduler er en del større enn for Bluetooth moduler, og 802.11a/b/g moduler er derfor mindre brukt på mobiltelefoner. Derfor er det ikke en teknologi som kan sies å direkte konkurrere med Bluetooth på mobiltelefoner, men som i steden fungerer som et tillegg på mer avanserte enheter, for å tilby høyere dataoverføringshastighet og støtte for trådløst LAN. Teknologien baserer seg på at en enhet fungerer som et aksesspunkt hvor andre enheter logger seg på. Tilkoblingshastighet regnes derfor som mindre viktig, fordi det forventes at en er koblet til nettet så lenge en er innenfor aksesspunktets dekningsområde.

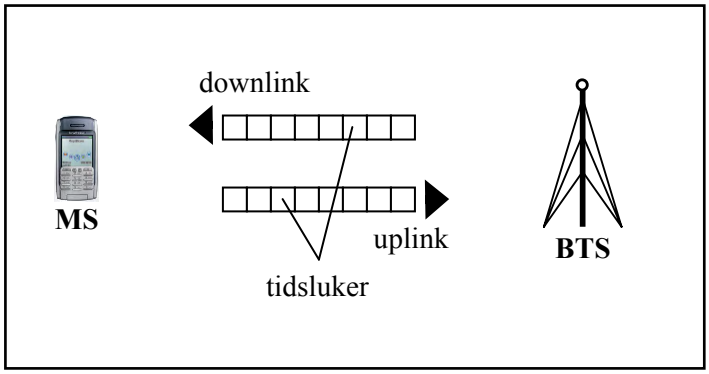
Sikkerheten ved autentisering og kryptering basert på 128 bit nøkler og WEP, har tidligere vist seg å ha flere svakheter, men i dag begynner 802.11i standarden [21], med opptil 256 bit krypteringsnøkler å komme på markedet. Denne har foreløpig ikke vist større svakheter. Selve oppkobling, autentisering og innlogging er mindre automatisert i 802.11 standardene enn i Bluetooth, og 802.11 standardene er derfor ikke like godt egnet for små enkle enheter med begrensede inntastingsmuligheter [20]. 802.11 standardene har derimot hatt en ekstrem suksess de siste årene, noe som har presset prisen per enhet langt ned. Et aksesspunkt er blitt en populær utvidelse av hjemmenettverk med ADSL eller lignende.

[1] 802.11 standardene spesifiserer MAC-lag for trådløst LAN. I denne oppgaven omtaler vi 802.11 standarden og de senere utviklede standardene (802.11a/b/g) som 802.11 standarder.

2.1.3 GSM

Den trådløse evolusjonen startet i Finland i 1991 da tidenes første GSM oppringing ble gjennomført. Siden da har GSM blitt den ledende, og raskest voksende, mobile standarden i verden. GSM er andre generasjons mobilnett, og skiller seg fra første generasjon ved bruk av digital teknologi og TDMA. Det er beregnet at antallet brukere av GSM på verdensbasis nådde 1 milliard i løpet av februar 2004, ifølge en pressemelding fra GSM World [28].

Dataoverføring på radiogrensesnittet deles i en ”uplink” og en ”downlink”, som tar for seg henholdsvis trafikk fra mobilstasjonene til basestasjonene og omvendt. Uplink og downlink tildeles ulike frekvenser, og hver frekvens deles i 8 tidsluker, illustrert i Figur 2.2. At GSM benytter TDMA betyr at opptil 8 brukere kan multiplekses på en og samme frekvens, hvorav en tidsluke tildeles hver bruker. Båndbredden i en slik tidsluke er normalt 9,6 kbit/s ved overføring av data, men under gode forhold kan en få opptil 14,4 kbit/s. Ved dataoverføring er det også mulig å slå sammen flere tidsluker, såkalt HSCSD. Opptil 4 tidsluker kan benyttes samtidig, og den maksimale teoretiske båndbredden blir da 57,6 kbit/s. [7]



Figur 2.2: GSM radiogrensesnitt.

GSM benytter linjesvitsjing, noe som betyr at en bruker av GSM legger beslag på båndbredde selv om det ikke sendes data over linjen. Båndbredden utnyttes derfor dårlig ved datakommunikasjon over GSM. En annen effekt er at det fort kan bli kostbart å sende data over en GSM linje. Dette kommer av at en betaler for tiden en er tilkoblet, og ikke for mengden data som sendes over linjen.

Et radiomedie kan aksesseres av hvem som helst. Derfor er autentisering av mobilabonnenter viktig i systemer som GSM. [8] beskriver denne prosessen. Kort fortalt foregår det slik: En hemmelig autentiseringsnøkkel (Ki) ligger lagret i abonnentens SIM kort, og en kopi av Ki ligger lagret i et Authentication Center (AuC). Under autentiseringen genererer AuC et tilfeldig tall, RAND, som sendes til MS. AuC og SIM kortet i abonnentens mobiltelefon benytter så A3 algoritmen for å generere en Signed Response (SRES) ut fra RAND og Ki. Hvis SRES generert av MS er lik den som ble generert av AuC, er abonnenten autentisert.

RAND og Ki benyttes også av A8 algoritmen for å generere krypteringsnøkkelen Kc. Kc benyttes sammen med TDMA rammenummeret og krypteringsalgoritmen A5 for å kryptere data som sendes over radiolinken. [8]

I tillegg til autentisering av mobilabonnenter, må også selve GSM terminalen (ME) identifiseres av nettverket. Dette skjer ved hjelp av et unikt nummer (IMEI), som ligger lagret i ME. [8]

2.1.4 GPRS

GPRS står for General Packet Radio Service, og er en utvidelse av GSM mobilnettet som tilbyr pakkesvitsjet datatransport. Tjenesten omtales ofte som 2.5G fordi den kan ses på som en forløper til UMTS.

I likhet med HSCSD kan en GPRS tilkobling tildeles flere tidsluker på en gang, og radiogrensesnittet er som på Figur 2.2. Nytt i GPRS er at tidsluker tildeles dynamisk, og dette gjør det mulig for en mobiltelefon å være tilkoblet mobilnettet uten å legge beslag på radioressursene. I tillegg kan opptil 8 tidsluker tildeles samtidig. Tidsluker i GPRS dedikeres dataoverføring, mens GSM reserverer en del av båndbredden for tale. Alt tatt i betraktning blir den teoretisk maksimale båndbredden på hele 171,2 kbit/s, eller 21,4 kbit/s over én tidsluke. En del praktiske hindringer, som at GPRS deler radioressursene med GSM, at flere brukere må dele tidslukene seg imellom, og restriksjoner på GPRS terminalene, gjør at den typiske båndbredden blir noe lavere. [7]

Kostnadene med å sende data over GPRS er betydelig lavere enn over GSM. Dette bunner i at kostnadene for GPRS regnes ut ifra mengden data som overføres. En av egenskapene ved GPRS er at mobiltelefonen alltid er oppkoblet. Dette er praktisk mulig nettopp fordi radioressursene tildeles dynamisk, og fordi en derfor ikke betaler for tiden en er oppkoblet. [7]

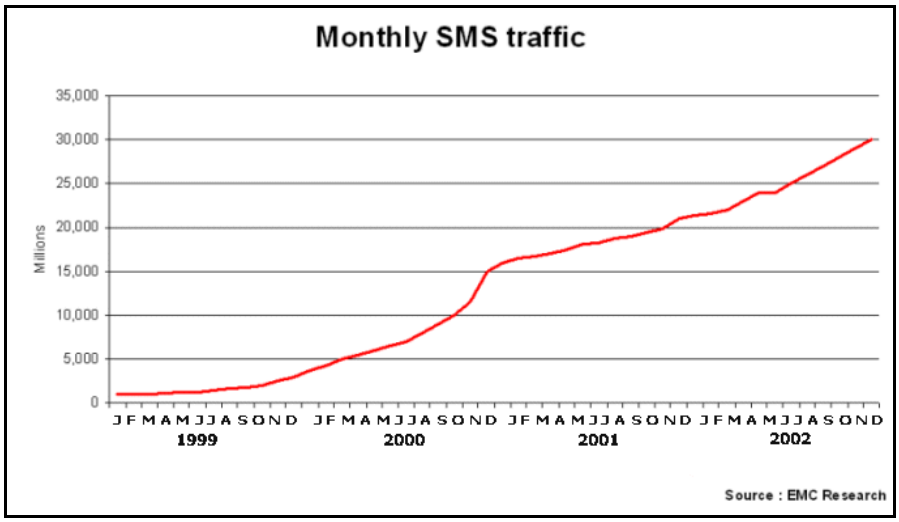
Sikkerhetsmessig er GPRS mer utsatt for angrep enn GSM. Dette bunner stort sett i at GPRS benytter IP i kjernenettet, istedenfor SS7 protokollen, som GSM er basert på. Mens spesifikasjonene for IP er fritt tilgjengelige på Internett, har få hackere kunnskaper nok om SS7 til å være i stand til å hacke denne protokollen. [18]

På den funksjonelle siden er sikkerheten i GPRS ekvivalent til den eksisterende sikkerheten i GSM. SGSN utfører autentiserings- og krypteringsprosedyrer basert på de samme algoritmer, nøkler og kriterier som i GSM. De største forskjellene ligger i at krypteringsalgoritmene som benyttes av GPRS er optimalisert for overføring av pakkebasert data. I tillegg benyttes IP-tunnellering og privat IP adressering innen operatørens GPRS backbone for å hindre uautorisert tilgang til dette, og brukertrafikk til interne nettverkselementer blokkeres. [8]

2.1.5 SMS

SMS ble introdusert tidlig på 1990-tallet, og ble raskt den mest suksessfulle metoden for å sende og motta korte tekstmeldinger på mobiltelefoner. Siden har det dukket opp en mengde bruksområder for SMS, og bruken av tekstmeldinger, illustrert av Figur 2.3, har eksplodert. Opprinnelig var GSM den eneste bæreren av SMS. I dag finnes også muligheter for at operatørene kan tilby GPRS som bærer av SMS, med dertil høyere hastighet enn hva GSM tillater [19]. Ifølge [19] krever dette visse endringer hos operatørenes SMS Centre (SMSC), slik at ikke alle operatører vil tilby SMS over GPRS.

Vi vil ikke gå nærmere innpå MMS av den årsak at MMS kun er økonomisk gunstig dersom det sendes store mengder data på en gang. MMS involverer dessuten flere av teknologiene som er omhandlet i dette dokumentet, som SMS og WAP. Teknologier som utmerket kan utveksle informasjon på egen hånd.



Figur 2.3: Antallet sendte tekstmeldinger skyter i været. [29]

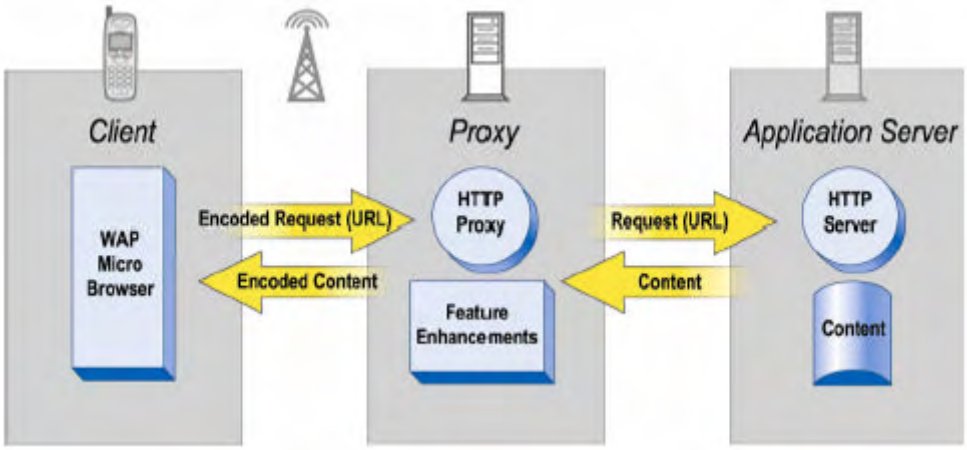
SMS benytter signaleringsdelen av GSM nettet, som innebærer at abonnenten først lokaliseres og autentiseres før meldingen sendes. Selve SMS meldingen består av en header og maksimum 1120 bit data. En 8 bit binær SMS, som vanligvis benyttes for å sende lyd og bilder, kan derfor inneholde opptil 140 oktetter. Mens en vanlig tekstmelding, som er på 7 bit, kan inneholde opptil 160 tegn. [30]

I motsetning til telefonsamtaler over GSM, benyttes det ingen direkte oppkobling mellom sender og mottaker av SMS. En SMS rutes først gjennom GSM nettverket for så å mellomlagres i et SMSC før den sendes videre til mottakeren. Dette tar vanligvis under 2 sekunder, men ved stor trafikk kan forsinkelser på opptil flere dager forekomme. For å motvirke noe av forsinkelsesprosessen, kan en SMS merkes med høy, middels, eller lav prioritet. [30]

Kostnadene for å sende en SMS varierer fra operatør til operatør. Vanligvis ligger prisen for en SMS i dag på godt under 1 krone. Som [19] påpeker, er prisen for å sende en SMS over GSM og GPRS den samme. Den harde konkurransen mellom operatørene er en årsak til at prisene for å sende SMS stadig synker.

2.1.6 WAP

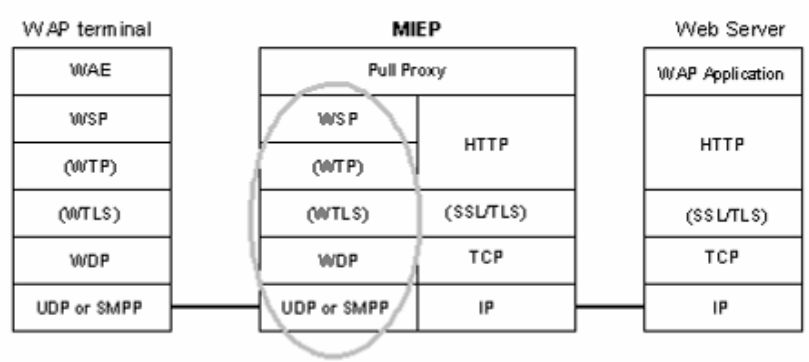
WAP ble første gang spesifisert i 1997 av de ledende mobiloperatørene for å tilby filtrert og komprimert data fra Internett til mobiltelefoner. Arkitekturen er basert på en klient/tjener struktur, hvor tjeneren står som en GW mellom Internett og mobilterminalen. Ved å legge mest mulig intelligens på GW'en kan programvaren på mobilen gjøres så liten og effektiv som mulig.



Figur 2.4: WAP GW arkitektur. Proxy filtrerer og komprimerer data. [9]

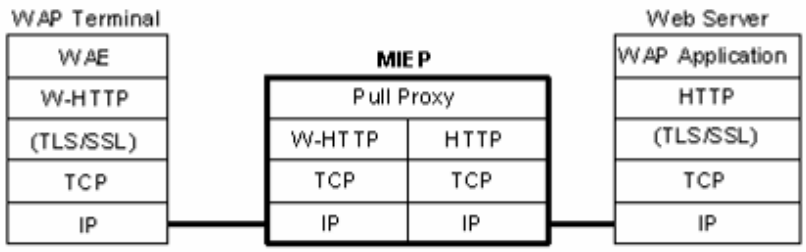
En WAP GW filtrerer og komprimerer trafikk mellom Internett og mobiltelefonen. Andre terminaler på Internett kan ikke kontakte mobiltelefonen direkte, men må kommunisere gjennom GW'en. Dette er spesielt viktig i mobilkommunikasjon hvor det ofte er sammenheng mellom overført datamengde mot mobiltelefonen og kostnader. GW'en hindrer derfor misbruk ved at noen sender uønsket og kostbar data til klienten.

Spesifikasjonene tilhørende WAP arkitekturen omfatter alt fra overføringsprotokoller til presentasjonsprotokoller. Mellom WAP klienten og GW'en i telenettet er det spesifisert en helt ny protokollstakk. Denne skal kunne kjøre over en rekke kommunikasjonsmedier som SMS, GSM, GPRS og UMTS. Ved for eksempel å benytte SMS, vil en forespørsel etter et dokument sendes i en SMS til GW'en, som igjen laster ned dokumentet fra nettverket, og returnerer dette i en SMS tilbake til WAP klienten. WAP protokollen tilbyr dermed indirekte tilgang til HTTP protokollen kjent fra Internett, men kaller sitt applikasjonslag for WSP. For å slippe tung konvertering i GW'en, er også et nytt dokumentformat spesifisert: WML. Denne har et forenklet innhold i forhold til vanlige HTML dokumenter, og passer bedre til skjermer med lavere oppløsning.



Figur 2.5: WAP 1.x protokollstakk. [31]

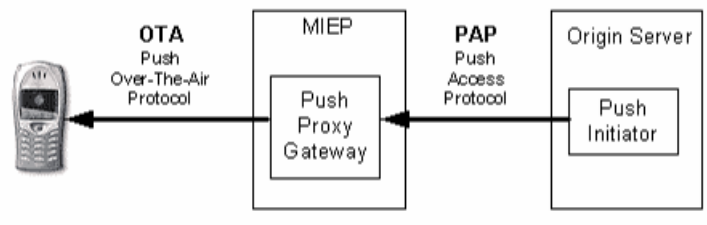
En ny versjon av WAP protokollstakken er også spesifisert, WAP 2.0, som forenkler og forbedrer ytelsen over nyere nett, som GPRS og UMTS. Stakken ligner mer på tradisjonelle IP-nett, og overføres over TCP, men tilbyr samme sikkerheten som tidligere versjoner ved at alle forespørsler går gjennom en WAP GW som på Figur 2.6.



Figur 2.6: WAP 2.x protokollstakk. [31]

WAP fungerer over en rekke medier, og det førte tidlig til at den ble implementert på svært mange mobiltelefoner. Dette har praktisk betydning for mobiltelefonenes applikasjonsplattformer, ettersom det betyr at mobiltelefoner som tilbyr nedlasting fra datanett, kan tilby dette uten å ha implementert verken GPRS eller en vanlig IP stakk. En applikasjonsplattform gir derfor stort sett tilgang til det øverste laget, altså HTTP forespørsler og svar, og ikke tilgang til lavere lag. En er derfor begrenset til WAP/HTTP basert kommunikasjon, og systemer der den mobile enheten opptrer som en klient.

Et av de større problemene med denne arkitekturen er at det er svært vanskelig for enheter på nettet å initialisere kontakt med mobiltelefoner. WAP 1.2 fikk derfor tilføyd en teknologi kalt Push, som introduserte muligheten for å overføre informasjon til klienten uten forespørsel fra brukeren. For å oppnå dette kreves det en Push Proxy Gateway, som noder på Internett kan kontakte for å overføre data til WAP baserte mobiltelefoner. En egen protokoll basert på XML benyttes for å snakke med GW'en, og inneholder informasjonselementer om hvor en melding skal videresendes. Dette kan være telefonnummer, IP eller WAP id. Hvis mobiltelefonen har mulighet for GPRS tilkobling, kan nettet sette opp en PDP-kontekst og overføre meldingen over IP. Dersom dette ikke er mulig, vil meldingen kunne bli levert direkte over SMS, men noe av målet med arkitekturen er å kunne tilby en mer integrert og funksjonell løsning enn hva vanlig SMS tilbyr.



Figur 2.7: WAP Push noder. [31]

Sikkerheten i WAP arkitekturen er implementert i et eget WTLS lag som benyttes mellom mobiltelefonen og WAP GW'en (Figur 2.5). Dermed blir overliggende protokollag beskyttet, uavhengig av om underliggende bærerteknologi har kryptering eller ikke. Videre ut på Internett kan GW'en benytte SSL.

2.1.7 Høyere lag

På grunn av egenskapene de lavere lagene har i de mobile kommunikasjonsprotokollene, og da spesielt WAP, som gir mobiltelefoner HTTP støtte uten å kreve TCP/IP på lavere lag, må en også se på hva som er aktuelle protokoller på applikasjonslagsnivå.

Nettopp på grunn av WAP, benytter hovedtyngden av nettverksapplikasjoner på mobilterminaler helst protokoller over HTTP for å støtte flest mulig terminaler. Dette kan være binære proprietære protokoller implementert over HTTP, hvor det benyttes standard HTTP meldinger som POST eller "URL parametere" i GET-forespørsler for å sende data.

En annen aktuell løsning er å benytte Web Services, som er en forenklet versjon av tradisjonelle RPC kall. Web Services benytter XML koding, og kan sendes over HTTP, noe som har gjort at dette anses som svært aktuelt på mobile terminaler. Dette er til tross for at XML basert koding vil ha mye overflødig data i forhold til binære protokoller, noe som gjør at dette krever mer ressurser på mobilterminalen. Blant annet benyttes Web Services i Parlay X systemet [11], hvor en tjener med tilgang til mobilnettverket kan tilby mobilnettverkstjenester via Web Services.

Uansett hvilken protokoll som benyttes over HTTP, må overliggende protokoller aktivt spørre for å få tilsendt data, da HTTP ikke opprettholder toveiskommunikasjon. Dette kreves ikke dersom TCP eller UDP benyttes direkte over GPRS eller GSM. Ved å benytte TCP vil en kunne opprette toveiskommunikasjon mellom to noder, og begge nodene vil kunne sende data direkte til mottaker. Det samme gjelder dersom SMS meldinger benyttes. Se oversikt over protokollene i Figur 2.9, Kapittel 2.1.8.

2.1.8 Vurdering av kommunikasjonsteknologier

I oversikten over kommunikasjonsmedier som finnes på mobilterminaler i dag, delte vi teknologiene inn i kategorier for kort og lang rekkevidde. I tillegg så vi på høyere lags protokoller, og hvilke fordeler og ulemper som knyttes til å benytte disse i de ulike kommunikasjonsløsningene. Alternativene for kommunikasjon over kort rekkevidde er 802.11 standardene og Bluetooth. I dag er det svært få mobiltelefoner som leveres med 802.11 implementert, mens Bluetooth er nær ved å bli en standard på mobiltelefonene som nå kommer på markedet. Rekkevidden av Bluetooth, på implementasjonene som leveres på mobiltelefoner, ligger på 10 eller 30 meter, noe som ved enkelte løsninger kan være i korteste laget. 802.11 standardene har lengre rekkevidde, og er blitt svært populære som trådløs internettilgang i vanlige hjemmenettverk. Både Bluetooth og 802.11 standardene har etter hvert implementert like god sikkerhet, selv om Bluetooth har prioritert en enklere krypteringskonfigurering som er bedre tilpasset små enheter.

Avhengig av hvilke måter en mobiltelefon er ment å ta direkte kontakt med andre enheter eller nettverk, har de to kommunikasjonsteknologiene forskjellige egenskaper. Teknologiene kan derfor leve side ved side, og trenger ikke direkte settes opp mot hverandre. Bluetooth vil likevel, på bakgrunn av store fordeler innen strømforbruk, pris og forenklet tilkobling, etter all sannsynlighet være den standarden som blir levert på de fleste mobiltelefoner i fremtiden. Selv om Bluetooth i dag kommer litt til kort på grunn av lav båndbreddekapasitet, vil båndbredden i fremtidige versjoner av standarden stige i takt med at kapasiteten til mobiltelefoner og PDA'er øker. For mange løsninger for kort rekkevidde kan en derfor si at Bluetooth er det mest realistiske valget for trådløs kommunikasjon for mobilterminaler. For fremtiden skal en likevel ikke utelukke 802.11 standardene, som i senere tid har hatt stor suksess på markedet, og både pris og størrelse på hardware er blitt presset vesentlig ned. Ettersom batterikapasitet og datakapasitet øker på mobilterminaler, er det ikke utenkelig at flere av disse etter hvert også vil leveres med WLAN for å kunne tilby blant annet IP-telefoni ved lokale aksesspunkter.

Tabell 2.1: Sammenligning av trådløse standarder på mobiltelefoner, modifisert fra [36].

Teknologi / Egenskap	Bluetooth 802.15.1	WLAN 802.11a/b/g	GSM	GPRS	SMS	WAP
Applikasjons-fokus	Kabel-erstatning	Web, Video, E-mail	Telefoni	Nettverks-tilgang	Meldings-tjenester	Nettverk Internett
Batteritid (dager)	1-7	0.1-5	1-7		-	-
Noder per nettverk	7	30-255	1000		-	-
Båndbredde (maks. kbit/s)	720	11 000-54 000	9.6-57,6	21,4-171,2	Avhengig av bærer	
Rekkevidde (meter)	1-10+	1-100	1000+		Avhengig av bærer	
Fordeler	Pris, Anvendelig-het	Hastighet, Fleksibilitet	QoS[1]	Pris Alltid på	Enkelt Bred støtte	Rekkevidde Kvalitet

For fjernstyring av hjemmenettet via mobilnettet, er GSM og GPRS aktuelle bærere for datatrafikk. Disse har igjen tjenester på høyere lag, som SMS og WAP, som har andre egenskaper og egne prissystemer. Vi kan i tillegg dele løsninger basert på GSM og SMS i løsninger der datakommunikasjonen kun går via mobil-/telenettet, og løsninger der teknologien benyttes til å koble seg til andre nettverk.

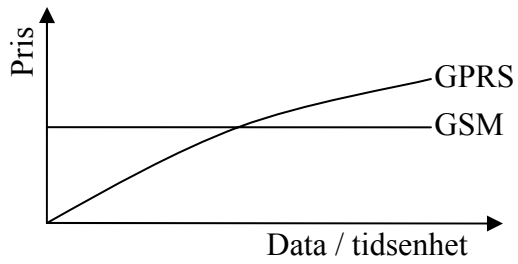
Dersom vi benytter GSM eller SMS via mobilnettet, kan vi utnytte at adresseringen gjøres i selve mobilnettet ved hjelp av globale telefonnummer. Dette gjør at de fleste løsninger basert på GSM eller SMS ikke krever tilleggsutstyr eller -tjenester annet enn direktekommunikasjon mellom mobilterminaler i mobilnettet. Når SMS benyttes for dataoverføring, støter en derimot på ulemper med blant annet kostnader, overføringshastighet og latens. For eksempel har SMS i dag en relativt høy sendekostnad i forhold til overført datamengde, siden SMS prises per melding uavhengig av hvor mye data hver melding inneholder. I tillegg er den svært variable latensen i SMS lite egnet til de fleste datakommunikasjonssystemer hvor det er nødvendig med synkron kommunikasjon.

Ved å benytte dataoverføring over GSM i mobilnettet vil en få opptil 9,6 kbit/s overføringshastighet, eller opptil 57,6 kbit/s ved HSCSD. Dette er i nærheten av GPRS, og langt bedre enn SMS, men kostnadene for dataoverføring blir sett ut fra oppkoblet tid, og ikke overført datamengde. Dersom en konstant skal overføre data, kan denne løsningen være lønnsom i forhold til GPRS, men den vil være mindre lønnsom dersom en overfører lite, og gjerne intervallbasert data. En kan selvsagt koble ned linjen dersom en har et lengre opphold i datatrafikken, men dette vil på grunn av oppkoblingstiden føre til at det tar lengre tid å starte overføringen på ny.

[1] Quality of Service. Dersom GSM benyttes til å direkte kontakte en enhet i telenettet/mobilnettet, vil en få en linjesvitsjet forbindelse med garantert båndbredde.

Sikkerheten i en løsning hvor GSM eller SMS benyttes til å sende data direkte mellom to enheter i mobilnettet er god. Det er et krav at data som sendes trådløst over GSM er kryptert, og mobiltelefon og SIM kort blir autentisert før en oppringing kan utføres. En løsning basert på denne typen kommunikasjon vil derfor sjelden kreve ekstra kryptering eller autentisering på høyere nivå.

En GPRS løsning vil være basert på at en kobler seg til et annet nettverk. Som regel tilbyr mobiloperatørene tilgang til for eksempel Internett og MMS tjenester via GPRS. Operatørene tar vanligvis betalt for overført data og ikke tilkoblet tid som i GSM. Avhengig av hvilket nett en kobler seg til vil operatørene kunne ta varierende betaling for overført data. Normalt vil en derfor kunne ha en konstant forbindelse ettersom det er vanlig å ikke ta betalt for signaleringsmeldinger og tilkobling. En GPRS løsning vil derfor være lønnsom dersom en ikke har konstant maksimal datatrafikk. Figur 2.8 illustrerer forholdet mellom overført data per tidsenhet og kostnader i løsninger basert på GPRS og GSM.

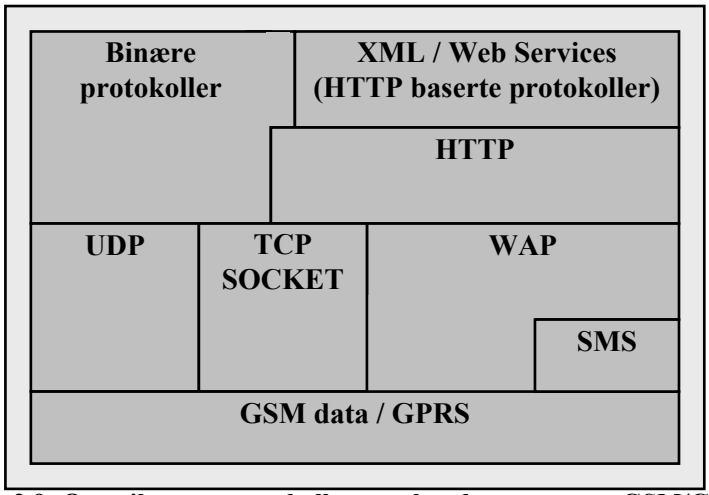


Figur 2.8: Sammenligning av prisutvikling for GPRS og GSM ved overført data per tid.

Adressering av den mobile enheten kan være et problem i GPRS, da en ikke kan adressere mobiltelefoner tilkoblet et IP-nettverk ved hjelp av telefonnummer. Ettersom IP-adresser vanligvis blir tildelt dynamisk av mobiloperatøren, må løsninger basert på GPRS benytte tilleggstjenester i nettverket, som WAP Push (se nedenfor), eller implementere egne lokasjonsservere, for å tilby adressering. Disse påvirker også sikkerheten, da GPRS kun krypterer datatrafikken i mobilnettet. Kryptering og autentisering må derfor implementeres på høyere lag, siden trafikken sendes via andre nettverk. En fordel ved å benytte GPRS i forhold til kun å benytte mobilnettet, er at GPRS forenkler interaksjonen mot noder i andre nettverk. For eksempel kan en kontakte alle enheter på Internett via en GPRS tilkobling mot Internett, og motsatt. Dette er vanskeligere å oppnå i en GSM løsning som kun går via mobilnettet.

WAP er i likhet med GPRS avhengig av at en kobler seg til et nettverk, men både GSM, SMS og GPRS kan benyttes som bærere av WAP. På grunn av dette finnes WAP implementert på så å si alle mobiltelefoner på markedet i dag. WAP krever derimot HTTP basert kommunikasjon, noe som kan være uegnet og noe tyngre å implementere for enkelte løsninger. Dette vil si at kommunikasjon mot en WAP enhet ikke kan intialiseres direkte fra andre noder på nettverket. Derimot inkluderer WAP spesifikasjonen WAP Push, som tilrettelegger for adressering ved å benytte mobiltelefonens telefonnummer. Den åpner også for at kontakt kan initialiseres fra andre noder i nettverket. Mange mobiloperatører tilbyr derimot WAP Push kun gjennom spesialavtaler, noe som har ført til at det benyttes i liten grad i dagens systemer.

Ved å benytte TCP sockets eller UDP trafikk direkte mot nettverket, står en fritt til å implementere protokoller på høyere lag. En vil da også få muligheten for at enheter i nettverket kan initialisere kontakt mot mobiltelefoner ettersom kommunikasjonen er basert på IP. En mister derimot muligheten for å benytte SMS som bærer, og sikkerheten blir noe dårligere fordi det ikke er en server som filtrerer trafikk mot mobiltelefonene. Se Figur 2.9 for en oversikt over hvilke protokoller og kommunikasjonsteknologier som er aktuelle å benytte ved kommunikasjon mellom enheter over mobilnettet.



Figur 2.9: Oversikt over protokoller som kan benyttes over GSM/GPRS.

Mange av dagens mobiltelefonbaserte styringssystemer er basert på SMS. Selv om dette kan være den dyreste og tregeste måten å sende data på, regnes den ofte som den metoden som er enklest å implementere. Hva som egner seg best som kommunikasjonsteknologi er derfor avhengig av hvor mye og hvor ofte data sendes, og hvor avansert løsningen skal være. I tillegg til dette er også støtten for å utnytte de ulike kommunikasjonsteknologiene varierende for tredjepartsprogramvare på mobiltelefoner, noe som igjen påvirker valget av en løsning.

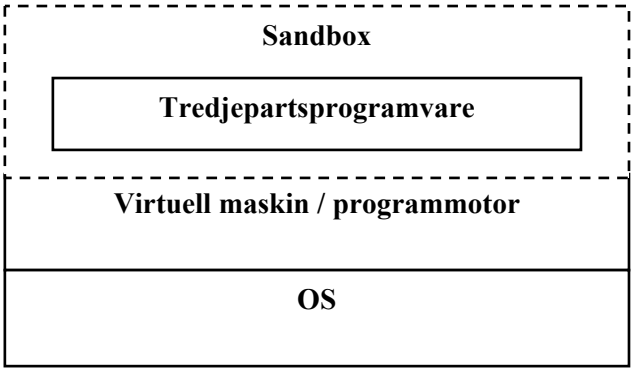
2.2 Operativsystemer og programmeringsspråk

For få år siden ble de fleste mobiltelefoner levert med et fast sett med programmer og funksjonalitet. Alternativene for å legge inn ekstra programvare var enten forbeholdt servicesentre eller produsentene. Begrenset hardware kombinert med de utfordringer som ligger i å beskytte brukere mot ødeleggende software, var de største hindringene. Etter hvert begynte mulighetene å bli flere, men da også i svært begrensede miljøer. Tredjeparts programvare fikk stort sett kun begrenset tilgang til skjermen, og applikasjoner som kom på markedet var stort sett enkle underholdningsapplikasjoner og enkelte mindre huskelapp- og kalkulatorprogrammer.

Mobile enheter har derimot endrest seg drastisk i forhold til kapasitet de siste to til tre årene. Avanserte PDA'er har fått utvidelser med en rekke kommunikasjonsteknologier, og mobiltelefoner har etter hvert overtatt en del av egenskapene til PDA'ene. Dette har ført til at mobilterminalene som tidligere baserte seg på spesialisert programvare, nå ofte kommer med et større OS som har flere utvidelsesmuligheter.

Ved i tillegg å la tredjepartsprogrammer kunne utnytte ressurser som GSM og GPRS på en forsvarlig måte, har en fått uante muligheter for applikasjonsutvikling på mobiltelefoner. Nyere utviklingsplattformer bør også kunne gi støtte for høyere lags langdistanse kommunikasjonsteknologier som SMS, MMS, WAP, og eventuelt kortdistanse kommunikasjon som terminalen støtter (Bluetooth, WLAN og IR).

Hovedproblemet med å tilby tredjepartsprogramvare på mobiltelefoner, er muligheten for at usignert kode kan utgjøre en potensiell trussel mot systemet, og gjøre det ustabilt. Dette er et større problem på mobile plattformer enn for tradisjonelle operativsystemer ettersom mobile enheter så å si aldri slås helt av, og ikke bør restartes. Derfor må OS'et på mobiltelefonen kontrollere all kode som kjøres. Dette krever blant annet effektiv minnehåndtering som kan forhindre minnelekkasje fra programmer, og som frigjør systemressurser så fort de ikke lenger er i bruk. Metoden som benyttes av de fleste operativsystemene i dag, er å implementere en egen programmotor som står for kontroll og eksekvering av programkoden. Ofte sies det at programmet kjøres i et såkalt sandbox miljø, som skal beskytte systemene som ligger utenfor.



Figur 2.10: Eksempel på beskyttet eksekvering av programkode.

Selv om OS'et skal beskytte mot ødeleggende programmer, er det nok av utfordringer programvaren selv må håndtere. Den må kunne håndtere situasjoner som sjelden oppstår i større systemer. Dette gjelder blant annet håndtering av svært ustabile kommunikasjonsmedier, og hva som skal skje dersom det ikke er nok ledige systemressurser for å utføre en handling. Derfor må alle nivåer i systemet programmeres på en sikker måte, helt fra operativsystemnivået til applikasjonsnivået.

En annen utfordring som en applikasjonsutvikler vil støte på, er de mange ulike konfigurasjonene som mobiltelefoner kommer med. Det finnes ekstremt mange forskjellige typer display. En finner alt fra svart-hvitt til millioner av farger, i tillegg til variasjoner i størrelse fra noen få tusen punkter til flere hundre-tusen. Det er opp til OS'et å bestemme hvor stor tilgang programmene kan ha til det grafiske grensesnittet. En applikasjon skal se korrekt ut på tvers av ulike typer mobiltelefoner, samtidig som den ellers bør følge mobiltelefonens brukergrensesnitt for å forenkle og forkorte opplæringsprosessen for brukerne. I tillegg til dette er det varierende hva som finnes av input ressurser til systemet. Som eksempel kan det nevnes alt fra tradisjonelle telefontastatur til touch screen. Alt i alt kan en si at med de krevende oppgavene et operativsystem har på trådløse enheter, kan det ikke nødvendigvis kalles et mini-OS, men et annerledes OS.

For utviklers del er det også viktig at programkode kan benyttes på tvers av terminaler, ettersom markedet består av en rekke mobiltelefontyper og proprietære løsninger. Dette kan kun oppnås ved at en benytter hardwareuavhengige applikasjonsplattformer. Vi har i de neste kapitlene tatt for oss de mest aktuelle systemene som benyttes på mobiltelefoner. For hvert av disse har vi blant annet gått inn på brukervennlighet, ressursutnyttelse, datalagring, eksekveringshastighet, og muligheter når det gjelder utvikling av ny programvare, som nevnt over.

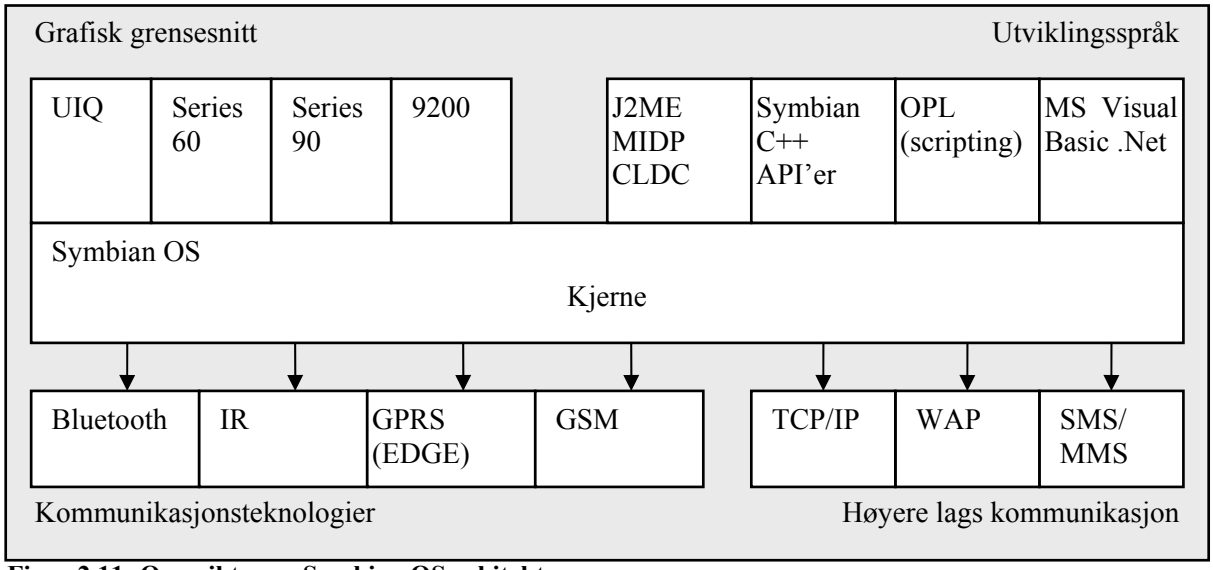
2.2.1 Symbian OS v7.0s

Symbian ble startet i juni 1998 av en rekke av de ledende telekommunikasjonsselskapene, for å lage en felles plattform for mobiltelefoner. Medlemslisten anno 2004 består av Ericsson, Nokia, Panasonic, Psion, Samsung Electronics, Siemens og Sony Ericsson, selv om enda flere har lisens for å benytte OS'et. Formålet med Symbian var å skape en bedre og felles utviklingsplattform gjennom et åpent OS, men selvsagt også som et svar på at mobiltelefoner har fått kraftigere hardware, og har muligheter for å kjøre større OS.

Symbian inkluderer en rekke egenskaper som tidligere kun var vanlige på PDA'er. Blant annet har OS'et et filsystem, og er i stand til å benytte forskjellige typer lagringsmedier. Programmer som kjører på Symbian har også muligheter for å aksessere filsystemet, og lagre og opprette filer. Symbian har allerede støtte for 3G standarder, og flere mobiltelefoner er allerede kommet på markedet. I skrivende stund har det blitt levert rundt 15 mobiltelefoner [32] med Symbian OS.

Symbians arkitektur kan deles i 2 deler. Hovedkjernen som tar seg av nettverksressurser og protokollstakker, og det grafiske grensesnittet som kan endres fra leverandør til leverandør. På grunn av sikkerhetshensyn, må det ved implementasjoner av OS'et innhentes lisens for å endre på lavere lags protokoller i kjernen. Stort sett er dette unødvendig da det finnes svært gode API'er mot nettverk, mobiltelefon og multimedia. Selve det grafiske grensesnittet er derimot helt åpent for endringer fra leverandørene. I dag finnes det fire offisielle grafiske plattformer (Figur 2.11). UIQ er beregnet for mobilterminaler som har en større touch-screen. Nokia har utviklet 9200, Series 60 og 90 som har produktserier med tilsvarende nummerering. De har derimot valgt å distribuere Series 60 versjonen som et offisielt GUI for Symbian OS. Det blir i dag derfor levert på Symbian mobiltelefoner fra andre leverandører også. Dermed forventes det at applikasjoner utviklet for denne kombinasjonen vil kunne kjøre på flere mobiltelefoner.

Symbian ble designet med tanke på å være utvidbart, og gir derfor gode utviklingsmuligheter for tredjepartsleverandører. Som standard leveres mobiltelefoner basert på Symbian med støtte for applikasjoner utviklet i Java, C++, OPL (Open Programming Language), og Microsoft Visual .NET (VB .NET) og Visual Basic (VB). Både Java og C++ applikasjoner kan installeres ved hjelp av OTA teknologi. For C++ har Symbian sitt eget installasjonssystem kalt SiS (Symbian installation System), og ved hjelp av dette kan C++ applikasjoner enkelt og greit lastes ned over blant annet WAP og HTTP. Nedlasting av større filer over WAP kan derimot gi problemer. For Java støttes OTA som spesifisert av Sun. For alle programmeringsspråkene gjelder det at kompilerte programmer kjøres i et beskyttet miljø, slik at de ikke skal kunne utføre uønskede eller kostbare tjenester uten at brukeren er klar over det. OS'et i seg selv støtter også flere mulige innenheter som numerisk mobiltastatur, håndskriftgjenkjenning og trykkfølsomme skjermer. Disse kan kontrolleres fra både C++ og Java implementasjoner.



Figur 2.11: Oversikt over Symbian OS arkitektur.

Symbian ligger langt fremme innen implementasjon av nyere kommunikasjonsspesifikasjoner, og inkluderer en TCP/IP stakk for både IPv4 og IPv6. Personal Area Network (PAN) er også inkludert, og fungerer over IrDA IR, Bluetooth og USB. Sikkerhet er støttet gjennom HTTPS, WTLS, SSL og TLS. Meldingshåndtering er støttet gjennom SMS, EMS og MMS.

Symbian OS har implementert god støtte for Bluetooth via C++ og Java. Bluetooth implementasjonen for C++ er basert på HCI via UART, slik at den skal fungere mot nesten enhver annen implementasjon av Bluetooth. I første omgang er det kun spesifisert støtte for L2CAP og RFCOMM protokollene, og Serial Port Profile i Bluetooth spesifikasjonen. For C++ er det også støtte for OBEX (Object Exchange) protokollen. Som med andre implementasjoner, fokuseres det på å støtte de mer grunnleggende profilene. Deretter er det opp til utviklerne å implementere ytterligere profiler. Java implementasjonen for Bluetooth følger JSR-82 API'et som spesifisert, men OBEX er ikke støttet. Ellers følger Java implementasjonen J2ME CLDC 1.0 og MIDP 2.0, i tillegg til CDC og Personal Profile for større enheter (se eget avsnitt om Java).

Som tidligere beskrevet, er brukergrensesnittet i Symbian separert i en egen del slik at det kan forandres uavhengig av kjernen. Dette er med hensyn på lisens likevel stort sett forbeholdt de som produserer enhetene. Støtten for applikasjoner i C++ er likevel god, med muligheter for å styre hele skjermen, og en rekke ferdige kontroller for å lage menyer. Disse er likevel ofte basert på GUI-implementasjonen som finnes i Symbian OS på mobiltelefonene, og programmer må ofte redesignes ut i fra hvilken GUI-implementasjon som finnes. Java implementasjonen støtter MIDP 2.0, som inneholder en rekke ferdige elementer som menyer, og gir i tillegg tilgang til hele skjermen om nødvendig.

2.2.2 Pocket PC Phone Edition 2003 og Smartphone 2003

Pocket PC Phone Edition og Smartphone er to OS utviklet av Microsoft for bruk i mobile enheter. Pocket PC Phone Edition er en utvidet versjon av Pocket PC, og er beregnet på PDA'er som skal tilby mobiltelefonfunksjonalitet. Smartphone er derimot myntet spesifikt på mobiltelefoner. Både Pocket PC og Smartphone bygger på Windows CE, som er et OS for håndholdte PC'er. Av den grunn er det store likhetstrekk mellom Pocket PC, Smartphone og Windows CE. Utviklingsverktøyene som benyttes er eMbedded Visual .NET og eMbedded Visual C++. Vi har sett bort fra eldre versjoner av disse OS'ene fordi de ikke støtter viktige nettverksteknologier som Bluetooth og WLAN. Verken Pocket PC Phone Edition 2003 eller Smartphone 2003 kommer med innebygget støtte for Java eller andre programmeringsspråk, men er bakoverkompatible med eMbedded Visual Basic.

Utviklingsverktøyene for Pocket PC Phone Edition og Smartphone gir full tilgang til alle deler av OS'et gjennom en rekke API'er. Nettverks-API'ene inkluderer Bluetooth, WAP, SMS, GPRS og Winsock. I tillegg har Pocket PC støtte for DirectPlay API'et, som forenkler prosessen med å implementere nettverkskommunikasjon i applikasjoner. Pocket PC kan også fungere som en HTTP server, noe som kan være nyttig ved fjernstyring av enheten. En bør også merke seg at en mobiltelefon ikke nødvendigvis er utstyrt med Bluetooth selv om OS'et støtter det.

Brukergrensesnittet er fullstendig konfigurerbart. Og skulle det være nødvendig, er det mulig å få full tilgang til rammebufferet (VFB) gjennom et Gaming API (GAPI). Det er en del forskjeller mellom brukergrensesnittene på en Pocket PC og en Smartphone, slik at programmer laget for Pocket PC ikke nødvendigvis fungerer på Smartphone og omvendt. Ved å benytte teknikker som betinget kompilering og Runtime Version Checking kan en sikre at programmer vil støtte begge plattformene.

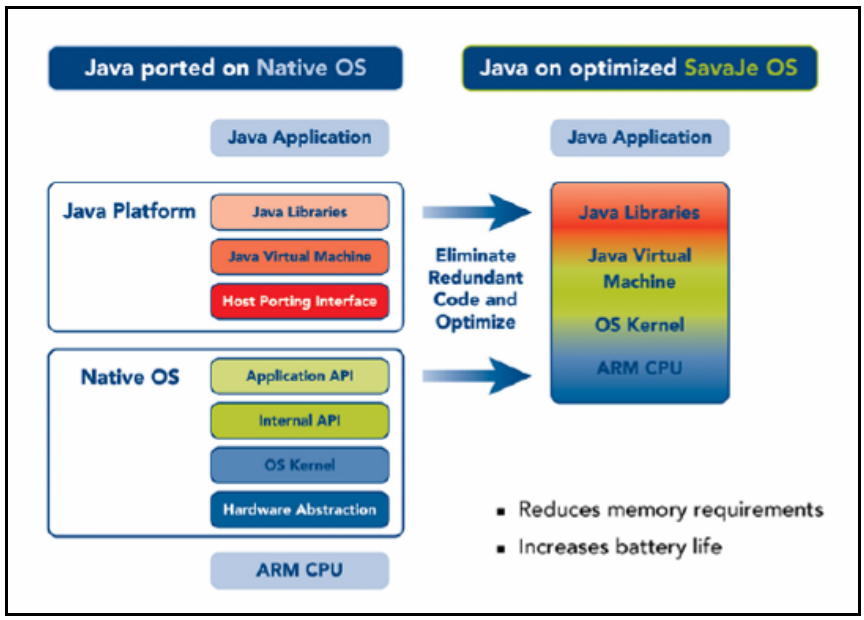
Innlasting av programvare kan gjøres på flere måter, blant annet ved å laste ned fra en web side med Pocket Internet Explorer, ved å sende en link med e-post, eller ved å sende filen som vedlegg til en e-post. På Smartphone baserte enheter må det i tillegg tas hensyn til at programmet i de fleste tilfeller må være sertifisert for at det skal kunne installeres og kjøres.

Filsystemet i Windows CE arver fra filsystemet på PC. Filstrukturen er derfor mer komplett i forhold til andre mobile OS, og en har stor frihet med hensyn på filstørrelse, filnavn og filtype på lik linje med et PC-system.

Mer om Pocket PC Phone Edition 2003 og Smartphone 2003 finnes i "Microsoft Pocket PC 2003 SDK" [15] og "Microsoft Smartphone 2003 SDK" [16].

2.2.3 SavaJe OS 2.1

SavaJe OS er utviklet av SavaJe Technologies som den ultimate Java plattformen for neste generasjons mobiltelefoner. Dette har de oppnådd ved å integrere en Java VM i selve OS'et. Den tradisjonelle løsningen har vært å legge en Java VM oppå OS'et. SavaJe OS vil derfor kunne kjøre Java applikasjoner raskere fordi kallene ikke må oversettes fra VM'en til OS'et. SavaJe OS er kompatibel med alle applikasjoner som er laget for J2ME MIDP og Personal Basis Profile.



Figur 2.12: Java arkitektur i SavaJe OS i forhold til andre OS. [25]

SavaJe OS benytter naturligvis Java som API, og API støtten inkluderer Bluetooth, WAP, SMS og GPRS. Filsystemet er et standard PC filsystem, og overføring av programmer og data til en mobiltelefon som kjører SavaJe OS, kan skje OTA.

Selv om SavaJe OS er en ideell plattform for utvikling i Java, er det likevel ennå ikke realistisk å utvikle applikasjoner utelukkende for SavaJe OS. Grunnen til dette er at SavaJe OS først vil bli implementert i mobile enheter i slutten av 2004. Derimot vil SavaJe kunne bli en sterk konkurrent på mobilmarkedet i fremtiden. Dersom en utvikler applikasjoner for Java, vil disse med stor sannsynlighet også kjøre på SavaJe OS.

Mer om SavaJe OS og SavaJe Technologies finnes i plattformoversikten for SavaJe OS. [25]

2.2.4 Palm OS 5

Palm OS er utviklet av PalmSource. Det mest brukte utviklingsverktøyet for Palm OS er CodeWarrior IDE (Integrated Development Environment). Programmeringsspråket som benyttes av CodeWarrior IDE er C. Palm OS har ikke innebygget støtte for andre programmeringsspråk, inkludert Java, men det var tidligere mulig å laste ned en Java VM for Palm OS fra Sun sine nettsider. Denne er nå tatt bort fra nettsidene til Sun, og den eneste muligheten for å kjøre Java på Palm OS er å finne en tredjeparts Java VM på Internett. Støtte for å kompilere Visual Basic og Visual Basic .NET programmer til Palm OS er tilgjengelig fra utviklingsverktøyet AppForge Crossfire [27].

Palm OS 5 har støtte for Bluetooth gjennom API'er som tilbyr kommunikasjon over sockets, emulering av seriell port, og Exchange støtte. Palm OS tilbyr også TCP/IP og UDP/IP nettverkstjenester via et socket API. Et API for SMS er også inkludert. Derimot har ikke Palm OS API'er for GPRS eller WAP. Det er opp til den enkelte produsent å inkludere GPRS og WAP støtte i sine mobile enheter.

Fordi Palm OS baserte enheter har begrenset minne- og lagringskapasitet, og fordi synkronisering med PC'er skal skje effektivt, benytter ikke Palm OS et tradisjonelt filsystem. Isteden lagres data i såkalte "records", som grupperes i databaser. En database i Palm OS filsystemet tilsvarer en fil i et tradisjonelt filsystem.

Palm OS 5 støtter opptil 16 bit farger, mens GUI'et er begrenset til 8 bit. GUI'et programmeres ved hjelp av strukturer i C som representerer de forskjellige GUI elementene. Ved hjelp av en Form API kan en også implementere sine egne GUI objekter, såkalte Gadgets.

Mer informasjon finnes i SDK'en for Palm OS 5. [17]

2.2.5 MontaVista Linux CEE 3.1

MontaVista Linux CEE[1] er utviklet av MontaVista Software, og er som navnet tilsier, et Linux basert OS. De største fordelene med MontaVista Linux CEE er at det er basert på åpen kildekode, og støtter alle standard Linux API'er. Utviklingsmiljøet for MontaVista Linux CEE er primært Red Hat Linux 7.3 eller 9.0, men med VMWare 3.0, er det også mulig å benytte Windows NT eller Windows 2000.

En standard implementasjon av MontaVista Linux CEE har ikke støtte for Bluetooth enheter. Derimot er det opp til hver enkelt mobilprodusent å utvide OS'et for å legge til støtte for dette. Siden MontaVista Linux CEE bygger på Linux, og er open source, er det i og for seg ikke et stort problem for mobilprodusenter å utvide funksjonaliteten til OS'et.

[1] MontaVista Linux Consumer Electronics Edition er spesielt myntet på små enheter som mobiltelefoner.

MontaVista Linux CEE inkluderer, i likhet med Palm OS, heller ikke en Java VM. Derimot er en Java VM tilgjengelig for nedlasting [24].

Svært få mobiltelefoner baserer seg per i dag på MontaVista Linux CEE. Dette, i tillegg til at Bluetooth og Java ikke er standard i MontaVista Linux CEE, gjør OS'et til et lite egnet alternativ.

Mer informasjon om MontaVista Linux CEE 3.1 finnes i dokumentet "Montavista Linux Consumer Electronics Edition 3.1". [23]

2.2.6 Brew

Brew plattformen blir utviklet av Qualcomm [33], og kom første gang på markedet i januar 2001. Den er en svært tynn og åpen applikasjonsplattform, som lett skal kunne inkluderes på alt fra enkle til mer avanserte mobiltelefoner. Ved å tilby mobilprodusenter et godt utviklet implementasjonssystem ønsker Qualcomm at så mange mobiltelefoner som mulig skal implementere plattformen. Totalt ligger de fleste implementasjoner på rundt 150 kB. En rekke selskaper har integrert plattformen på sine enheter [34]. Blant annet har Nokia, Siemens, Samsung og Panasonic mobiltelefoner med Brew.

Brew plattformen fokuseres like mye mot selve distribusjonen av programvare, som mot utviklingen av applikasjoner. Den blir derfor delt i to hoveddeler bestående av Brew SDK og BDS (Brew Distribution System). Ved hjelp av Brew SDK'en kan programmer skrives i C/C++, og er delvis uavhengige av underliggende hardware. Ved at flere virtuelle maskiner og andre plattformer er blitt designet for å kjøre over Brew, har Brew i dag støtte for en rekke programmeringsspråk og plattformer.

Ved hjelp av BDS kan alle typer filer lastes ned ved hjelp av OTA teknologi, inkludert utvidelser for å kjøre andre plattformer. I dag støttes blant annet kjøring Java-, Flash- og XML-filer. Qualcomm hevder med dette å ha en bedre løsning enn for eksempel rene Java baserte mobiltelefoner, da en ved hjelp av Brew vil kunne laste ned en ny VM når nye spesifikasjoner dukker opp.

Selve distribusjonen av programmer er altså et av grunnmålene med Brew. Dette har blitt implementert på den måten at all kode må signeres og verifiseres før programmet legges på spesialiserte distribusjonsservere hos gjeldende mobiloperatør. Dermed har en full kontroll over programvare og fakturering.

Ettersom Brew er en svært liten plattform, støttes få høyere lags protokoller i grunnmodulen. Det er derfor lite støtte for WAP, Web Services eller lignende protokoller. Det er derimot implementert støtte for sockets og serieport kommunikasjon. Ved hjelp av tilleggsbiblioteker som følger Brew standarden, vil en derfor kunne støtte høyere lags protokoller. Disse bibliotekene vil også fungere på tvers av mobiltelefoner da Brew skal være i stand til å kjøre all standardisert kode. I tillegg finnes muligheten for å benytte virtuelle maskiner som for eksempel Java VM, som har innebygd støtte for flere protokoller. I dag finnes det kun implementasjoner av J2ME MIDP 1.0, men det er forventet at nyere versjoner vil komme. Det er ingen direkte støtte for Bluetooth i Brew, men også her vil en kunne se implementasjoner som avhengig av implementasjonene på mobiltelefonene, vil kunne benytte seriekommunikasjon på lavere lag.

Grafisk har en full kontroll over skjermen på mobiltelefonen, og enkelte produsenter av mobiltelefoner velger å designe hele brukergrensesnittet gjennom Brew. Likevel har plattformen støtte for å avbryte og pause programmer ved innkommende samtaler og meldinger. En del av de sikkerhetsaspektene som ligger i en såpass lavnivå implementasjon, håndteres ved at det ikke skal være mulig å distribuere utestet kode, da alle programmer må distribueres gjennom kjente operatører. En rekke ferdige moduler for grafiske elementer, som menyer og lignende, er også støttet i Brew, noe som gjør det lettere å designe programmer på tvers av ulike mobiltelefoners design. Brew er likevel en liten plattform med lavnivå tilgang til ressurser, istedenfor å tilby standardiserte høynivå API'er mot disse. Programmer som utnytter lavnivåressurser kan derfor ikke sies å være helt plattformuavhengige, da disse som regel må aksesseres ulikt fra mobiltelefon til mobiltelefon.

2.2.7 Mophun

Mophun ble utviklet av Synergenix Interactive i Sverige i 1999 som et svar på den økende interessen for utvikling av spill til mobiltelefoner. Gjennom en egen utviklingsplattform, og "engine", benyttes Mophun i dag i en rekke mobiltelefoner for å gi muligheter for økt ytelse til grafikk og lignende. Flere mobiltelefonleverandører benytter i dag Mophun-systemet i stedet for å utvikle sine egne systemer. Mobiltelefonleverandører som i dag satser på Mophun som spillplattform, er Sony Ericsson, Nokia, Siemens og Motorola.

Mophun baserer seg på en mindre "run time engine" som fungerer som en virtuell maskin for programvaren. Flere av funksjonene som API'et tilbyr, oversettes nesten uten forandring til interne prosessorkall. Mophun har i tester [39] vist en ytelse på flere hundre prosent mer enn andre plattformer. En klar grunn til dette er at programmering mot API'ene gjøres i form av C eller C++, noe som gjør det enklere å oversette kallene.

Selv om Mophun blir regnet som en ren spillplattform, kan den i teorien også benyttes til andre applikasjoner. Spill setter ofte svært høye krav til ressurser, og Mophun har allerede, i de fleste implementasjoner, god støtte for de ressursene en mobiltelefon har å tilby. Dette gjelder utnyttelse av både Bluetooth og GPRS teknologi. Innen Bluetooth er det kun støtte for "Serial Port" profilen.

Ettersom mobil programvare ofte er liten, har det gitt muligheter for at svært små firmaer og enkeltpersoner kan bidra i dette markedet. De fleste programmeringsplattformer satser derfor også mye på å tilby et distribusjonsnett for utviklere. Dette gjelder også Mophun, som har et strengt oppbygd distribusjonssystem hvor hvert produkt må sertifiseres hos Synergenix. Selve overføringen til mobiltelefonen gjøres, som ved andre systemer, med WAP eller MMS. For å unngå størrelsesbegrensningene som ligger i WAP gateway'ene, kan programfilene deles opp i mindre deler som nummereres og overføres en etter en. Når riktig antall filer er mottatt, setter Mophun sammen programmet, og det er klart for å kjøres på mobiltelefonen.

Ettersom Mophun egentlig er en modul som kjøres over det OS'et som ligger på mobilen, kan også selve Mophun motoren erstattes på en del mobiltelefoner. Dette gjelder mobiltelefoner som har et OS som også gir muligheter for innlasting av ny programvare.

Krav om sertifisering av et hvert Mophun program er også satt med tanke på sikkerhet. En programvareplattform som er tett knyttet til hardware og OS kan potensielt utgjøre en sikkerhetsfare. Gjennom sertifisering sikrer en til dels at usjekket programvare ikke skal kunne kjøres på mobiltelefonen.

Grafisk har selvsagt Mophun store muligheter. Det gis full mulighet til å kontrollere hele skjermen, selv om telefonspesifikke hendelser, som SMS og telefonsamtaler, vil kunne overstyre når disse inntreffer. Ettersom systemet skal være raskt og fleksibelt, er det ikke lagt til ferdige grafiske elementer som lister og menyer, men det er isteden gitt bedre støtte for sprites, tiles, og andre typiske spillelementer. Dette gir ikke samme mulighetene til et felles gjenkjennelig brukergrensesnitt på tvers av programmer og ulike mobiltelefoner, som vi kjenner fra andre utviklingsmiljø. Det er derfor mange regner plattformer som denne som ikke helt hardwareuavhengig. Med mindre en bygger opp et helt dynamisk grensesnitt på egen hånd, og ikke benytter telefonspesifikke API'er, må en stort sett gjøre en del forandringer og en ny kompilering for hver mobiltelefontype en utvikler programmet til.

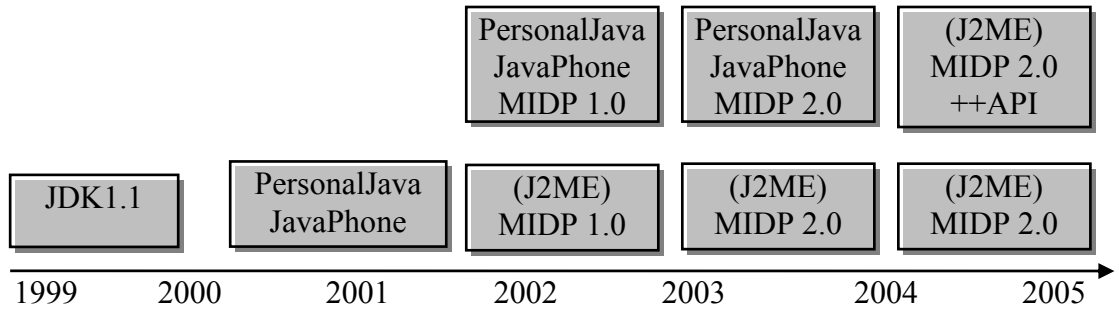
Det eksisterer også andre applikasjonsplattformer som har de samme målene som Mophun. To andre aktører er TTPCom's "Wireless Graphics Engine" og In-Fusio's "Execution Engine". Førstnevnte har gått inn som en del av Mophun plattformen, og de fleste andre implementeres ikke lenger på nye mobiltelefoner, så vi går derfor ikke videre inn på disse.

Mer informasjon om Mophun finnes på Synergenix Interactives hjemmesider [40].

2.2.8 Java 2 Micro Edition

I motsetning til applikasjoner laget for å kjøre direkte på mobiltelefonens eget operativsystem, kjører Java programmer fullstendig på en virtuell maskin installert på mobilenheten. I dag leveres denne som regel inkludert på mobiltelefonen, men de større og mer avanserte operativsystemene gir også mulighet for at en virtuell maskin kan legges inn og oppdateres i ettertid. Dette vil si at dersom den virtuelle maskinen følger de standarder som er satt, vil applikasjoner kunne kjøres på tvers av hardware plattformer. På den andre siden krever den en større kjøreplattform, ettersom koden som kjøres ikke bare skal beskytte mot ødeleggende programvare, men også oversette Java bytekode til maskinkode. Dette fører igjen til at ytelsen på Java programmer er nokså lav.

Java standarden er kjent som den mest utviklede plattformuavhengige løsningen for utvikling av programvare til stasjonære PC'er, og blir utviklet av Sun Microsystems. Ett av målene til Sun de siste årene, har vært å tilby en Java versjon for enhver plattform. For mobile enheter har det eksistert en rekke versjoner de siste årene.

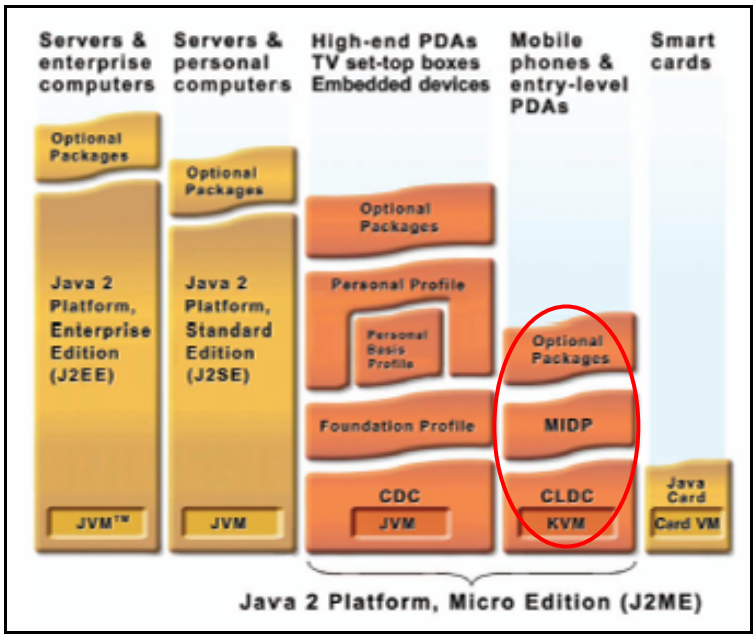


Figur 2.13: Tidslinje over Java implementasjoner benyttet på mobiltelefoner.

De første mobiltelefonene benyttet en forenklet versjon av implementasjonen som finnes på stasjonære PC'er. I 2000 bestemte Sun seg for å forbedre dette, og gav ut spesifikasjonene til et eget applikasjonsmiljø for mobiltelefoner bestående av PersonalJava og et JavaPhone API. Dette støttet en rekke mobiltelefonspesifikke funksjoner, og krevde langt mindre minne enn standardversjonen. Ett år etter fant de derimot ut at de ønsket å lage en fastere inndeling av Java versjonene, og delte de inn i Java 2 Enterprise Edition (J2EE), Java 2 Standard Edition (J2SE) og Java 2 Micro Edition (J2ME). Førstnevnte er for servere, neste for standard PC'er, og siste for enheter med lavere ytelse.

J2ME kan igjen deles inn i flere versjoner beregnet på ulike enheter. Ved å tilby ulike moduler for å støtte forskjellig hardware, grafiske grensesnitt og ressurser, har J2ME i dag en inndeling som vist i Figur 2.14. Connected Limited Device Configuration (CLDC) og Mobile Information Device Profile (MIDP) er modulene som er beregnet for mobiltelefoner. Sammen med KVM[1] kreves det rundt 160-512 kB minne for å støtte denne konfigurasjonen.

[1] Virtuell maskin for små enheter, deriblant mobiltelefoner.



Figur 2.14: Oversikt over Java versjoner. [45]

En viss grunnfunksjonalitet må støttes dersom en implementasjon av for eksempel MIDP skal være standard. Den første versjonen, MIDP 1.0, hadde svært begrensede muligheter for grafiske elementer. Dette førte til at det ble lagt til andre plattformer, eller egne proprietære grafiske Java API'er, i de fleste mobiler med Java støtte. I 2003 kom derfor en ny spesifikasjon, MIDP 2.0 (JSR-118), som standardiserte langt bedre støtte for grafikk gjennom et eget Game API, som minner mye om Pocket PC sitt GAPI. I tillegg spesifiseres det kontinuerlig en rekke valgfrie API'er som utvider funksjonaliteten ut over MIDP. Disse er definert gjennom Java Community Process (JCP), og hvert nye API, profil eller konfigurasjon spesifiseres som en Java Specification Request (JSR).

Som standard støttes HTTP kommunikasjon i MIIDP 1.0/CLDC 1.0, mens TCP sockets/secure sockets, UDP datagram og Web Services støttes gjennom valgfrie API'er i MIDP 2.0/CLDC 1.0. Grunnen til dette er at for eksempel WAP protokollen ikke krever en ekte TCP/IP tilkobling på lavere lag. Dermed er det mulig at noen mobiltelefoner ikke direkte støtter TCP sockets, og dette må da holdes utenom standardspesifikasjonen for å oppnå plattformuavhengighet. Det samme gjelder Bluetooth støtten i Java. Gjennom et standardisert Bluetooth API, JSR-82 tilbys støtte for de grunnleggende profilene i Bluetooth standarden. I dag støttes L2CAP og RFCOMM protokollene, Service Discovery Profile, Serial Port Profile og OBEX. Som ellers i andre Bluetooth implementasjoner, ønsker en kun å støtte de laveste protokollene, og heller la applikasjoner utvide disse dersom det kreves. I CLDC har Sun spesifisert et eget kommunikasjons-API, General Connection Framework (GCF), som alle kommunikasjonsteknologier aksesseres gjennom. På denne måten vil applikasjoner programmeres rimelig likt, uavhengig av hvilket kommunikasjonsmedie som benyttes.

Plattformuavhengighet på mobiltelefoner er svært utfordrende, da de kommer i alle typer, størrelser og konfigurasjoner. I MIDP er det derfor spesifisert en rekke minimumskrav. Blant annet kreves en skjerm på minimum 96x54 piksler. For å likevel kunne utnytte ressurser ut over dette, er de fleste grafiske komponenter som er spesifisert i MIDP, designet dynamiske slik at de utnytter de ressursene som finnes. Som standard kreves også en eller annen form for inn-enhet, og spesifikasjoner finnes for standard keypad, QWERTY tastatur og touch screen.

Også innen lagring av data krever ikke Java at det underliggende OS'et støtter et større filsystem eller lignende. Spesifikasjonen krever at terminalen har minimum 8 kB med minne som ikke slettes ved omstart eller tap av strøm. I Java aksesseres dette via et eget API, som gir muligheter for å lagre data fra applikasjoner.

Sun var tidlig ute med å forstå at OTA innlasting av programvare var svært viktig. Spesielt over WAP er det en stor fordel å kunne laste ned programvare. Problemet med dette er at WAP er beregnet for små filer. Selv på mobile enheter blir applikasjoner ofte større enn det WAP servere tillater. Sun introduserte i 2001 en OTA teknikk som ved hjelp av en beskrivende fil, som lastes ned før selve programmet, gjør WAP serverne i stand til å tillate de større filene.

Svært mange mobiltelefoner i dag kommer med en VM som følger Sun sine spesifikasjoner. Dessverre har de ulike versjonene som har vært utgitt, og de mange proprietære API'ene laget av mobilprodusentene, gjort at plattformuavhengigheten ikke er like god som den burde vært. MIDP 2.0, som nå begynner å komme på enkelte mobiltelefoner, forventes å forbedre dette.

2.2.9 Vurdering av mobile applikasjonsplattformer

Vi har nå presentert de største plattformløsningene på mobilmarkedet, men det finnes også andre plattformer. Felles for de vi ikke har presentert her er at de er spesialiserte plattformer rettet inn mot et snevert antall produkter, og i de fleste tilfeller spesiell hardware. Flere av disse ikke-standardiserte plattformene er derimot støttet av applikasjonsplattformene Java, Brew eller Mophun, så vi vil ikke gå nærmere inn på disse, annet enn det som er evaluert under kapitlene om Java, Brew og Mophun.

En konsekvens av å produsere applikasjoner for spesialiserte plattformer, er at applikasjonene må designes og utvikles for hver enkelt plattform. Hvis en mobilapplikasjon, som er beregnet for et stort marked, skal støtte slike plattformer, betyr det at det må produseres en mengde utgaver av applikasjonen. Dette vil være svært ulønnsomt. Vi mener isteden at den beste løsningen vil være å lage et plattformuavhengig produkt. En slik løsning vil gi støtte for de fleste nyere mobiltelefoner, og vil nå ut til en stor målgruppe, samtidig som den krever kun én utgave av applikasjonen. Vel å merke så lenge det kun blir benyttet standard API'er. Benyttes et ikke-standard API, vil ikke applikasjonen kjøre hvis ikke plattformen har støtte for dette, og da er hele vitsen med plattformuavhengighet borte. Plattformuavhengighet er også til stor hjelp for brukere som prøver å finne riktig utgave av programmet.

Tabell 2.2: Oversikt over mobile OS og støttede programmeringsspråk.

OS / Språk	Symbian OS v7.0s	Palm OS 5	Pocket PC Phone Edition 2003	Smartphone 2003	SavaJe OS 2.1	MontaVista Linux CEE 3.1	Andre
C	X	X	X	X		X	X
C++	X		X	X		X	X
J2ME	X	X[1]	X[1]	X[1]	X	X[1]	X[2]
VB	X[3]	X[3]	X	X			
VB .NET	X[3]	X[3]	X	X			
C#			X	X			
OPL	X						

I gjennomgangen av de mobile applikasjonsplattformene legger vi merke til at de mest utbredte programmeringsspråkene blant de mobile OS'ene er C, C++ og Java. Språkene C# og OPL støttes av få applikasjonsplattformer, og er dermed et dårlig valg dersom en ønsker å nå ut til et bredt publikum med et produkt. De to andre språkene, Visual Basic og Visual Basic .NET, kjører i likhet med Java oppå en virtuell maskin. Denne krever ekstra lagringsplass på mobiltelefonen, og dette er også en årsak til at mange mobiltelefoner ikke leveres med ferdig installerte virtuelle maskiner, noe som også kommer frem i Tabell 2.2.

Siden Visual Basic i hovedsak er støttet av kun Pocket PC Phone Edition og Smartphone, se Tabell 2.2, hvis en ikke har benyttet et spesielt verktøy under utviklingen, og i tillegg er faset ut til fordel for Visual Basic .NET, mener vi at Visual Basic ikke har noen fremtid på det mobile markedet.

Mophun og Brew er to applikasjonsplattformer som mer enn noe minner om de virtuelle maskinene vi kjenner fra Java og Visual Basic .NET. Den store forskjellen er at Mophun og Brew primært baserer seg på C++, noe som gjør programmer skrevet for disse applikasjonsplattformene tilnærmet like raske som maskinkode. Dessverre må Mophun programmer kompileres på ny for hver hardware plattform de skal kjøres på. Brew programmer skal være plattformuavhengige så lenge en benytter standard Brew API'er, men så snart en går bort fra disse, vil Brew programmer stilles i samme bås som Mophun programmer. Derimot inkluderer noen implementasjoner av Brew også en Java VM. Mobiltelefoner som kjører proprietære OS uten en inkludert Java VM, men som inkluderer Brew med en tilhørende Java VM, har dermed likevel en mulighet for å kjøre Java programmer. Mophun er i hovedsak en spillplattform, men også andre typer applikasjoner kan dra nytte av Mophuns høye ytelse. Derimot gjør mangelen på Java og høynivå GUI elementer Mophun til et mer tungvint utviklingsmiljø enn Brew.

Både Mophun og Brew har sine egne distribusjonssystemer som kan forenkle OTA nedlasting av programmer for brukerne. Til sammenligning har ikke Java noe sentralt distribusjonssystem, men har derimot muligheter for OTA innlasting av programvare.

[1] Støttes via selvstendig nedlastbare tredjeparts Java VM'er.
[2] Mange proprietære mobile OS samt enkelte implementasjoner av Brew leveres med en innebygd Java VM.
[3] For utvikling i disse språkene kreves AppForge Crossfire [27].

C og C++ er også såkalte plattformuavhengige språk, men et program skrevet i ett av disse språkene er ikke plattformuavhengig, og en må lage versjoner av det for hvert OS det skal kjøres på. Derimot har et program skrevet i C eller C++ den fordelen at det kjører rett på hardware. Disse språkene er dermed raske, noe som er en stor fordel for tunge programmer og spill. Mophun er et godt eksempel på akkurat dette. Siden disse språkene kommuniserer direkte med hardware, får en også full kontroll over alle funksjoner på mobiltelefonen, noe som ofte savnes i andre språk. En kan også si at andre språks funksjonalitet er mer eller mindre hemmet av å ikke ha full tilgang til funksjonalitet på et hardware-nivå. Derimot knyttes det også fordeler til dette. Det vil som regel bli enklere, og dermed raskere og billigere, å utvikle programvare fordi mye funksjonalitet ligger i standardiserte API'er.

Som det kommer frem av Tabell 2.2, kan SavaJe og MontaVista ses på som outsidere på mobiltelefonmarkedet. I tilfellet SavaJe er dette nokså klart i og med at ingen mobile enheter, som kjører denne plattformen, ennå er kommet på markedet. Det er likevel sannsynlig at SavaJe vil bli en stor aktør i fremtiden med sin unike Java implementasjon. Og det kan derfor være strategisk å utvikle programmer i dag, som vil kjøre på SavaJe i morgen. MontaVista kan på sin side skilte med implementasjoner på noen få mobiltelefoner. Disse er derimot for få til å utgjøre et betydelig marked. Spikeren i kisten for MontaVista er manglende Bluetooth implementasjon og ingen inkludert Java VM. Av disse årsaker går vi ikke videre inn på verken MontaVista eller SavaJe i denne rapporten.

Det er nå klart at markedet er spekket med ulike implementasjoner av ymse mobile applikasjonsplattformer. Tross mange forsøk på å skape den ultimate plattformen, ser mangfoldet bare ut til å øke. Dette er uten tvil ikke i en applikasjonsutviklers interesse, og det er tilnærmet umulig å støtte alle plattformer samtidig. Dermed blir utvikleren tvunget til å velge den gruppen av brukere som vil bringe inn størst inntekt. Dette innebærer å velge den mest utbredte plattformen, eventuelt skrive applikasjonen i et språk som støtter plattformuavhengighet, gitt at alle nødvendige teknologier er understøttet.

Vi har allerede vurdert de ulike teknologiene for trådløs kommunikasjon opp mot hverandre. Bluetooth virker å være en egnet kommunikasjonsform for små enheter, som mobiltelefoner, men krever at plattformen har egnede API'er. WLAN er bedre egnet for enheter som stiller høyere krav til hastighet, men samtidig kan tolerere høyere strømforbruk. Uansett valg av kommunikasjonsteknologi for lang rekkevidde, kan det nevnes at alle plattformer i dag har mulighet for kommunikasjon ved hjelp av SMS, mens socket støtte er nødvendig for kommunikasjon over GPRS eller GSM. Mens lavnivå språk, som C++, støtter alt som finnes av teknologi på en mobiltelefon, finnes ingen standardiserte API'er. En må derfor passe på at alle API'er som benyttes, støttes av alle mobiltelefoner som programmet skal kjøre på, noe de sjelden gjør. Høyere lags protokoller må derfor implementeres for hånd i lavnivå språk. Høynivå språk, som J2ME, spesifiserer standardiserte API'er, og omgår derfor disse problemene.

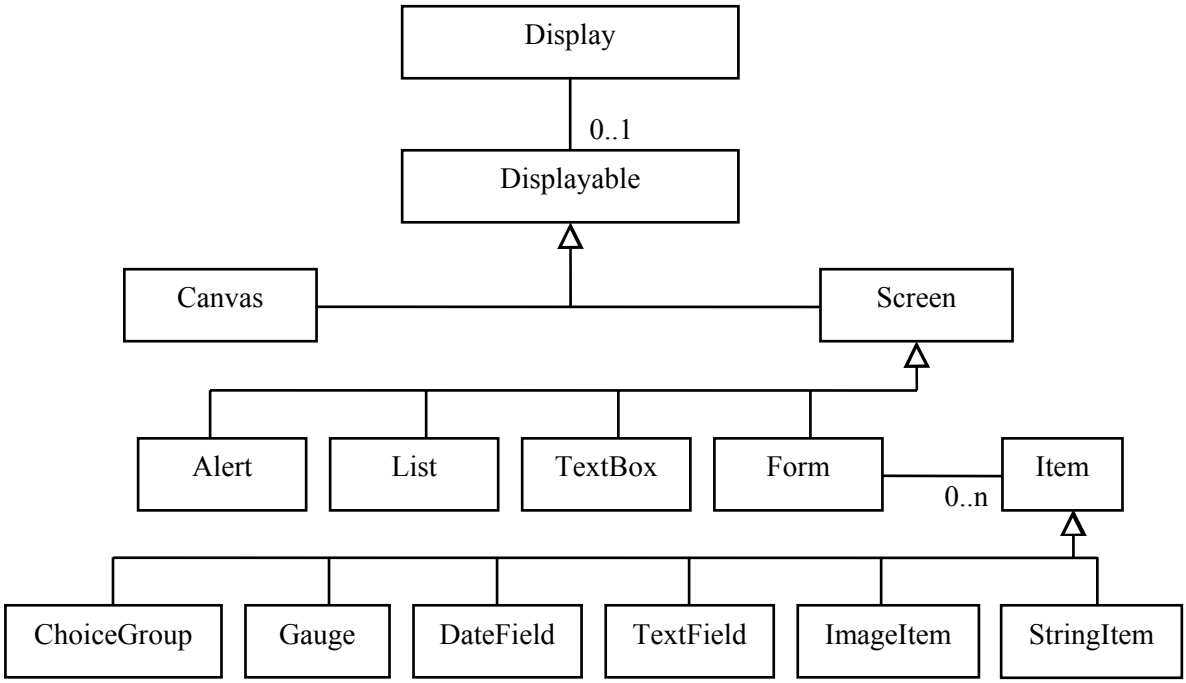
Som Tabell 2.2 har vist, er J2ME det aller mest utbredte programmeringsspråket, hvis en ser bort fra at mange av plattformene ikke leveres med en Java VM installert. Visual Basic .NET har mange likhetstrekk med J2ME, og begge inkluderer stort sett alle viktige API'er, men relativt få mobiltelefoner kan i dag kjøre Visual Basic .NET applikasjoner. C og C++ er de eneste språkene som kan konkurrere med J2ME i utbredelse, men lavnivå språk taper i forhold til J2ME fordi plattformuavhengighet nær sagt er en nødvendighet. Og med tanke på støtte for sikkerhet og nettverksteknologier som Bluetooth og TCP socket, spesifiserer J2ME API'er for dette. J2ME har derfor ikke noen store ulemper i forhold til lavnivå språk, som C og C++. Og som vi skal se i Kapittel 2.3 om brukergrensesnitt, mangler ikke J2ME noe på denne fronten heller.

J2ME har helt klart store fordeler overfor Brew, Mophun, og andre ikke-standardiserte applikasjonsplattformer. J2ME tilbyr plattformuavhengighet, og samtidig støttes alle teknologiene vi har bruk for. C++ støtter alle teknologier på et lavere nivå, men programmer skrevet i C++ er derimot ikke plattformuavhengige. I utbredelse kan ingen OS måle seg med J2ME, som i likhet med Brew og Mophun leveres på tvers av både åpne og proprietære OS.

2.3 Brukergrensesnitt

De fleste høynivå programmeringsspråk inneholder et sett med standard brukergrensesnitt (UI) komponenter for både GUI og input. Disse komponentene har som formål å tilby brukere av mobiltelefoner og andre enheter et konsistent grensesnitt på tvers av applikasjoner. I tillegg finnes vanligvis muligheter for å designe egne komponenter, noe som ofte benyttes i spillutvikling. Lavnivå programmeringsspråk har derimot sjelden slike innebygde komponenter, og UI'et må da programmeres fra bunnen av. Resultatet av dette kan bli at en applikasjon får et annerledes UI enn en annen applikasjon på den samme mobiltelefonen, og som [6] gir uttrykk for, er sjelden målet med en applikasjon å forvirre brukerne. I tillegg bryter dette med anbefalte retningslinjer som sier at mobile applikasjoner bør være enkle å lære og bruke [6].

J2ME plattformen er et godt eksempel på dette. J2ME MIDP applikasjoner inneholder et Displayable objekt som vises på skjermen, se Figur 2.15. Det finnes to typer Displayable objekter i MIDP [1]. Et lavnivå Canvas objekt, som tillater applikasjoner å håndtere grafikk og input. Og et høynivå Screen objekt, som omslutter komplette UI komponenter. Ved bruk av Screen objekter er det den innebygde programvaren i mobiltelefonen som sørger for å håndtere input og presentere skjermbildet i tråd med mobiltelefonens innebygde UI. Screen objekter anvender et høyt abstraksjonsnivå, og gir liten kontroll over utseende og brukervennlighet. MIDP applikasjoner kan eventuelt benytte kombinasjoner av Screen og Canvas objekter for bedre kontroll over utseende, og for å presentere et integrert UI, se Figur 2.15.



Figur 2.15: Eksempel på UI klassehierarki tatt fra Java MIDP 1.0. [1]

Nytt i MIDP 2.0 er klassen CustomItem, som har mye til felles med et Canvas objekt, men som benyttes i en Form. Også et nytt lavnivå objekt har funnet veien inn i MIDP 2.0 i form av et GameCanvas objekt. Fordelen med dette objektet er at det gir muligheter for å designe applikasjoner som skal kjøre i fullskjerm, men som navnet tilsier, er dette objektet mest av alt beregnet på spillutvikling. Canvas objekter bør uansett kun benyttes som en siste utvei fordi de krever nøye planlegging for å være portable. Isteden bør det benyttes høynivå objekter der dette er mulig [6].

UI'et på mobile enheter er svært avhengig av plattform og design. Hvilke knapper og andre input muligheter som finnes, hvordan disse er plassert på enheten, og hvordan plattformen håndterer disse når brukeren navigerer i applikasjoner og menysystemer. Det grafiske brukergrensesnittet (GUI) er avhengig av skjermens størrelse og fargedybde, og hvorvidt det er liste- eller ikonbasert. Minstekravene til UI på J2ME MIDP enheter er, som spesifisert av Sun, keypad, QWERTY tastatur, eller touch screen, og en 1 bit fargeskjerm med 96x54 pixler [6]. Bruk av Screen objekter i J2ME MIDP, eller tilsvarende objekter i andre programmeringsspråk, er den beste metoden for å oppnå plattformuavhengighet fordi plattformen selv håndterer GUI og input. For å beholde plattformuavhengigheten ved bruk av Canvas eller tilsvarende objekter, er det viktig å ikke anta tilstedeværelsen av ikke-standard knapper. I tillegg må skjermstørrelsen, og eventuelt fargedybden, til mobilenhetene detekteres og håndteres på en slik måte at applikasjonen vises riktig på alle enheter. [1]

Utfordringene ved å designe en god mobilapplikasjon er mange. Noen av de viktigste retningslinjene for et godt design, ifølge Little Springs Design [6], er som følger. Input av tekst på mobiltelefoner er upraktisk, og eventuelle alternativer er alltid å foretrekke. Ett av målene med applikasjonens design bør være å minimere antallet tastetrykk som er nødvendige for å utføre en handling. Mobilapplikasjoner bør være nøkternt oppbygde, slik at brukerne slipper å kaste bort tid på innhold som ikke er til nytte, slik som unødvendig informasjon eller bilder. Og sist, men ikke minst, hastighet har alt å si, og applikasjonen bør designes med tanke på dette. Hvis applikasjonen skal utføre tunge arbeidsoppgaver, må brukeren til enhver tid holdes oppdatert med hva som foregår, slik at det aldri vil virke som om applikasjonen henger.

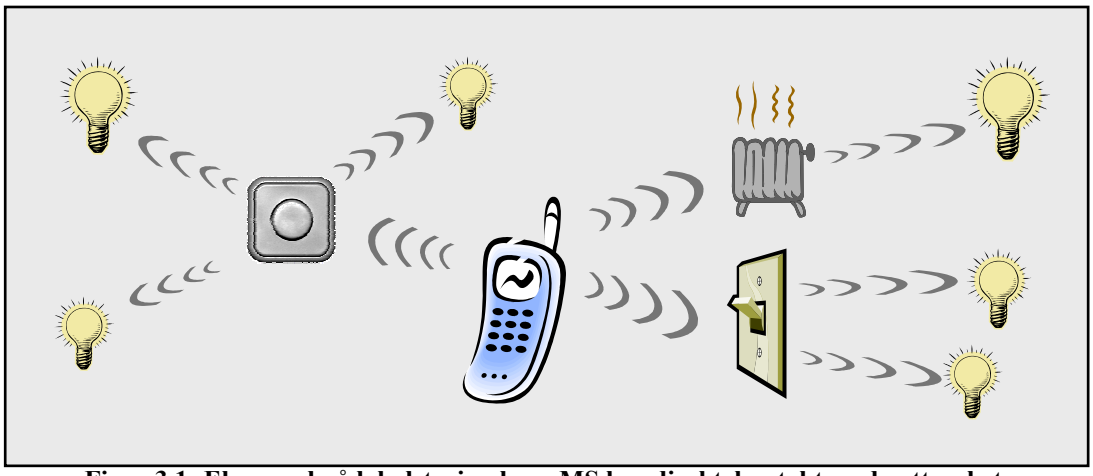
Dette betyr at et bra UI design skal være intuitivt og enkelt å navigere. Brukeren skal ikke kaste bort tid på å lære seg et nytt UI, og derfor er det viktig at UI'et til mobilapplikasjonen og mobiltelefonens OS minner så mye om hverandre som mulig. Med det plattformuavhengige språket J2ME, vil en applikasjon basert på en MIDP implementasjon med et Screen basert UI, få et brukergrensesnitt på linje med mobiltelefonens OS på tvers av alle plattformer.

3 Mobilapplikasjon for styring av SEA Bluetooth nett

3.1 SEA nett og problemstillinger ved mobiltelefonutvidelse

En kan dele styring av SEA sitt Bluetooth nett i lokal- og fjernstyring. Lokalstyring innebærer at mobilterminalen befinner seg så nære nettverket at den kan ha direktekontakt med enheter i nettverket. Fjernstyring vil på sin side bety at vi utnytter andre nettverk som et mellomledd, for å oppnå lengre rekkevidde. Lokalstyringen tilbyr derimot ofte raskere respons, og innbærer mindre eller ingen kostnader ved bruk da vi ikke trenger å gå innom andre nettverk.

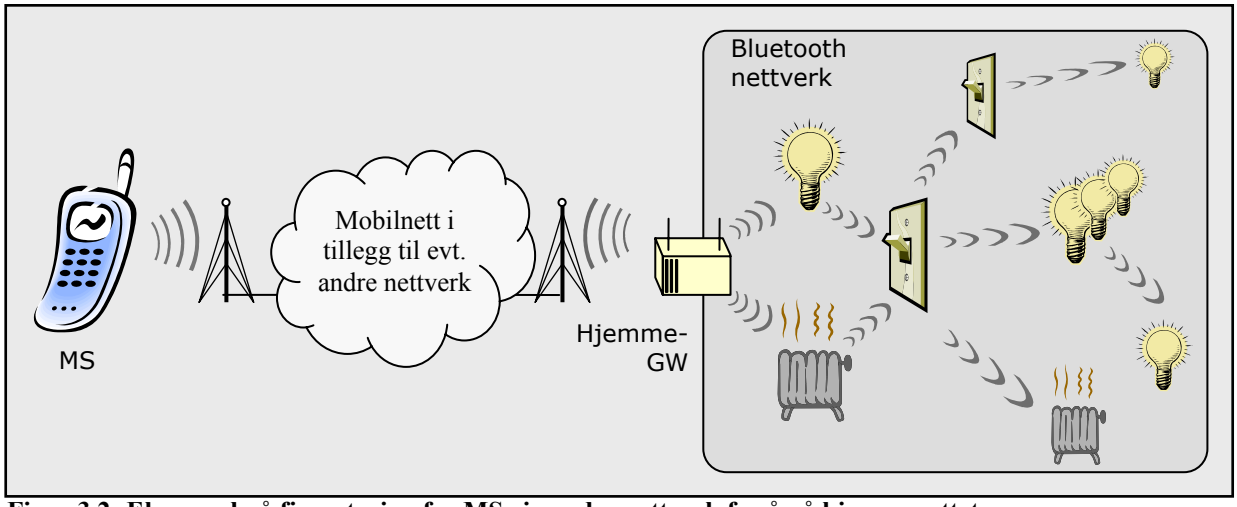
SEA sitt nett består i dag av små intelligente enheter som kommuniserer med hjelp av Bluetooth. Brytere, sensorer og termostater fungerer som styreenheter, og kontrollerer styrbare enheter som lys og ovner. Ved å implementere en protokoll som gir mulighet for oppbygging av rutingtabeller og videresending av datatrafikk, åpnes det for at en bryter kan styre enheter som ikke er innen direkte rekkevidde av bryteren.



Figur 3.1: Eksempel på lokalstyring hvor MS har direktekontakt med nettverket.

SEA ser på mulighetene til å introdusere en mobiltelefon i dette nettet, slik at en vil kunne ta kontakt med nettet eller enhetene direkte for å sette opp og styre disse. Dersom nettverket benytter en annen kommunikasjonsteknologi enn mobilterminalen, vil en også kunne benytte en GW som overfører meldinger ut på Bluetooth nettverket. En direkte tilknytning via Bluetooth er likevel å foretrekke da dette ikke krever en ekstra GW.

Som Figur 3.2 viser, ser SEA også på muligheten for å fjernstyre et slikt nett ved hjelp av mobilkommunikasjon. Dette må gjøres ved å introdusere en hjemme-GW i nettet som både har tilgang til Bluetooth og mobilnett, og kan videresende trafikk mellom disse. Dermed får en muligheter til å fjernstyre og overvåke hjemmenettet, og SEA-enheter vil kunne gi tilbakemeldinger og alarmer til en mobilterminal.



Figur 3.2: Eksempel på fjernstyring fra MS via andre nettverk for å nå hjemmenettet.

3.2 Tekniske krav til funksjonalitet

3.2.1 Krav til programvare og brukergrensesnitt

Mobiltelefoner er i dag allemannseie. Og samtidig som de etter hvert er blitt svært avanserte, bør brukerne intuitivt og effektivt kunne benytte sine mobiltelefoner, og programmene på dem.

Som en del av SEA sin totalløsning, er selve applikasjonen som brukerne må installere på sine mobiltelefoner for å kunne fjernstyre et SEA Bluetooth nettverk. Denne applikasjonen bør følge de retningslinjene som er gitt under Kapittel 2.3. Dette betyr at applikasjonen skal inkludere et enkelt og brukervennlig brukergrensesnitt (UI), som brukerne kan gjenkjenne fra mobiltelefonens OS. Brukere ønsker ikke å sløse bort tid på hjelpefunksjoner eller brukermanualer [6]. I tilfelle applikasjonen iverksetter funksjoner som potensielt kan bruke lang tid, må den gi brukerne raske tilbakemeldinger. Dette vil hindre at brukerne får følelsen av at programmet henger. Videre må applikasjonen ha en enkel menystruktur, helst uten undermenyer, som er enkel å navigere. Så langt det er mulig bør en følge den såkalte "80/20 regelen"[1] [6]. Den må ha en startside som brukerne enkelt kan komme tilbake til. Og mens startskjermen bør linke til alle deler av applikasjonen, bør alle andre skjermer kun linke til startsiden, og dersom det er logisk, ha en mulighet for å gå tilbake til forrige skjerm eller starten av en prosedyre. Brukerne må også ha muligheten til å nå en Hjelp-skjerm som vil sørge for informasjon om applikasjonen.

Applikasjonen skal programmeres i et plattformuavhengig språk. Dette sparer utviklingskostnader, og imøtekommer en større brukergruppe. Samtidig vil applikasjonen sannsynligvis være fremoverkompatibel med fremtidige mobiltelefoner.

[1] Applikasjonen designes rundt de 20 prosentene av funksjonaliteten, som vil møte 80 prosent av brukernes behov.

Programmet skal ikke benytte lyd eller mobiltelefonens vibrator dersom dette ikke er absolutt nødvendig. I så fall skal programmet respektere brukernes innstillinger på mobiltelefonen, og ikke prøve og overstyre disse da de reduserer batteriets levetid, kan distrahere brukerne, og er uheldige dersom brukerne befinner seg i stille miljøer [6].

Det grafiske brukergrensesnittet skal være det samme uavhengig av hvilken kommunikasjonsteknologi som benyttes. Små avvik kan være nødvendig dersom store teknologiske forskjeller krever det, men ikke større enn at brukerne kan kjenne seg igjen. Overgangen mellom de forskjellige kommunikasjonsmetodene, som hvis mobiltelefonen skifter mellom Bluetooth og GPRS, skal optimalt være transparent for brukerne.

Applikasjonen skal ha muligheter for å lagre informasjon, som adresser og lignende, slik at brukerne ikke behøver å huske å taste inn denne informasjonen flere ganger.

Programmet skal kunne kjøre i bakgrunnen, og motta innkommende meldinger fra SEA Bluetooth nettverk. Dette er nødvendig for å varsle brukerne om hendelser, eller logge viktige begivenheter. Noen plattformer, blant annet enkelte MIDP implementasjoner, vil derimot resette applikasjonen når et annet program kommer i forgrunnen, som eksempel hvis mobiltelefonen mottar en telefonsamtale. Applikasjonen vil i enkelte tilfeller også slutte å kjøre. Tilstandsdata og brukerdata må derfor lagres når dette er nødvendig, slik at brukerne kan fortsette arbeidet uten å miste data [6].

Mengden data som sendes over telenettet bør minimeres. Brukere vil slutte å benytte applikasjoner som koster dem mye tid eller penger, spesielt hvis mobilenheten bare kan utføre en prosess om gangen, det vil si at det ikke er mulig å fortsette prosessen i bakgrunnen mens en tar en telefonsamtale eller lignende. [6]

Til slutt skal applikasjonen kunne installeres OTA. For brukerne vil dette være den enkleste metoden for å legge applikasjonen inn på mobiltelefonen.

3.2.2 Krav til lokalstyring

I en løsning der en mobiltelefon skal ta direktekontakt med nettverket, må det selvsagt være mulig for applikasjonen å kontrollere Bluetooth implementasjonen på terminalen. Som beskrevet i Kapittel 2.1.1, består Bluetooth spesifikasjonen av en rekke protokoller og profiler. I dag benytter SEA en Bluetooth brikke som tilbyr kommunikasjon over Serial Port Profile. Over denne har SEA implementert en egen stakk for ruting/relaying, sikkerhet og applikasjonsmeldinger. For enklest å kunne implementere deler av denne protokollen på mobilenheten, bør det være mulig å utvikle applikasjoner direkte mot RFCOMM/Serial Port Profile.

Protokollen på mobilterminalen kan derimot forenkles noe i forhold til dette. Blant annet vil ikke mobiltelefonen være med på relaying av trafikk. Den må likevel kjenne til rutingen hvis den skal være mobil i nettverket.

Med lokalstyring er det meningen at en mobiltelefon kan komme inn i nettverket uten å vite adresser eller lignende på enhetene. En må derfor ha mulighet til å utføre søk etter kompatible enheter. Og etter en tilkobling, må det i mobilterminalen være mulig å varig lagre adresser og egenskaper for enheter. Dermed kan en oppnå raskere tilkobling og styring mot kjente enheter, noe som er avgjørende for systemets funksjonalitet og brukervennlighet.

Ved tilkobling mot enheter i nettverket, må det eksistere en viss grad av sikkerhet i form av autentisering og kryptering. Selv om det er ønskelig at en tilkobling med en mobiltelefon er så enkel som mulig, må det skje en autentisering som hindrer at en mobiltelefon direkte kan styre alle SEA-enheter. I tillegg er kryptering viktig der alarmenheter er en del av nettverket. Slik sikkerhet kan godt implementeres på høyere lag, men sikkerhetsmekanismer spesifisert i Bluetooth spesifikasjonen bør også kunne aktiveres i applikasjonen, ettersom dette kanskje vil benyttes i SEA sitt nett.

De samme kravene gjelder også dersom en benytter en lokal GW, som oversetter mellom kommunikasjonsteknologien som benyttes på mobiltelefonen, og den som benyttes i SEA nettet. Mobiltelefonen må fremdeles ha mulighet til å søke etter kompatible GW'er, og sikkerhet må implementeres for innlogging og kryptering av data mot GW'er.

3.2.3 Krav til fjernstyring mot hjemmegateway

Så fort det benyttes en rekke ulike nettverk for å nå en node, vil forsinkelse bli en faktor. I et system som er basert på tilbakemelding til brukeren når en styrer nettet, er en avhengig av at forsinkelsen ved dataoverføring ikke er for stor. I alle fall vil lav forsinkelse være et krav i en større løsning med alarm muligheter. Det blir også langt verre å presentere et tilsvarende UI dersom forskjellen i forsinkelsen er for stor mellom lokal- og fjernstyring av nettet.

En annen forskjell mellom lokal- og fjernstyring er at det som regel innebærer en omkostning for å benytte sistnevnte. For brukeren kan det derfor være viktig at applikasjonen gir beskjed når en benytter tjenester som koster noe ved bruk. En applikasjon bør derfor kunne markere når data overføres, og be om bekreftelse fra bruker før den benytter tjenester som koster penger. Dette, kombinert med at en ønsker lav forsinkelse, gjør at applikasjonen må kunne lagre mest mulig av statiske data fra hjemmenettet.

Muligheten for fjernstyring av et nettverk åpner for langt større sikkerhetsproblematikk enn for lokalstyring av nettverket. Ettersom nettverket er tilgjengelig over et mye større geografisk område, er det også større muligheter for at uvedkommende vil prøve å aksessere nettverket. Derfor må det implementeres sikkerhet ved innlogging til systemet og ved overføring av meldinger.

Mobiltelefoner benyttes i dag til en rekke ulike tjenester, og en kan ikke forvente at brukeren vil ha applikasjonen aktiv hele tiden. For derimot å kunne implementere alarmfunksjonalitet, må applikasjonen kunne motta meldinger til enhver tid. For at ikke brukeren skal måtte ha applikasjonen konstant aktiv, må applikasjonen kunne kjøres i bakgrunnen, eller kunne våkne dersom innkommende meldinger sendes fra hjemmenettverket uten forespørsel.

4 Anbefalt applikasjonsplattform og kommunikasjonsteknologi

4.1 Anbefalt applikasjonsplattform

Plattformer som Symbian OS, Smartphone og MontaVista Linux CEE, har forbedret mulighetene for applikasjonsutvikling på mobiltelefoner. Disse tilbyr applikasjonsutviklingsmuligheter for mobiltelefonprodusentene. I begrenset omfang åpner dette også for tredjepartsutvikling. Som regel vil det dreie seg om C/C++ støtte, som resulterer i svært effektive programmer og god ressursutnyttelse. Likevel vil det være nødvendig med separate utviklingsprosesser for de ulike plattformene. Ofte kan det til og med være nødvendig med tilpassinger for hver enkel plattformimplementasjon.

Brew og Mophun tilbyr en viss grad av plattformuavhengighet. Brew satser i dag tyngst på det amerikanske markedet. Mophun er på sin side primært designet for spillutvikling, og utvikling av ordinære applikasjoner tynges ned av mangel på standardiserte GUI-elementer.

J2ME fra Sun Microsystems har derimot bred støtte blant mobiltelefonprodusentene, og har etter hvert fått oppslutning hos andre applikasjonsplattformprodusenter. Dette går blant annet frem av at J2ME er den applikasjonsplattformen som i dag oftest leveres på mobiltelefoner, og at en rekke produsenter tar del i spesifiseringen av nye API'er.

For SEA er det viktig å ha en løsning som er enkel for brukeren å laste ned og benytte. J2ME har i dag gjennom standardiserte API'er god støtte for OTA, høynivå kommunikasjonsprotokoller og UI. Det er fremdeles begrenset støtte for lavnivå kommunikasjons-API'er i J2ME, men vi mener at støtten for disse etter hvert vil bli forbedret. Basert på disse argumentene, og de tidligere evalueringene av applikasjonsplattformer, anser vi derfor at J2ME vil være det beste valget av applikasjonsplattform for å løse SEAs problemstilling.

4.2 Anbefalt kommunikasjonsteknologi

Antallet mobiltelefoner som i dag benytter 802.11 standarder, heretter kun referert til som 802.11, er svært begrenset. De finnes hovedsakelig på enkelte dyrere PDA baserte mobiltelefoner, som har et svært begrenset marked. Selv om det sannsynligvis vil bli vanligere å se mobiltelefoner støtte 802.11 i fremtiden, har Bluetooth blitt akseptert som en standard blant de fleste mobilprodusentene. Teknologiene er heller ikke direkte i konkurranse, ettersom Bluetooth vil dekke behovet for kommunikasjon over kort rekkevidde mot håndsett, printer eller PC, mens 802.11 vil bli benyttet til Internett og IP-telefoni. Selv om 802.11 skulle dukke opp på flere mobiltelefoner i fremtiden, vil ikke dette fortrenge mobilenes Bluetooth implementasjon. Sikkerheten er tilsvarende høy i begge teknologiene, selv om Bluetooth har et enklere oppsett. Vi mener derfor at Bluetooth vil være det beste valget for kommunikasjon over kort rekkevidde på mobiltelefoner. Dette passer også bra mot det nettet SEA benytter i dag, i tillegg til at åpne API'er for Bluetooth implementasjoner begynner å bli tilgjengelige for mobiltelefoner.

For fjernstyring av hjemmenettet finnes flere aktuelle teknologier. Grunnet de svært ulike egenskapene til de ulike kommunikasjonsteknologiene, kan vi ikke uten videre anbefale en teknologi fremfor en annen. SMS og direktetilkoblet GSM er ukompliserte vedrørende adressering, men er dyre og trege å benytte i sammenhengen SEA ønsker. Tilkobling til eksterne nettverk med IP over GPRS eller WAP, gir en mer fleksibel løsning, men medfører også utfordringer når det gjelder adressering. WAP er svært godt støttet på mobiltelefoner i dag, og var ment å tilby valgfri aksesstilgang, og muligheter for adressering mot mobiltelefonen. Teknologien har derimot ikke tatt av slik en trodde den ville, og er blitt svært avansert. Det ser derfor ut til at en for fremtiden ønsker å tilby vanlig IP-teknologi mot Internett og andre nettverk, fra mobiltelefoner som benytter GPRS eller GSM-data mot disse nettverkene. Mengden av mobiltelefoner som i dag støtter TCP og UDP i J2ME, er derimot begrenset. Vi tror likevel at dette vil være vanlig å tilby på nesten alle mobiltelefoner som kommer fremover.

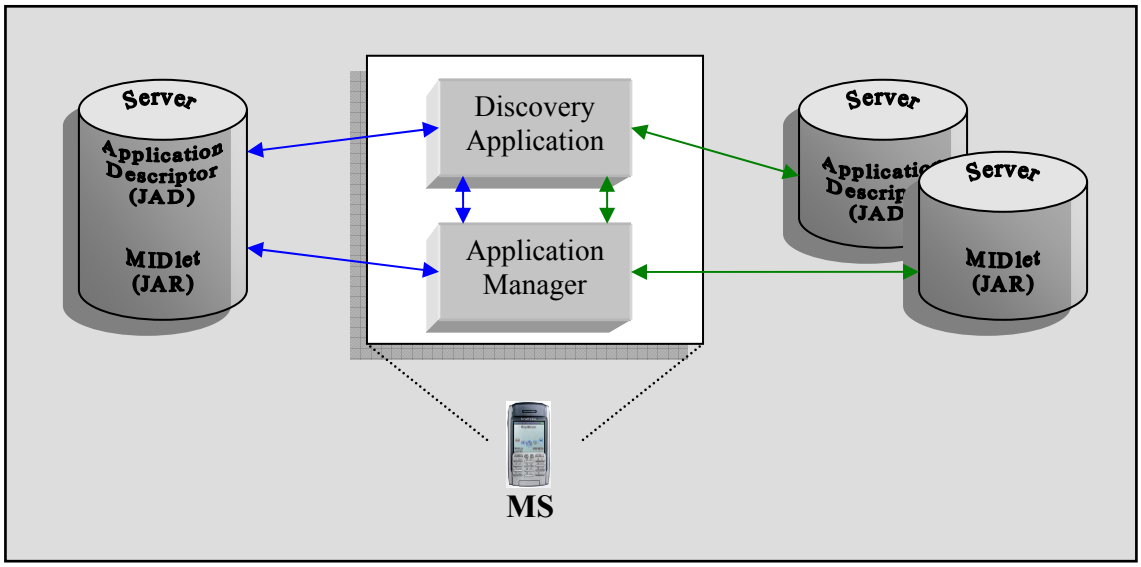
Vi vil anbefale å benytte SMS i en enklere løsning, der det settes lavere krav til oppdateringer, alarmfunksjonalitet og interaksjon, ettersom en SMS løsning er raskere og billigere å implementere i forhold til andre løsninger. I en mer interaktiv løsning, som gir muligheter for å aksessere nettverket fra andre terminaler på Internett, ville vi ha benyttet GPRS med tilgang til et IP-basert APN. Dersom en oppnår en toveiskommunikasjon mellom to mobilterminaler på det samme nettverket, vil en da kunne sende forespørsler i begge retninger, noe en ikke oppnår ved å benytte HTTP over WAP.

5 Løsninger

5.1 Deployering av programvare

Deployeringen av applikasjonen begynner med opprettelsen av en MIDlet[1] suite. En MIDlet suite består av en eller flere MIDlets, og henviser til en JAR og en JAD fil. JAR fila inneholder MIDlet'en og eventuelle ressurser som applikasjonen trenger. JAD fila inneholder informasjon om applikasjonen, som hvor JAR fila kan lastes ned fra, og hvor stor lagringsplass denne trenger. JAD fila kan ligge på den samme serveren som den tilhørende JAR fila, eller den kan ligge på en helt annen server. Poenget er at JAD fila må lastes ned først slik at mobiltelefonen vet hvor den finner JAR fila. MIDlet suiten lokaliseres til en eller flere HTTP servere for publisering.

På mobiltelefonen er det Java Application Manager som er ansvarlig for å laste ned, installere og kjøre MIDlet'en. Før applikasjonen kan installeres på mobiltelefonen, må MIDlet'en lokaliseres. Ifølge [2] skjer dette i tre steg. Først søker mobiltelefonen etter informasjon om MIDlet'en. Deretter lastes JAD fila ned. Og til slutt lastes JAR fila ned. Figur 5.1 beskriver prosessen med å lokalisere og laste ned MIDlets.



Figur 5.1: To eksempler på lokalisering og nedlasting av MIDlets.

Kommunikasjonen mellom MS og server foregår ved hjelp av HTTP forespørsler. Muchow [2] beskriver i detalj de ulike parametrene forbundet med etterspørsler av JAD filer og MIDlets. Eventuelt kan klienten benytte WAP, og i disse tilfellene går kommunikasjonen gjennom GW'er som oversetter fra WAP til HTTP kall. I tillegg stilles visse krav til nevnte GW'er og klienter, noe Muchow går nærmere innpå.

[1] Applikasjoner som kun bruker API'er definert i MIDP og CLDC, refereres til som MIDlets.

Når MIDlet'en er vellykket nedlastet, skjer installasjonen av applikasjonen automatisk, såkalt OTA installasjon. Applikasjonen presenteres deretter for brukeren slik det er normalt for enheten. Det siste steget i prosessen innebærer rapportering av status til serveren. Dette skjer ved å sende en statuskode til en URL som spesifiseres i JAD fila.

Programvaren på mobiltelefonen kreves også å støtte oppgradering av MIDlet suiter. Dette og avinstallering av MIDlet suiter er konstruert for å fungere med minst mulig innblanding fra brukeren. Stort sett trenger brukeren kun å bekrefte handlingen. Programmereren må ikke ta andre hensyn annet enn å fylle ut attributtene i JAD fila korrekt.

5.2 Lokalstyring

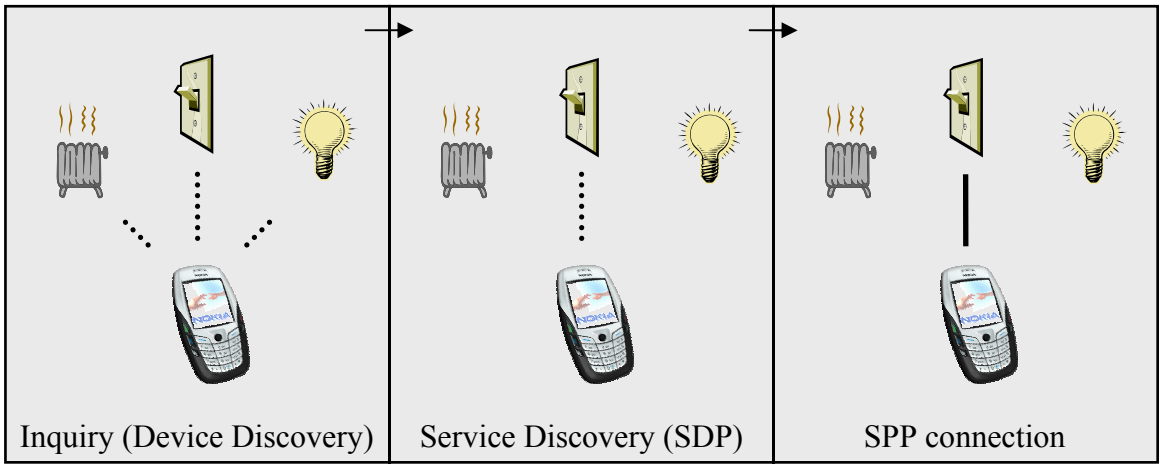
5.2.1 Bluetooth

5.2.1.1 Arkitektur

Tilkoblingen til SEA sitt nett er spesiell i forhold til det som er vanlig for Bluetooth applikasjoner på mobiltelefoner. Vanligvis opprettes det en kobling mot en kjent enhet, som ved synkronisering av data mot en PC, eller mot en annen mobiltelefon for spill eller samtaleapplikasjoner. I SEA-løsningen er det derimot ønskelig å ha direkte kontakt mot en rekke enheter for å styre disse. SEA har ruting og relaying muligheter for sine enheter, men det er ikke tenkt at mobiltelefonen skal være en aktiv node i relayingen. I første omgang ønsker en at mobiltelefonen skal søke etter styrbare enheter, og ta direkte kontakt med disse.

Bluetooth-protokollstakken gir tilgang til en rekke protokoller, som gjør den i stand til å søke etter enheter i nærheten og tjenester for disse. For Bluetooth deles denne prosessen i to. Å søke etter enheter i nærheten, uttrykkes i Bluetooth-terminologien som "inquiry". Et inquiry kan ta mellom 8 og 10 sekunder, og vil returnere en 6 byte unik ID (MAC-adresse), og en såkalt "class of device" record for hver Bluetooth enhet. Class of device gir en generell indikasjon, i form av klassetype, på hvilke tjenester en Bluetooth enhet støtter. De ulike klassene er spesifisert av Bluetooth SIG, og en kan derfor ikke uten videre spesifisere nye dersom en for eksempel ønsker en unik spesifisering av egne enheter eller tjenester. Et ekstra søk etter tjenester, representert ved service records, gjøres derfor enkeltvis i en separat prosess. Service records kan være egendefinerte eller basert på profilene som finnes i Bluetooth spesifikasjonen. Ved hjelp av service records, kan altså applikasjonen presentere de enheter som er ønskelig, og som støtter de aktuelle protokollene. Selv om ID'er for standard profiler, som L2CAP og SPP, finnes i Bluetooth spesifikasjonen, kan en legge inn ekstra informasjonselementer i en service record, og dermed skille ut sine egne enheter.

Derfor vil en i SEA sitt tilfelle, kunne sjekke om en enhet støtter SPP, som SEA benytter i dag. Samtidig kan en benytte en egen beskrivelse i en service record, slik at en ikke presenterer enheter som ikke støtter SEA sin stakk, for brukeren.



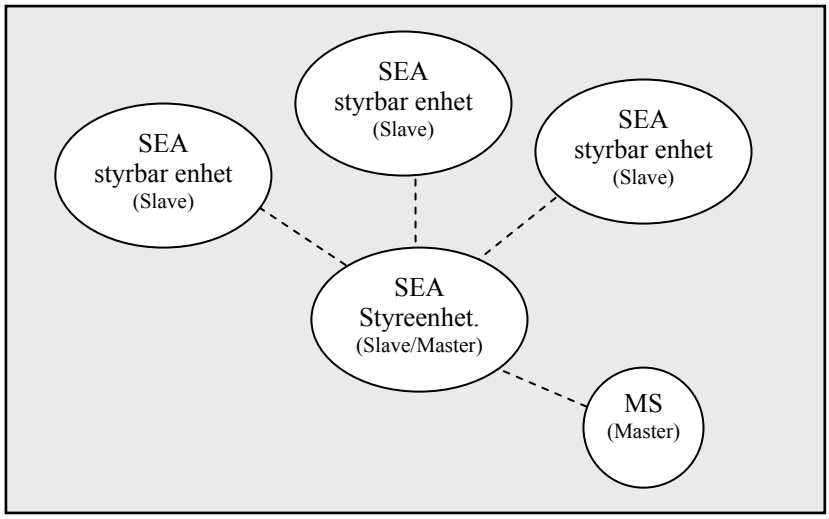
Figur 5.2: Søk og tilkobling for en mobiltelefon mot SEA enheter over Bluetooth.

SEA enhetene benytter kommunikasjon direkte over SPP. Denne profilen tilbyr toveis, pålitelig datakommunikasjon ved hjelp av RFCOMM.

Noen negative faktorer i en løsning som den SEA skal ha, er at både søket etter enheter, og oppkoblingen mot valgt enhet, vil ta merkbart med tid. Selve søkingen etter enheter vil kun skje når en ønsker å installere nye enheter, ettersom en vil lagre adressen til enheter en ofte kobler til. I en installasjonsprosess kan derfor dette være akseptabelt. Tilkoblingen vil derimot alltid ta omtrent ett sekund.

Så lenge vi snakker om direkte kontakt mot en og en enhet, bør en kunne benytte autentisering og kryptering, som finnes i Bluetooth spesifikasjonen. Ved hjelp av pairing innebygd i Bluetooth standarden, kan en felles nøkkel benyttes for å få muligheten til å utføre autentisering og kryptering. Ved relaying er det derimot bedre å benytte ende til ende sikkerhet på høyere lag i SEA-protokollen.

I følge Bluetooth spesifikasjonen [10], er det vanligvis enheten som initialiserer en tilkobling, som blir master i et piconett. I SEA sitt nett, vil et piconett som regel bestå av en styreenhet som er master i et nett mot flere styrbare enheter, som blir slaver. Idet en mobiltelefon kobles mot styreenheter, vil vi få et scatternett, hvor styreenheten opptrer som både slave og master (Figur 5.3).

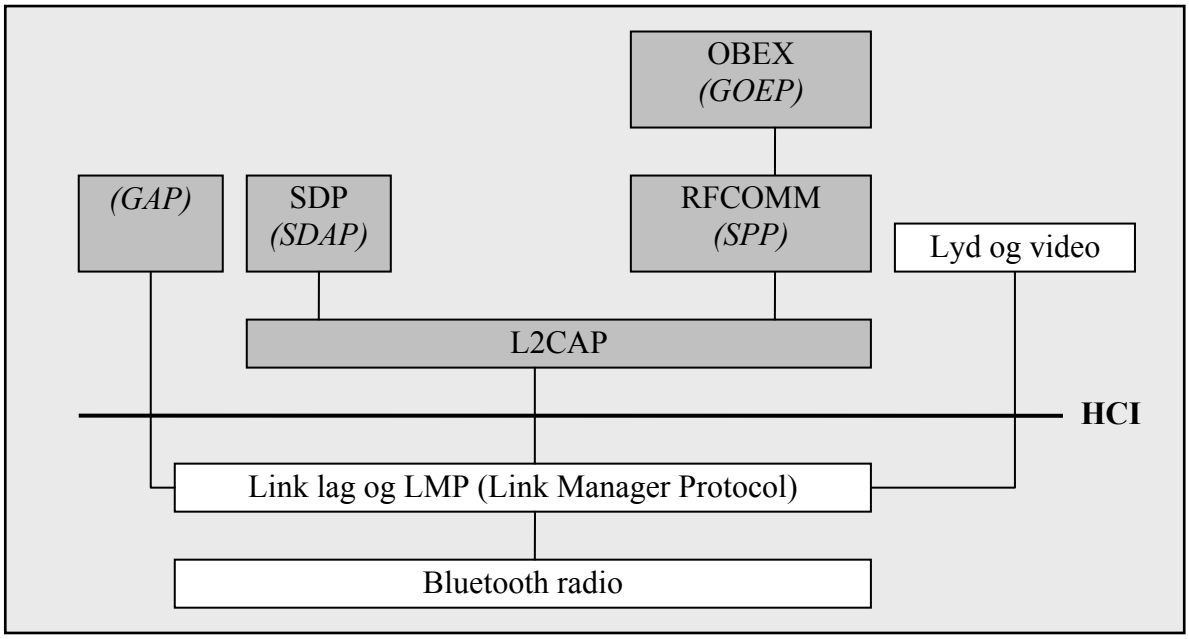


Figur 5.3: Scatternett ved tilkobling av en mobiltelefon.

5.2.1.2 Implementasjon

Motorola hadde hovedansvaret for en spesifikasjon som skal være en felles standard for Bluetooth kontroll fra J2ME. Den ble ferdig 22. mars 2002 og er spesifisert under JSR-82. Standarden definerer et grensesnitt mot Bluetooth, og så er det opp til mobilleverandørene å implementere støtte for denne. På denne måten kan samme Java program benyttes uavhengig av hvordan Bluetooth er implementert på mobiltelefonen, så lenge den støtter JSR-82.

Selv om det ikke er et absolutt krav med MIDP 2.0, er det naturlig å ha støtte for denne før JSR-82 implementeres. Få leverandører har per i dag kommet med Bluetooth støtte. En oppdatert liste finnes på [35], hvor det 4. april 2004 er 11 mobiltelefoner som støtter JSR-82. Bortsett fra en mobiltelefon, er dette telefoner som er basert på Symbian OS, som har implementert JSR-82 i sin 7.0s versjon. Mobiltelefonene som benytter proprietære OS vil mest sannsynlig kreve noe lengre tid for å få lagt inn støtte, men implementasjonen av JSR-82 regnes ikke for å være for kompleks. Ettersom den kun implementerer de laveste lagene og profilene i Bluetooth stakken, kreves kun en oversetting mellom Java kall og stakken som allerede er implementert mobiltelefonene for å støtte headset profiler og lignende [3].



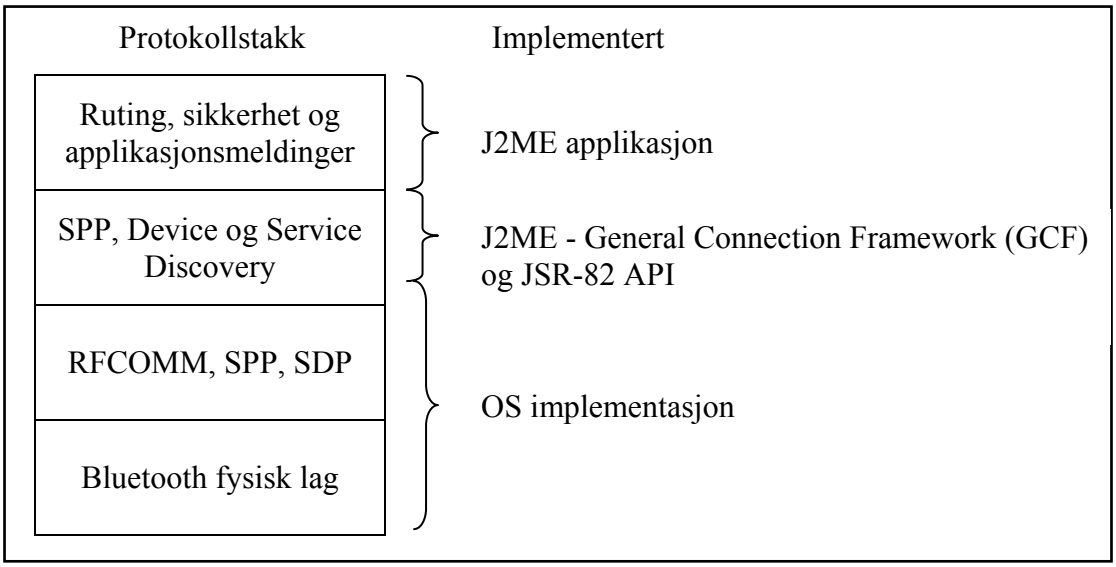
Figur 5.4: JSR-82 spesifiserer tilgang til flere protokoller og profiler (mørkegrå).

JSR-82 standarden spesifiserer tilgang til GAP, SPP, SDP, L2CAP og OBEX (Figur 5.4). Ved hjelp av GAP inquiry vil en kunne sette Class of Device for å begrense antallet enheter til den klassen SEA sine enheter tilhører. JSR-82 spesifiserer også muligheter for å gjøre service discovery med SDP. Dermed kan en finne de enhetene som er SEA kompatible og tilby tilkobling mot disse for brukeren.

Kommunikasjon over RFCOMM er tilsvarende som å kommunisere over en socket eller en serieport tilkobling ettersom JSR-82 er basert på GCF. Selve Java API'et har derimot ingen direkte kontroll over oppsett av Bluetooth implementasjonen på mobiltelefonen. Som regel er det derfor implementert et Bluetooth Control Center (BCC) på mobiltelefonen. Denne har som oftest et eget native program hvor brukeren kan endre oppsettet.

BCC tar seg også av nøkkelhåndtering ved pairing av Bluetooth enheter når en benytter RFCOMM i JSR-82. Dette vil si at dersom kryptering, og dermed pairing, skal benyttes i Bluetooth, vil BCC på mobiltelefonen ta over interaksjonen mot brukeren og spørre om pin kode. Ettersom implementasjonen av BCC kan være ulik fra mobiltelefon til mobiltelefon, vil det være vanskelig å forutse om denne interaksjonen passer dårlig med Java applikasjonen. Dette er gjort for at ødeleggende applikasjoner ikke skal kunne benytte Bluetooth uten at brukeren er klar over det. Det er en mindre faktor i et system hvor en oppretter kontakt mot en enhet og utfører en rekke operasjoner mot denne. I SEA sitt tilfelle, hvor en ønsker et raskt UI og en utfører stadige tilkoblinger mot flere enheter, kan det derimot være irriterende ekstra interaksjon for bruker. Hvis en i tillegg benytter kryptering på Bluetooth nivå vil ikke en J2ME applikasjonen kunne lagre og tilby en slik nøkkel automatisk.

Figur 5.5 viser en oppsummert protokollstakk- og implementasjonsoversikt dersom en benytter JSR-82 i en løsning for SEA. En får tilgang til SPP, device og service discovery interface via GCF og API'et, og en kan implementere egne protokoller over disse. Tilgang til lavere nivå er derimot styrt av OS'et og BCC på mobiltelefonen.



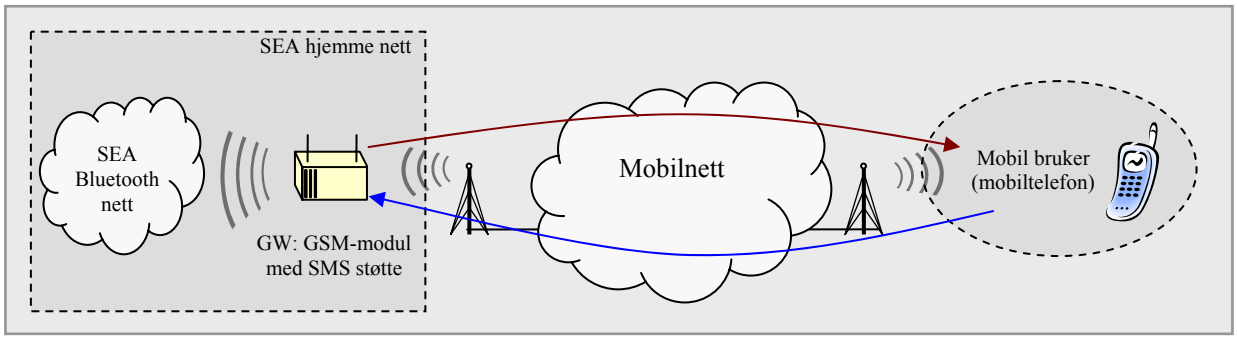
Figur 5.5: Protokollstakk for J2ME og Bluetooth.

5.3 Fjernstyring

5.3.1 SMS basert kommunikasjon

5.3.1.1 Arkitektur

Selv om SMS de siste årene har hatt en ekstrem suksess, har det større svakheter som kommunikasjonsmedie ved styringssystemer. Maks leveringstid er i SMS kommunikasjon satt svært høyt, da det kun utnytter ledig kapasitet i nettet. I tillegg er kostnadene ved data overføring svært store sammenlignet med GPRS. Likevel er det mange systemer i dag som benytter SMS for fjernstyring av smarthus, da spesielt etter at personsøkertjenesten ble avsluttet. Mye av grunnen til dette er at løsninger basert på SMS kan benyttes fra nesten alle GSM telefoner på markedet. En SMS løsning baserer seg kun på mobilnettet med telefonnummer som adressering, og krever derfor ikke ekstra tjenester eller tredjepartservere. Det er i derimot også bakdeler med dette, ettersom som det hindrer muligheter for å enkelt kunne styre hjemmenettet fra for eksempel terminaler tilkoblet Internett.



Figur 5.6: SMS styring via SMS gateway med bluetooth støtte.

I en SMS løsning vil meldinger mellom en hjemme-GW og den mobile brukeren sendes og adresseres ved hjelp av fastsatte telefonnummer. Begge sider vil ha mulighet for å initialisere kontakt med den andre. Grunnet forsinkelse som kan oppstå mellom SMS serverne er det likevel lite egnet som eneste kommunikasjonsmedie i alarmsystemer. Det gjør også at UI på mobiltelefonen ikke kan oppdateres fortløpende med informasjon fra hjemmenettet, når brukeren navigerer i menyene. Siden en SMS har lang leveringstid og høy pris, bør en kun presentere informasjon fra hjemmenettet når brukeren aktivt spør om å laste dette ned.

Pris for å implementere en slik løsning vil være lav. De fleste GSM moduler i dag har god støtte for håndtering av SMS. Løsningen krever heller ikke vedlikehold av tredjepartsmoduler ettersom mobilnettet er det eneste som benyttes, og kommunikasjon går direkte mellom GW-modul og den mobile terminalen. For kunden vil kostnadene være pris på GW-modul, abonnement hos mobiloperatør og prisen for sending av SMS meldinger.

Sikkerheten i en SMS løsning vil være til dels enkel å implementere. Ettersom trafikken kun går via mobilnettet, som regnes som rimelig sikkert i dag, kreves det ikke ekstra kryptering for styring av smarthusløsninger. En må likevel forsikre seg mot at ikke uvedkommende kan koble seg til hjemmenettet. Det regnes i dag som vanskelig å ringe under en annen abonnents nummer, uten å ha fått tilgang til abonnentens SIM kort. I en SMS løsning kan derfor styring av nettet filtreres på spesielle telefonnummer. Eventuelt kan også en nøkkel sendes med i hver melding slik at en ikke er avhengig av at forhåndsprogrammerte telefonnummer. Ettersom en melding består av et fast antall byte vil dette som regel ikke gå ut over kostnader eller antall meldinger som må sendes.

5.3.1.2 Implementasjon

Siden ikke alle mobilterminaler med J2ME nødvendigvis støtter SMS, er dette ikke en del av det obligatoriske J2ME API'et. Et eget API, kalt WMA (Wireless Messaging API), er likevel blitt spesifisert. Det baserer seg på GCF, og gir aksess til SMS tjenesten på mobiltelefonen.

WMA, spesifisert under JSR-120, gir tilgang til å sende og motta SMS i en J2ME applikasjon. Meldinger kan kodes både som 8 bit datameldinger eller 7 bit tekst meldinger. Som beskyttelse mot at en applikasjon skal kunne fjerne meldinger som egentlig er ment til brukeren, må en SMS være kodet med ekstra header informasjon for at den skal kunne videresendes til en J2ME applikasjon. Denne ekstra informasjonen består av en bit i den normale SMS headeren, som impliserer at en opsjonell header vil følge i hoveddelen av meldingen. I den opsjonelle headeren vil en kreve et portnummer slik at mobiltelefon OS'et vet hvilken Java applikasjon som skal motta meldingen. Dette har en bakdel i den løsningen vi snakker om her. Blant annet kunne en tenke seg at hjemme-GW'en er basert på en mobiltelefon med SMS og Bluetooth støtte. En normal SMS sendt fra en mobiltelefon uten en Java applikasjon, vil dermed ikke være mulig å hente opp i applikasjonen på GW'en. Dette burde derimot ikke være noe problem dersom en benytter en spesialmodul eller en mobiltelefon med C++ som hjemme-GW.

Fra WMA v1.1 og MIDP 2.0 kan et Java program startes automatisk når en innkommende SMS blir sendt på en gitt port. Til dette benyttes MIDP-PUSH arkitekturen som gjør at et program ved kjøre- eller installasjonstid, kan registrere seg til å startes ved innkommende meldinger. Dette vil gjøre det mulig å kunne motta meldinger fra hjemmenettet, uten at brukeren trenger å ha en Java applikasjon kjørende på mobilterminalen til en hver tid.

5.3.2 Løsning basert på IP-nettverk med sentralisert server

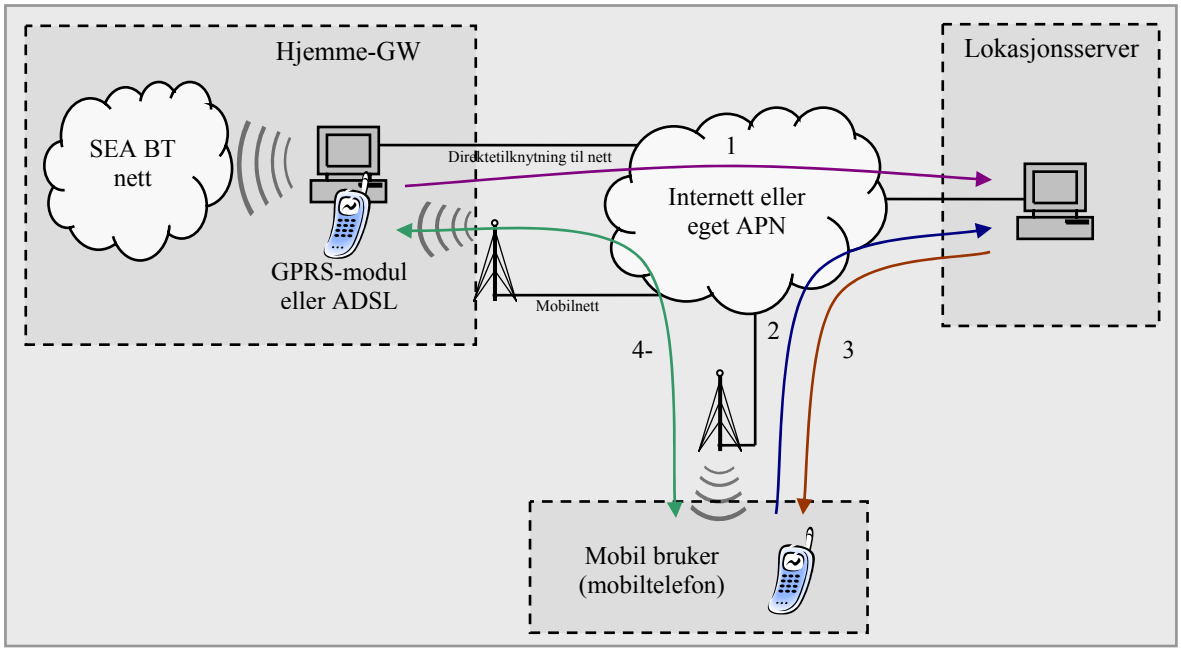
5.3.2.1 Lokasjonsserver arkitektur

Mange av løsningene som finnes på markedet i dag når det gjelder fjernavlesning og overvåking ved hjelp av mobile noder og GPRS, benytter en noe annerledes arkitektur enn den SEA trenger. Disse løsningene baserer seg som oftest på en server på Internett, som regelmessig blir kontaktet og oppdatert fra GPRS noder. Serveren kan så behandle måledataene og presentere disse oppsummert i HTML eller andre formater. SEA sin løsning krever derimot en bedre toveis kommunikasjon mellom to GPRS enheter i mobilnettet.

Dette er en utfordring ettersom GPRS og WAP teknologien er basert på at en kobler seg til et annet nettverk, og at datatrafikk ikke skal gå direkte mellom to mobiltelefoner i mobilnettet. Når en GPRS tilkobling utføres, har enten mobilterminalen eller HLR navnet på APN'et mobilnoden skal koble til. APN'et oversettes til adressen til en GGSN, som fungerer som en ruter mellom mobilnettet og et eksternt nettverk. Vanligvis tilbyr mobiloperatørene tilgang til for eksempel Internett, WAP og MMS APN. På grunn av at mobiloperatørene har begrenset antall IP-adresser, tildeles disse som regel dynamisk. Dette vil si at vi ikke nødvendigvis kjenner en fast adresse til en mobiltelefon til enhver tid, noe som gjør det vanskeligere å initialisere kontakt fra en node på det eksterne nettverket mot mobiltelefonen. I tillegg beskytter noen av mobiloperatørene GPRS nodene ved å ikke tillate nettverksinitialisert trafikk [26].

En tilkobling mellom to GPRS mobiltelefoner på det samme nettverket, må derfor løses med kommunikasjon mot en felles server som har en kjent adresse. Denne serveren kan ha varierende funksjonalitet ut i fra hvor kompleks og funksjonell en slik løsning skal være.

En løsning er å benytte en lokasjonsserver. Denne serveren fungerer som adresseoppslag for GPRS nodene. Både GPRS-modulen som befinner seg i hjemmenettet og mobiltelefonen som benyttes til fjernstyring, registrerer IP-adressen sin hos lokasjonsserveren når disse kobles til nettverket. Dette gir mulighet for at både hjemmenettet og mobiltelefonen kan be lokasjonsserveren om adressen til motstående part, og kan initialisere direktekontakt mot denne. Dette er samme funksjonalitet som finnes i SIP basert kommunikasjon kjent fra IP-telefoni og meldingssystemer. Figur 5.7 illustrerer et meldingsforløp i en slik løsning.



Figur 5.7: Fjernstyring av SEA-nett med lokasjonsserver for adresseoppslag.

[1] Gateway (PC eller GPRS modul) tar kontakt med en lokasjonsserver hos SEA og registrer adressen sin.
[2] En mobil modul foretar en spørring etter en registrert hjemme-GW.
[3] Lokasjonsserveren svarer med korrekt adresse.
[4] Den mobile enheten kjenner korrekt adresse til hjemme-GW'en og kan ta kontakt med denne.

Denne løsningen åpner også for at bredbåndtilgang i hjemmet kan benyttes som GW mot Bluetooth nettverket. I tillegg kan styringsapplikasjoner utvikles til Windows baserte PC'er med internettilgang.

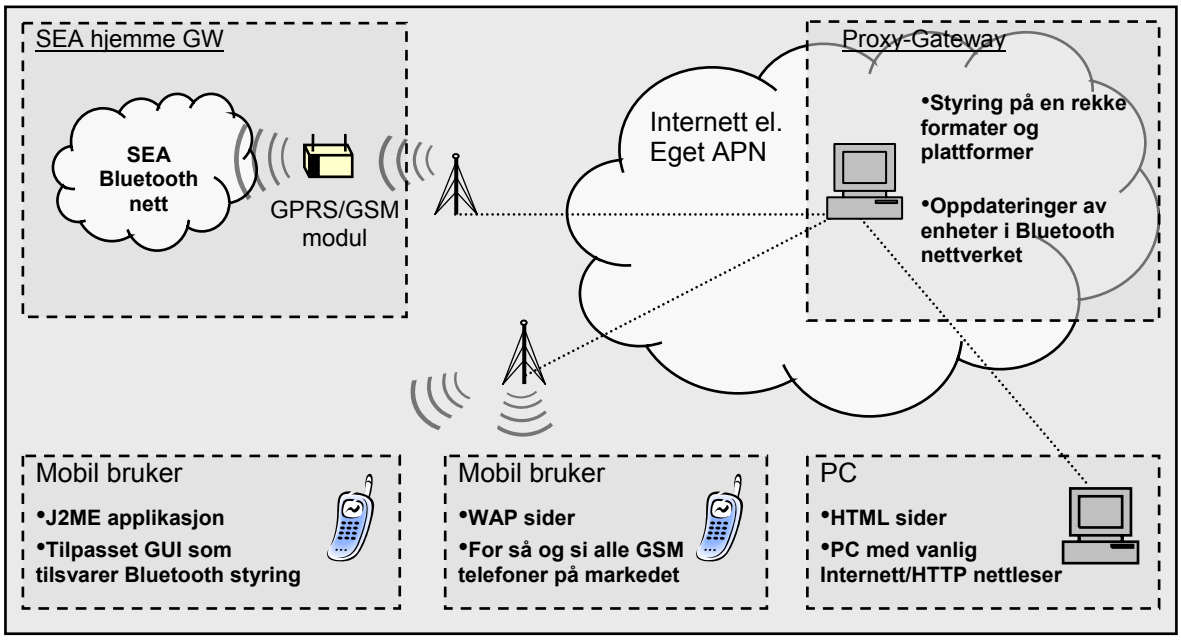
Hvis en først tar for seg selve registreringen og oppslaget av adresser på en lokasjonsserver, kan dette for eksempel gjøres med å benytte HTTP/WAP eller Web Services. Disse benytter pull basert arkitektur, som fungerer bra ettersom trafikken alltid vil initialiseres mot lokasjonsserveren. Dette vil forenkle utviklingen, og gjøre det lettere å få løsningen kompatibel med andre systemer og løsninger. Etter at en mobiltelefon har adressen den skal kobles mot, kan den starte trafikk direkte mot denne. Ved å benytte UDP eller TCP trafikk over IP, oppnår en rask toveiskommunikasjon hvor begge sider kan initialisere trafikk. Selv om HTTP eller Web Services har bedre støtte på mobile enheter, ville en løsning med disse bety at begge sider må kjøre en form for web server. Dette er for komplisert og tungt for små mobile enheter som vi benytter her. En binær protokoll over TCP/IP vil sikre at datakommunikasjonen holdes kompakt, og at hjemme-GW'en slipper en tyngre dekoding av meldinger. Et krav i en slik løsning er likevel at nettverksoperatørene tillater nettverksinitialisert trafikk, og at det ikke benyttes Overloading NAT rutere mot GPRS terminalene. Dersom dette benyttes, vil de få tildelt en IP-adresse som det ikke kan rutes trafikk til. En peer-to-peer UDP eller TCP tilkobling mellom GPRS-modulen i hjemmenettet og mobiltelefonen vil ikke være mulig. Ingen av dem kan sende den første meldingen for å sette opp NAT gatewayene.

Sikkerheten i GPRS nettverket regnes for å være god. Men så fort datatrafikken når andre nettverk kan en ikke anta at datatrafikken blir beskyttet. Autentiseringen og krypteringen vi finner i GPRS nettverket vil heller ikke kunne benyttes på IP-nettverket. Spesielt når en GPRS-node registrerer adressen sin mot lokasjonsserveren bør både noden og serveren autentiseres. Det begynner å komme støtte for preinstallerte betrodde tredjeparts sertifikater, som for eksempel Verisign, på mobilterminaler, slik at en kan benytte PKI (Public Key Infrastructure) for å autentisere serveren. Terminalene bør derimot kunne autentiseres gjennom et kjent felles passord på terminalen og serveren. Etter autentisering vil normal begge sider ha en session key som kan benyttes til kryptering av datatrafikken. Ved å benytte for eksempel SSL ved HTTP eller TLS ved sockets kan både autentisering og kryptering gjøres på en enkel måte.

Sikkerheten mellom en mobil enhet og hjemme-GW'en bør også implementeres med autentisering av mobilnoden. Her vil det enkleste være å benytte en felles kjent nøkkel. Kryptering kan sannsynligvis gjøres på samme måte som når mobiltelefonen befinner seg innen nettverket, hvor SEA har kryptering på høyere lag, ettersom hjemme-GW'en ikke trenger å dekryptere meldinger før de når den enheten de er adressert til.

5.3.2.2 Sentralisert Proxy-Gateway arkitektur

En løsning som ikke stiller de samme kravene til mobiloperatørens nett, er å sende all datatrafikk via en ekstern server. En slik proxy-gateway vil da kunne oversette mellom ulike protokoller, og kan tilby tilpassede grensesnitt for presentasjon og styring. Initialiseringen av kommunikasjon over GPRS vil alltid gå fra GPRS-moduler og mobiltelefoner mot proxy-gateway'en, slik at en unngår problemer med adressering og Overloading NAT i mobilnettet. Tilkoblingen vil så holdes oppe slik at datatrafikk kan sendes i begge retninger. Dersom det likevel skulle være nødvendig å kontakte mobilnodene fra serveren, kan dette gjøres ved at en implementerer SMS sending fra proxy-gateway'en. Dermed kan den sende SMS for å få hjemme-GW'en eller mobiltelefonen til å opprette kontakt mot denne. Dette ligner på deler av WAP Push arkitekturen, og en vil kunne benytte WAP Push implementasjonen som finnes på enkelte mobiltelefoner for å få til en automatisk oppkobling mot serveren. Eventuelt kan denne tilkoblingen programmeres direkte i J2ME.

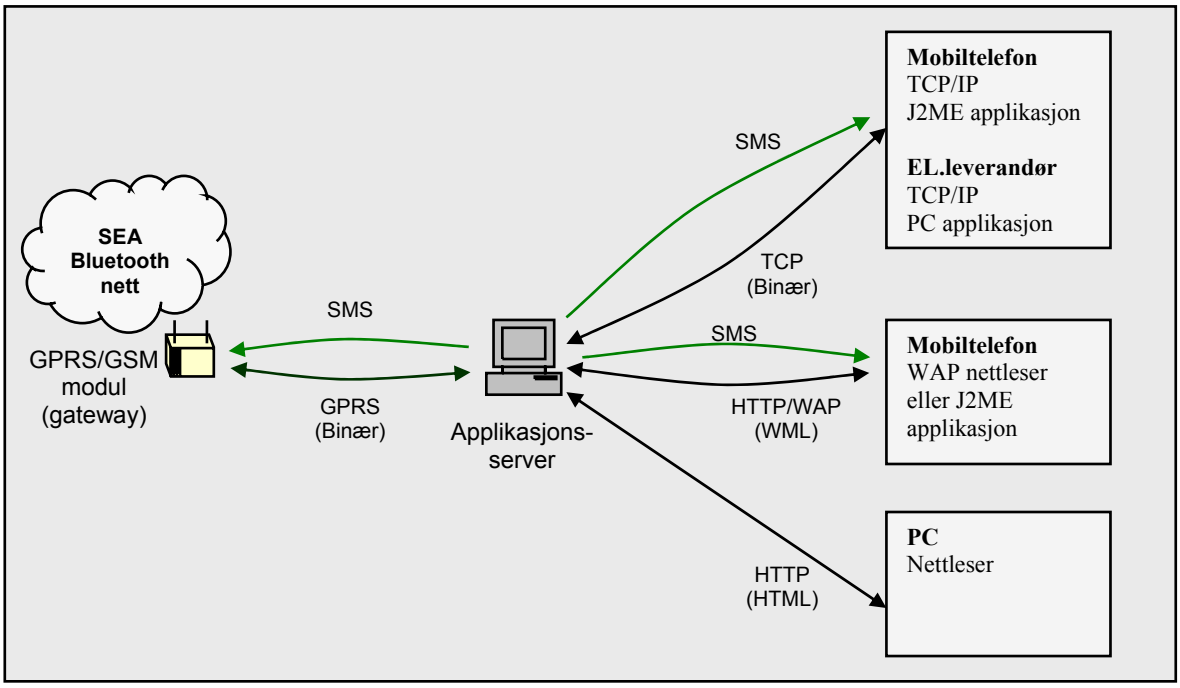


Figur 5.8: Fjernstyring av SEA-nett med proxy-gateway for datakommunikasjon.

Slik Figur 5.8 illustrerer kan proxy-gateway'en presentere innholdet på en rekke ulike formater. Mellom GPRS-modulen i hjemmenettet og proxy-gateway'en bør en derimot kun benytte en binær protokoll over TCP/UDP. Dermed sikrer en at korte meldinger sendes over GPRS hvor kostnadene er volumbaserte, og proxy-gateway'en kan sende data til GPRS-modulen når en tilkobling først er satt opp.

Ut fra proxy-gateway'en mot en fjernstyringsenhet kan en derimot presentere data på ulike formater, og over ulike protokoller. For en J2ME applikasjon som kjører på en mobiltelefon er det klare fordeler ved å benytte en binær protokoll over TCP eller UDP. En får da lite overhead på meldingene, og en får muligheten til å sende datatrafikk direkte fra proxy-gateway'en mot mobiltelefonen ved alarmer og lignende. Alternativt kan også Web Services eller en binær protokoll over HTTP benyttes, ettersom det kan forenkle implementasjonen på både server og klient, og gi bedre støtte på mobiltelefoner. Derimot medfører dette at mobiltelefonen aktivt må spørre for å få oppdatert informasjon fra proxy-gateway'en. Det blir dermed vanskeligere å implementere alarm systemer og lignende tjenester.

Et annet eksempel på presentasjonsformat, er å benytte WML dokumenter over WAP. En vil da kunne tilby mobiltelefon styring fra så å si alle GSM telefoner på markedet i dag gjennom innebygde WAP-lesere. Men dette vil gi begrensede UI og alarm muligheter. Det samme gjelder et vanlig HTML grensesnitt for stasjonære PC'er. Figur 5.9 illustrerer disse løsningene.



Figur 5.9: Protokoller og presentasjonsformater i en proxy-gateway løsning.

Sikkerhet i en løsning med en proxy-gateway må implementeres med toveis autentisering som forklart i Kapittel 5.3.2.1. Proxy-gateway'en bør benytte SSL eller TLS slik at hjemme-GW og mobiltelefon ved hjelp av sertifikatet vet at de er koblet mot korrekt server. Når en kryptert forbindelse er satt opp kan en kjent felles nøkkel kombinert med brukernavn sikre at korrekt bruker blir koblet mot korrekt hjemmenett.

5.3.2.3 Internett eller eget APN

Uavhengig av hvilken av de to presenterte serverløsningene en velger, vil en normalt benytte mobiloperatørens Internett APN, og serveren vil tilkobles Internett med en gyldig IP-adresse som det er mulig å rute trafikk til. Derimot kan en også benytte et eget APN. Dette krever en egen avtale med mobiloperatøren, men gir full kontroll over nettverket. Dermed kan en forbedre sikkerheten og tjenestekvaliteten. Sikkerheten økes fordi en kan sette brukernavn og passord for å komme inn på APN'et. Dermed blir antallet som har direkte tilgang til å kontakte hjemme-GW, mobilterminal og den sentrale serveren langt mindre, i forhold til at de er fritt tilgjengelige på Internett. Hvis en løsning med en proxy-gateway benyttes, vil en likevel ha mulighet for å tilby interaksjon mot Internett noder, uten at sikkerheten senkes betraktelig, ved at serveren også har tilgang til Internett. Tjenestekvaliteten kan også forbedres ettersom kapasiteten på nettverket bedre kan skaleres ut i fra antall brukere.

Uavhengig av hvilken løsning en benytter, kreves det en server som må være svært stabil ettersom den må være tilgjengelig til en hver tid. Derimot vil belastningen på en proxy-gateway være langt større enn en lokasjonsserver. Dette er ikke nødvendigvis et stort problem, fordi det er et område hvor en relativt enkelt kan skalere kapasiteten etter hvert som antallet brukere øker.

Vi har i disse løsningene stort sett kommentert GPRS. Det er derimot ikke noe i veien for å også benytte GSM som bærer. De fleste mobiloperatører tilbyr Internett tilgang også via GSM-data. I en løsning hvor en har en server på Internett er det derfor opp til brukeren hva han/hun ønsker å benytte.

5.3.2.4 Implementasjon

Ettersom vi i denne oppgaven fokuserer på applikasjonsutvikling for mobilterminaler, ser vi her på selve implementasjonen av disse løsningene på mobilterminalen. Mot mobiltelefonen er det aktuelt å basere kommunikasjonen på enten en binær protokoll direkte over TCP/UDP, interface kall via Web Services (over HTTP/WAP), eller direkte over HTTP/WAP. For alle disse kan en tolke applikasjonsdataene, og presentere de i et eget J2ME program.

I J2ME er GCF en felles abstraksjon mot de ulike kommunikasjonsteknologiene. Ved å benytte URI for å velge kommunikasjonsprotokoll, destinasjon og andre parametre, benyttes samme interface for å opprette en tilkobling. I MIDP 1.0 er det kun kommunikasjon over HTTP som er obligatorisk å støtte i J2ME implementasjonen. Dette er fordi mange mobiltelefoner kun har tilgang til HTTP forespørsler gjennom WAP. Ved å benytte HTTP til dataoverføring, vil en derfor sikre at applikasjonen fungerer på alle mobiltelefoner som støtter J2ME.

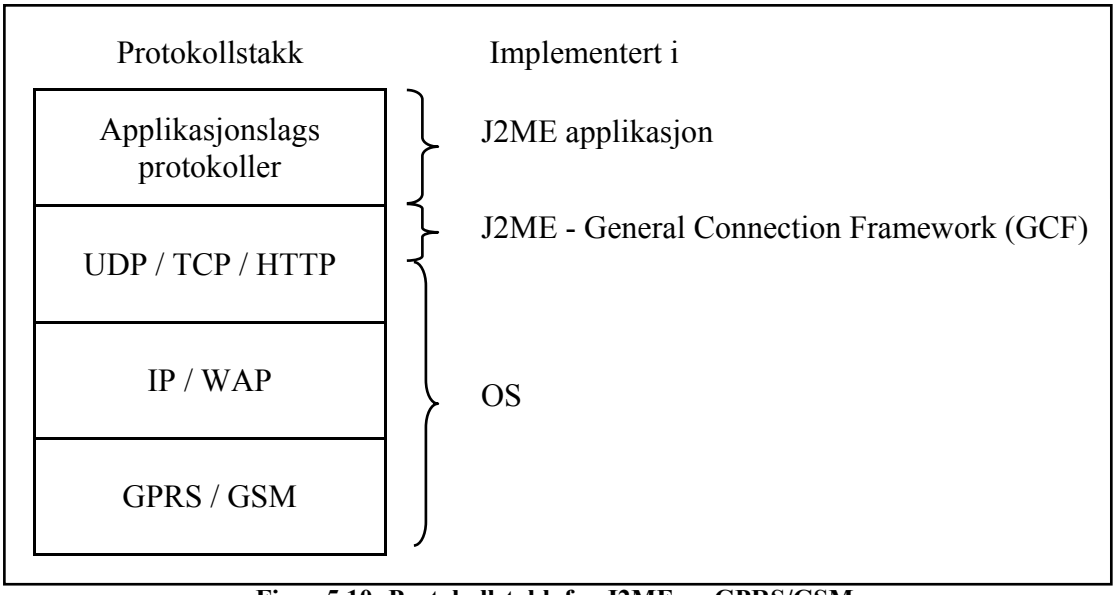
HTTP har to typer meldinger for sende spørringer og data til en server. I en GET melding kan en sende parametere i selve URL'en som sendes til serveren. Dette er tilsvarende som ofte gjøres på søkemotorer på Internett, hvor en ser parameterene i URL'en. Eksempel på dette er "http://www.smartenergy.no/getStatus?*UserID=bruker&Password=passord&Unit=Ovn1&ID=3*".
For å skjule parameterne i noe større grad og for å kunne sende større mengder data, kan også en egen POST melding benyttes. Ved POST sendes parameterne i headeren på meldingen og ikke i URL'en. Selv om HTTP v1.0 er en tilstansløs protokoll, finnes det metoder for at flere sekvensielle meldinger kan utveksles gjennom å benytte cookies og sessions. Uansett er en begrenset av at HTTP protokollen kun er pull basert og krever at mobiltelefonen aktivt spør etter oppdatert data.

Ettersom også Web Services er basert på HTTP trafikk, vil det være mulig å legge inn ekstra biblioteker for å støtte dette uten at det krever noe spesielt ved J2ME implementasjonen. kSOAP [41] er et åpent bibliotek som implementerer en del av Web Services standarden (v1.1), og den tar om lag 42 kB å implementere i en MIDlet. Sun Microsystem jobber likevel med et offisielt API for Web Services. Det er blitt spesifisert under JSR-172, og forventes å bli implementert som en del av J2ME på nyere mobiltelefoner.

I nye MIDP 2.0 er også spesifisert SSL/TLS støtte som er valgfritt å implementere. Dette gir mulighet for å enkelt benytte HTTPS for å autentisere serveren en kobler til, samtidig som en får kryptering av pakker som sendes. Også ved bruk av sockets i MIDP 2.0 har en mulighet for å benytte kryptering og autentisering gjennom TLS.

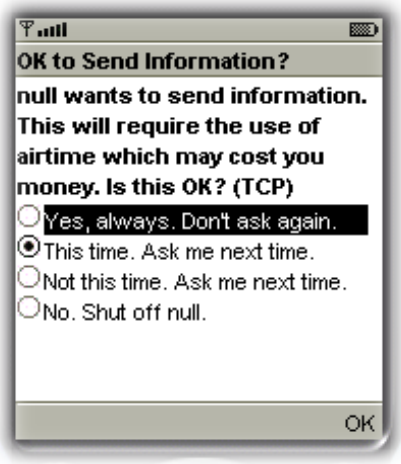
Ved introduksjonen av MIDP 2.0 ble også støtte for TCP sockets og UDP datagram via GCF spesifisert som valgfritt å implementere. Enkelte telefoner har derimot hatt støtte for dette en stund, selv med MIDP 1.0. Men ettersom dagens spesifikasjon ikke var endelig da disse ble implementert følger ikke alle den nye standarden. Dermed mister fordelen av plattformuavhengighet hvis en utvikler kun mot disse. Uansett begynner støtten for TCP og UDP å bli vanlig på en rekke implementasjoner av J2ME [42]. Dette gir oss mulighet for å både koble til og lytte for TCP og UDP kommunikasjon så lenge mobiltelefonen er koblet til et IP-nettverk og har en IP-adresse.

API'ene i J2ME baserer seg på høynivå kall og prøver så langt det er mulig å holde seg borte fra å spesifisere tilgang til kall som kanskje vil være vanskelig å implementere likt på ulike hardwareplattformer. For nettverkstilkoblinger som GPRS og GSM eksporters derfor kun tilgangen til for eksempel TCP sockets og HTTP protokollen via GCF. Det er derfor opp til mobilterminalens OS å utføre selve tilkoblingen og spesifisering av APN, telefonnummer, brukernavn, passord og lignende ved GPRS/GSM tilkoblinger. For at ikke ødeleggende programvare skal kunne utnytte kostbar kommunikasjon også J2ME spesifisert til å aktivt spørre brukeren når en MIDlet prøver å benytte mobilkommunikasjon.



Figur 5.10: Protokollstakk for J2ME og GPRS/GSM.

Som Figur 5.10 viser tilbyr GCF tilgang til UDP, TCP og HTTP. OS'et står for å tilby GCF støtte for disse og en applikasjon kan ikke styre GPRS, GSM eller WAP spesifikke parametere. Når en applikasjon prøver å benytte radiokommunikasjon vil derfor en egen kontrollfunksjon i implementasjonen av J2ME overstyre applikasjonen, og spørre brukeren om programmet skal få tilgang til ressursen. I tillegg vil brukeren bli spurt om spesielle paramertre som krevers for kommunikasjonen som skal benyttes, dersom dette kreves. I Figur 5.11 ser vi hvordan Sun sin emulator demonstrer hvordan en J2ME plattform bør gi ulike muligheter for å klarere et program for nettverkstilgang.



Figur 5.11: OS interaksjon ved nettverksforespørsel fra en J2ME applikasjon.

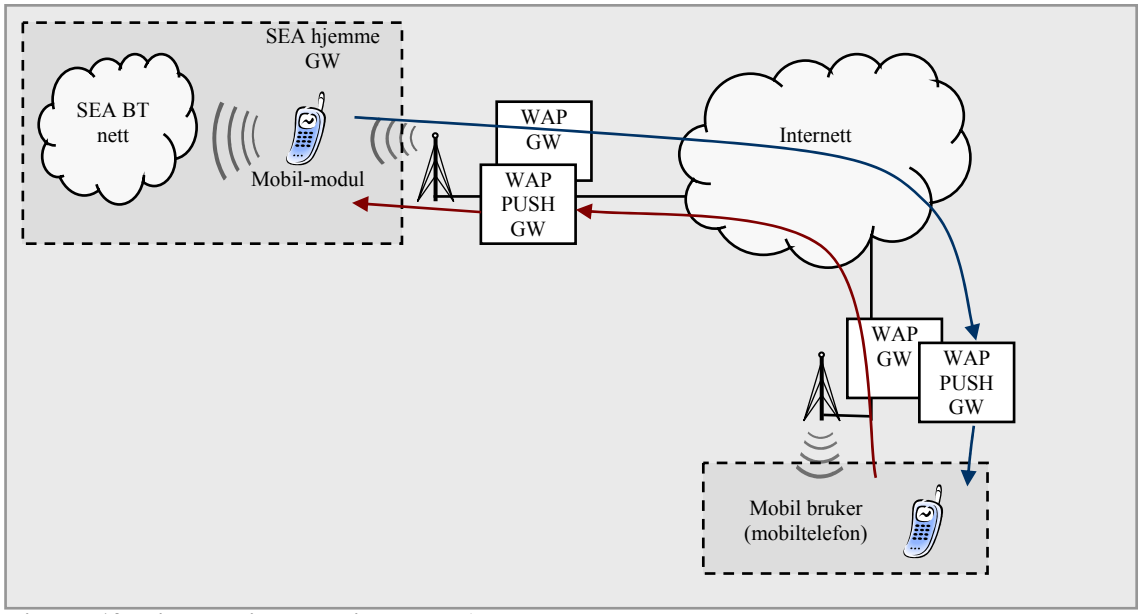
Vår erfaring med ulike J2ME implementasjoner viste at denne kontrollfunksjonaliteten er implementert svært ulikt fra leverandør til leverandør. Imot Sun sine anbefalinger gir for eksempel enkelte implementasjoner ingen muligheter for å tillate en klarert applikasjon å alltid få tilgang til mobiltjenester. Dermed kan det være vanskeligere for en utvikler å forutse hvor brukervennlig en applikasjon som ofte aksesserer nettverksressurser vil være.

En J2ME applikasjon fungerer som en tilleggstjeneste til andre tjenester på en mobilterminal. En bruker er derfor sjelden interessert i å ha en J2ME applikasjon som konstant kjører for å kunne motta alarmer eller lignende. I MIDP 2.0 har en derfor fått utvidet funksjonalitet for å automatisk starte programmer ved innkommende data. Denne funksjonaliteten kalles MIDP-Push og tillater at et program ved installasjon eller under kjøring, kan legge inn muligheten for å bli automatisk startet av OS'et ved innkommende TCP, UDP, SMS, eller annen datakommunikasjon.

I en løsning hvor vi ønsker å benytte SMS for å få applikasjonen til å utføre en tilkobling mot en server kan vi altså benytte MIDP-push dersom applikasjonen ikke er aktiv. I J2ME er støtte for SMS spesifisert i WMA som tidligere er kommentert i Kapittel 5.3.1.2.

5.3.3 Alternative løsninger

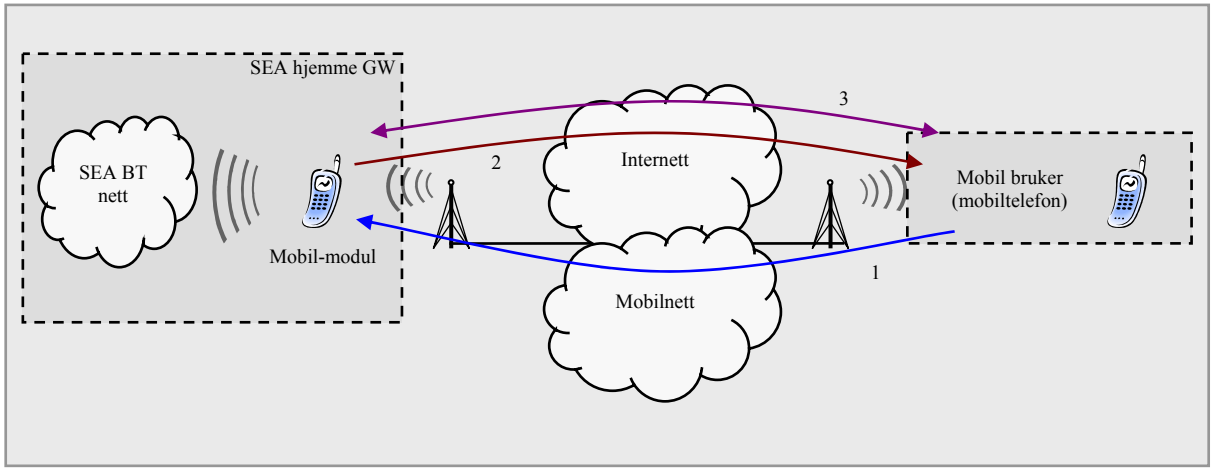
Som beskrevet i Kapittel 2.1.6 finnes det en løsning på problemet med å initialisere trafikk fra en node på det eksterne nettverket mot en mobiltelefon. Spesielt når WAP benyttes kreves en løsning, ettersom en kun kan benytte HTTP basert pull mot nettverket. Løsningen er WAP Push protokollene som tillater at en melding sendt fra en node på Internett videresendes til mobiltelefonen basert på telefonnummer som adressering. Meldingen sendes først til en WAP Push server hos mobiloperatøren som videresender denne over SMS eller GPRS. En løsning kunne derfor være å benytte WAP Push i begge retningene mellom en hjemme-GW og en mobiltelefon. En vil da kunne få en løsning tilsvarende SMS, men med mulighet for å utnytte GPRS tilkobling når dette er tilgjengelig. Løsningen er illustrert i Figur 5.12 og WAP Push GW'en kan sammenlignes med proxy-gateway'en i Kapittel 5.3.2. Forskjellen er at mobiloperatøren står for serveren som videresender meldingene.



Figur 5.12: Fjernstyring ved hjelp av WAP Push.

Løsningen er ikke enkel å benytte i dagens nett. I Norge er støtten for WAP Push begrenset. For å få tilgang til dette kreves det en egen avtale hos Telenor [43]. En hver bruker måtte derfor hatt en individuell avtale for å få tilgang til WAP Push tjenesten. J2ME støtten for å behandle mottatte WAP Push meldinger er også begrenset.

En annen alternativ løsning er en blanding av SMS og GPRS løsningene vi har presentert. SMS kan benyttes for å unngå problemet med dynamisk adressering i Internett tilgangen, ved at tildelt IP-adresse sendes til motstående part i en SMS. Når mottakersiden kjenner adressen kan den opprette en TCP-socket tilkobling mot avsenderen, og trafikk kan gå i begge retninger så lenge brukeren ønsker å holde oppe tilkoblingen. En vil da kunne tilby et mer dynamisk oppdatert UI over GPRS i forhold til en ren SMS basert løsning. En slipper også en adresseringsserver eller lignende. Et problem er at en likevel er begrenset av tregheten i SMS systemet som kan gjøre at det vil ta noe tid å få satt opp kommunikasjonen. Se Figur 5.13 for en illustrasjon av meldingsforløpet, der mobilterminalen initialiserer trafikken.



Figur 5.13: Fjernstyring ved hjelp av SMS til oppstart av GPRS kommunikasjon.

[1] Den siden som ønsker å opprette en tilkobling sender en SMS med sin internettadresse.
[2] Mottaker av SMS kan ta kontakt med sender via GPRS/Internett, og kommunikasjonen kan starte.
[3] Meldingsutveksling foregår over GPRS med TCP eller UDP som transportprotokoll.

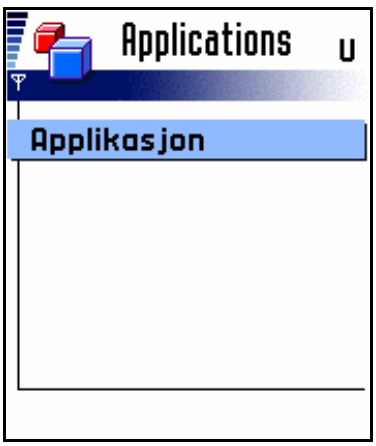
Implementasjon av en slik løsning blir tilsvarende det beskrevet under Kapittel 5.3.1.2 og 5.3.2.4. En J2ME applikasjon vil kunne starte en GPRS tilkobling ved å gjøre en tilkobling mot en kjent server eller mot lokal adresse. Dette vil få mobiltelefonen til å gjennomføre en tilkobling og gir mulighet for at en kan hente ut lokal IP-adresse som kan sendes via SMS til hjemme-GW'en.

GSM-data tilbyr også muligheten for å gjøre en tilkobling direkte i mobilnettet mellom en mobil og en GSM-modul. Dette er en mulighet vi ikke har i GPRS. I J2ME kan derimot en slik GSM løsning være vanskelig å implementere, ettersom det ikke er spesifisert standardiserte API'er for å kontrollere dataoverføring i en slik tilkobling.

5.4 Brukergrensesnitt

De to viktigste overordnede egenskapene til brukergrensesnittet (UI) er brukervennlighet og portabilitet. Vi må skape et UI hvor disse egenskapene gjennomsyrer alle aspekter ved løsningen.

Når applikasjonen er installert på mobiltelefonen, vil den være tilgjengelig for brukeren som et ikon eller et listevalg, som illustrert i Figur 5.14, avhengig av applikasjonsplattformen. Brukeren kan vanligvis starte applikasjonen ved å merke den i applikasjonsmenyen og trykke velg-knappen på mobiltelefonen.

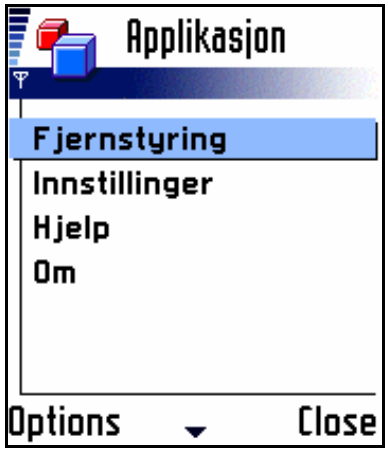


Figur 5.14: Applikasjonsmenyen etter installasjon på et menybasert OS.

Det første brukeren vil se når applikasjonen starter opp, er applikasjonens hovedskjerm. Applikasjonen vil ikke ha noen splash screen siden dette er mest nyttig for trial applikasjoner [6]. Hovedskjermen kan se ut som på Figur 5.15, og vil linke til alle deler av programmet. For å gå videre i programmet fra hovedskjermen, velger brukeren fra lista. Det er viktig at en liste som benyttes som navigasjonsmeny er av type IMPLICIT, ellers vil brukeren bli ledet til å tro at menyen er en liste med innstillinger [6].

Menyen i Figur 5.16 har, i tillegg til ”Velg” og ”Exit”, også kommandoer for ”Hjelp” og ”Om”. ”Hjelp” tilbyr brukeren hjelp for den aktive skjermen. ”Om” gir brukeren informasjon om programmet, og dette valget er tilgjengelig fra alle deler av applikasjonen. Dette kan være informasjon som er nødvendig hvis brukeren behøver teknisk hjelp, som applikasjonens navn, versjonsnummer og kontaktinformasjon.

Alle kommandoer assosieres med en kommandotype og prioritet [6]. For hver kommandotype, avhengig av mobiltelefonen, kan kommandoen med høyest prioritet assosieres med en knapp på mobiltelefonen. I tilfellet Figur 5.16, er kommandoen ”Velg” av typen ITEM, og har høyeste prioritet. Det vil si at dersom mobiltelefonen har en velg-knapp, assosieres ”Velg” med denne, og brukeren kan trykke velg-knappen for å gjøre et valg på hovedskjermen uten å åpne kommandomenyen. Hvis ikke mobiltelefonen har en velg-knapp, må brukeren først trykke en softkey, her merket med label’en ”Options”, for å åpne kommandomenyen, og deretter velge ”Velg” fra menyen ved hjelp av en softkey. Alle J2ME MIDP kompatible mobiltelefoner har to softkeys som assosieres med de to valgene som ses nederst på skjermen i Figur 5.15 og Figur 5.16.



Figur 5.15: Eksempel på listebasert hovedskjerm i en mobilapplikasjon.

Figur 5.16: Eksempel på kommandomeny.

En kommando label bør ikke være lengre enn 6 karakterer. Årsaken til dette er at mange mobiltelefoner ikke har plass på skjermen til å vise flere karakterer. I tilfelle en kommando label kun skal vises i en meny, er det trygt å anta at mobiltelefonen kan vise minst 20 karakterer. I MIDP 2.0 kan en derimot spesifisere én label for visning på meny, og én kortere label som softkey label. [6] Dersom applikasjonen skal være kompatibel med MIDP 1.x enheter, må MIDP versjonen detekteres slik at riktig label vises.

Ifølge Little Springs Design [6], bør enhver kommando assosieres med en egen unik label. Og label'en må være tilstrekkelig deskriptiv, slik at brukeren kan forutsi nøyaktig hva kommandoen vil utføre, det vil også si at en bør unngå forkortelser og ambiguitet. Enkelte labels er reserverte, og bør ikke benyttes, og i tillegg bør ikke labels som "Tilbake" benyttes. Minst én kommando for å gå videre i programmet bør defineres for hver skjerm. Dette kan være en "Utfør" eller "Ferdig" kommando av type SCREEN hvis brukeren er kommet til slutten av en veiviser eller lignende. Ellers kan det være kommandoer av typen ITEM eller OK. Én eller flere kommandoer av typen BACK bør også defineres for hver skjerm slik at brukeren kan komme seg tilbake til forrige skjerm i en prosess, eller helt tilbake til hovedskjermen. Kommandoen som returnerer brukeren til forrige skjerm bør ha høyest prioritet fordi denne ofte assosieres med en hardware knapp på mobiltelefonen. I tilfeller da brukeren ikke bør returneres til forrige skjerm, for eksempel hvis forrige skjerm var en passord skjerm, bør BACK kommandoen med høyest prioritet føre til en annen logisk destinasjon.

En kommando av typen EXIT bør defineres for hver skjerm i applikasjonen. Denne kommandoen skal sikre at brukeren alltid kan forlate en prosess, og også sikre at enhetens exit knapp fungerer slik brukeren forventer. Mobiltelefonens primære softkey bør hovedsakelig benyttes til navigering i applikasjonen. Hvis applikasjonen trenger mer enn to funksjoner, bør de resterende kombineres i en meny som assosieres med mobiltelefonens sekundære softkey, som på Figur 5.16.

Eventuelle grafiske effekter i applikasjonen må designes slik at de passer innenfor skjermstørrelse på alle enheter applikasjonen vil kjøre på. Grafikken må passe horisontalt på skjermen uten å skalere eller skrolle. Grafiske effekter som ikoner og logoer, som ikke er hovedhensikten med innholdet på skjermen, bør være så små som mulig for å spare minne, og fordi de kan avspore brukeren og virke forvirrende. Kantutjevning bør ikke benyttes på tekst i bilder, siden kantutjevning vil gjøre teksten diffus på skjermer med lav oppløsning. [6]

Det finnes en mengde andre gode retningslinjer for hvordan et brukergrensesnitt bør bygges opp. Vi anbefaler å lese "User Interface Design Guidelines for J2ME MIDP 2.0" [6] av Little Springs Design.

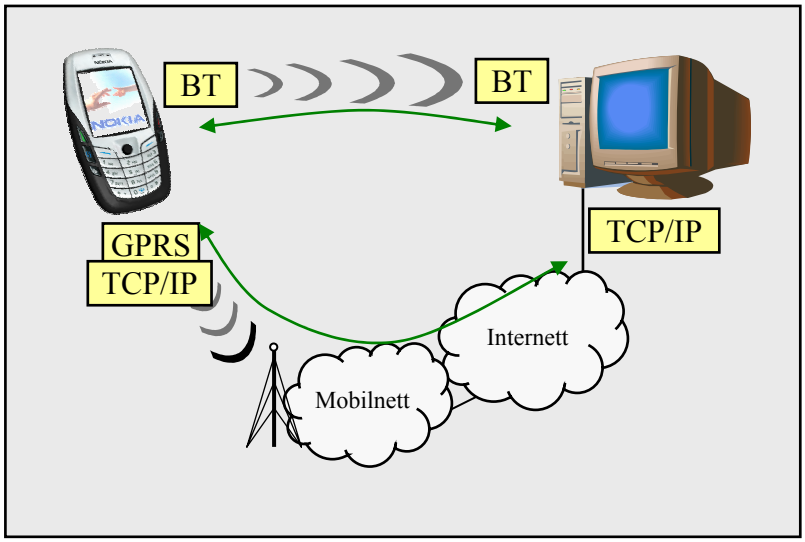
Contextual design kan i mange tilfeller være en passende metode for systemutviklere ved design av brukergrensesnitt. Denne metoden har flere fordeler. Systemutviklernes ubevisste handlinger konkretiseres og gjøres eksplisitte, og utviklingsprosessen får flere målbare trinn. Disse kan en senere analysere. For å utvikle et best mulig design for applikasjonen, bør contextual design vurderes.

6 Demonstrator

6.1 Funksjonalitet

Vi hadde få muligheter for å demonstrere komplette løsninger av de arkitekturer vi har foreslått i oppgaven. Derfor valgte vi å fokusere på en demonstrator som kunne vise at J2ME kan benyttes til å tilby et standardisert UI, uavhengig om en benytter mobiltelefonen som lokalstyring i Bluetooth nettet eller via andre nettverk.

Vi hadde ikke tilgang til flere mobiltelefoner eller en GPRS-modul hvor vi kunne implementere funksjonaliteten av en GW i hjemmenettet. Vi valgte derfor å benytte en Windows PC med tilgang til Internett og Bluetooth til å emulere en GW og Bluetooth enhet.



Figur 6.1: Oppsett av demonstrator.

Dermed kunne vi utvikle en J2ME applikasjon som ved å benytte TCP/IP trafikk over GPRS eller Bluetooth, kunne kontakte det samme programmet på PC. Ved å designe J2ME applikasjon slik at den automatisk bytter til GPRS når en Bluetooth tilkobling ikke er tilgjengelig, kunne vi teste et felles UI for både lokal- og fjernstyring av PC applikasjonen. Samtidig ville vi får en indikasjon på hvor stor forsinkelse en kan forvente seg i en løsning som benytter datakommunikasjon over GPRS.

For meldingsutveksling på applikasjonsnivå utviklet vi en enkel binær protokoll, som er identisk om data sendes over Bluetooth SPP eller TCP/IP. I sammenheng med SEA sitt smarthusnett, ble denne protokollen basert på meldinger for å slå av og på lys, samt overføring av temperaturmålinger og lignende.

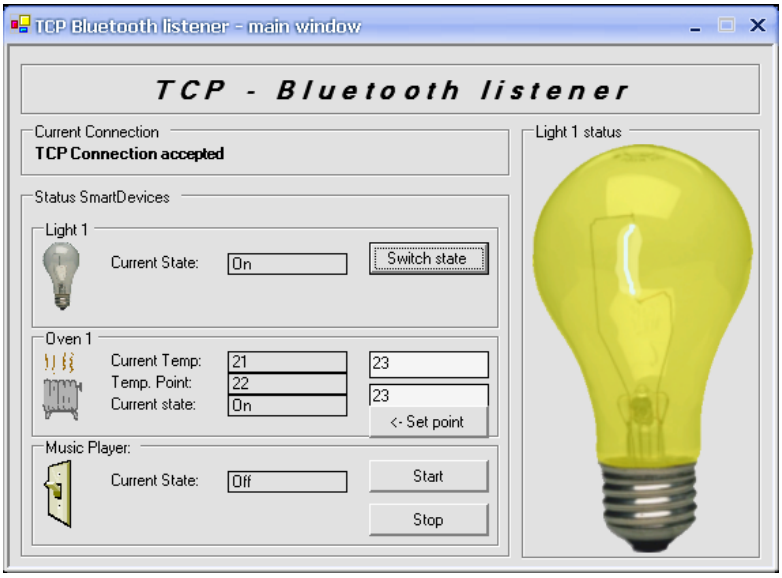
Vi ønsket også å teste ut OTA funksjonaliteten i J2ME, for å se hvordan implementasjonene av denne teknologien fungerer på de ulike mobiltelefonene.

6.2 Implementasjon

Følgende deler ble utviklet for å teste funksjonaliteten for Bluetooth og GPRS kommunikasjon i en J2ME applikasjon. For kildekode, se Vedlegg A.

6.2.1 Server applikasjon

En server applikasjon ble utviklet i C# i utviklingsverktøyet Visual Studio .Net for å raskt kunne få opp en lytter til meldinger på Bluetooth og TCP/IP.



Figur 6.2: Windows server applikasjon som lytter til Bluetooth og TCP/IP kommunikasjon.

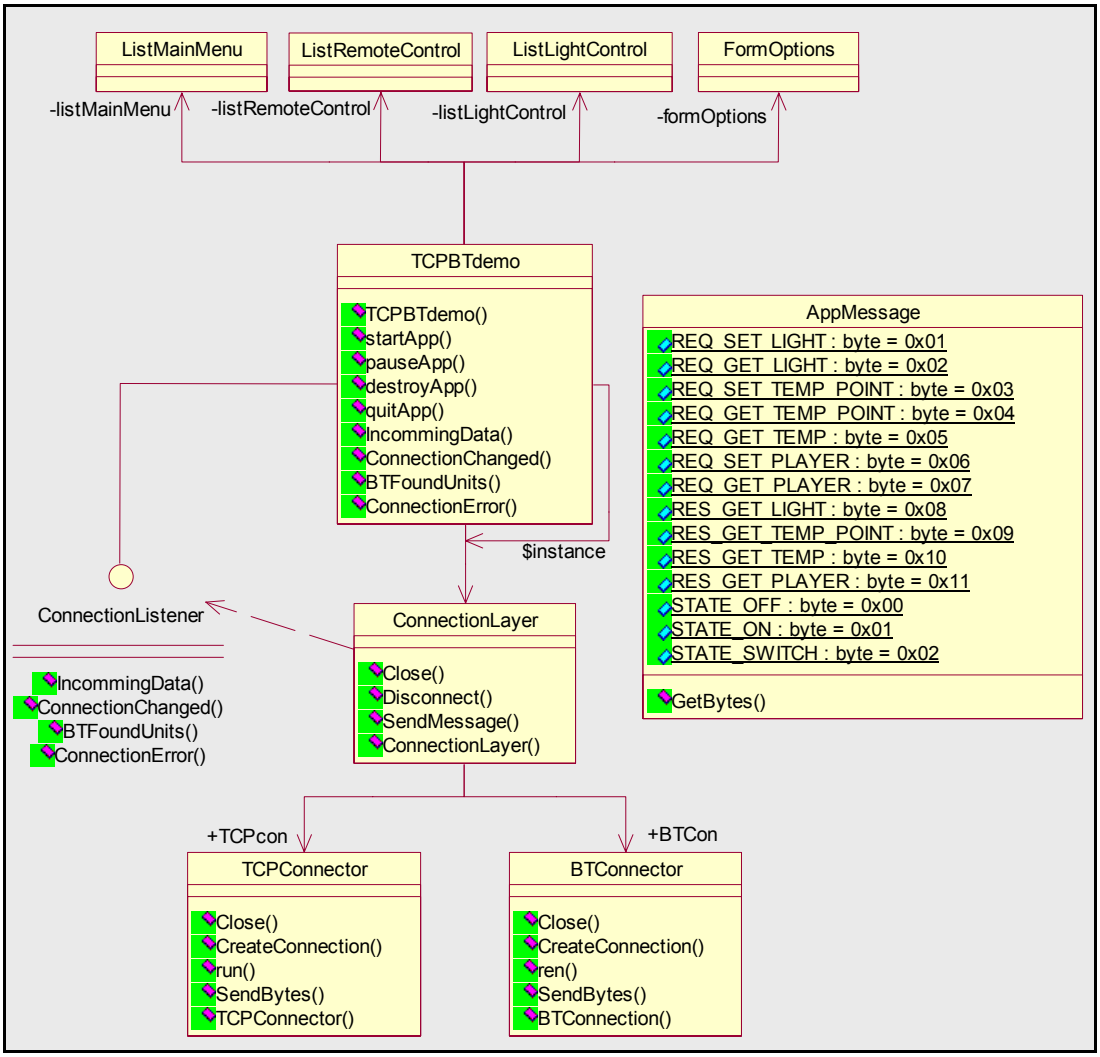
Bluetooth støtten i Windows er i dag svært begrenset. Microsoft har ingen offisiell støtte for utvikling mot Bluetooth enheter, noe som gjør det vanskelig å utvikle mot disse uten å kjøpe dyre tredjepartsimplementasjoner. I vår løsning benyttet vi derfor en COM-port emulering i programvaren til Bluetooth enheten på PC'en. Dermed fikk vi tilgang til å benytte COM-port støtte i .Net. Dette gir selvsagt ingen muligheter for å gjennomføre søk eller tilkobling på Bluetooth nivå via applikasjonen, men dette er noe vi ikke var avhengige av i denne demonstratoren. Programvaren mottar SPP tilkoblinger, styrer pairing og kryptering og setter opp en emulert COM-port for kommunikasjon.

Det ble ikke implementert meldingshåndtering for alle typer styring som vist i Figur 6.2 selv om protokollen som benyttes er spesifisert til å kunne styre alle enhetene, var det kun lysttyring som var nødvendig for å få utføre de testene vi skulle.

6.2.2 J2ME applikasjon

For utvikling av J2ME applikasjonen benyttet vi utviklingsmiljøet Borland JBuilder X som har innebygd støtte for utvikling av J2ME programmer. En SDK[1] fra Nokia ble benyttet for å få tilgang til JSR-82 API'et som ikke blir levert med som standard i Sun sin SDK.

For å kunne tilby et felles UI ble programmet bygget opp med en kommunikasjonsmodul som fungerer som et abstraksjonslag mot kommunikasjonsmediene. Når J2ME applikasjonen krever en tilkobling, sendes det en melding til kommunikasjonslaget. Kommunikasjonslaget vil så først prøve å benytte Bluetooth kommunikasjon. Dersom dette ikke lykkes gjøres et nytt forsøk på tilkobling over TCP socket.



Figur 6.3: UML klassediagram over J2ME demonstrator.

[1] Series 60 MIDP Concept SDK Beta 0.3.1, Nokia edition (oktober 2003) ble benyttet mest til kompilering av J2ME applikasjonen.

Meldingshåndtering ble også utviklet i en egen modul, og denne kan benyttes uavhengig av hvilket kommunikasjonsmedie som benyttes. Ved tilkobling over Bluetooth gjøres et kall mot GCF med en spesiell URI for å få gjennomført en SPP tilkobling. Lykkes tilkoblingen returneres en streamconnection[1]. Gjennom inquiry og service search kan en søke etter kompatible Bluetooth enheter og åpne RFCOMM porter, som kan benyttes til å bygge opp denne URI'en. For koden i Vedlegg A er inquiry og service search ikke implementert ettersom testing av dette ikke var formålet med demonstratoren. Begge deler er derimot blitt benyttet i testapplikasjoner mot SEA sine enheter og fungerer bra via de interfaces som JSR-82 eksporterer.

Tilsvarende kan en annen URI benyttes i GCF for å gjennomføre en TCP tilkobling[2]. Lykkes tilkobling mot en IP-adresse med tilhørende portnummer, returneres en streamsocket som applikasjonen kan overføre datatrafikk over. Denne er så å si identisk med en Bluetooth SPP stream og oppbygningen av applikasjoner som benytter ulike kommunikasjonsteknologier kan derfør gjøres svært likt i J2ME.

6.2.3 OTA

OTA teknologien i J2ME åpner for at en MIDlet suite kan lastes ned over WAP eller annen nettverkskommunikasjon. Dette ble testet ved å lage en enkel WAP side, hvor en legger ut en link til JAD fil som beskrevet i Kapittel 5.1.

6.3 Resultater

Under utvikling fikk vi testet demonstratoren på tre ulike mobiltelefoner[3], som alle støtter MIDP 2.0, og har JSR-82 implementert. Vi opplevde varierende resultater mellom de ulike mobilterminalene.

Implementasjonen av UI fungerte etter intensjonen, og tilpasset seg bra til standard UI'et på de enkelte mobiltelefonene. På Sony Ericsson P900, som har et langt større display en de andre, ble for eksempel softkeys vist som GUI knapper på skjermen, i stedet for faste knapper på mobiltelefonen. Dette tilsvarer hvordan mobiltelefonens UI fungerer til vanlig.

[1] GCF Bluetooth URI i demonstrator:
"*(StreamConnection) Connector.open("btspp://" + bluetoothAddress + ":" + port + "; authenticate=false; encrypt=false; master=false");*".
[2] GCF TCP socket URI i demonstrator:
"*(SocketConnection) Connector.open("socket://" + serverAddress + ":" + port)*".
[3] Under utvikling ble applikasjonen testet på Ericsson P900, Nokia 6600 og Nokia 6230.

Når en benytter kommunikasjonsmedier, er det derimot BCC og OS'et som tar over interaksjonen med brukeren. Dette kan ikke styres fra J2ME. Implementasjonen av denne funksjonaliteten var svært forskjellig på de ulike mobiltelefonene. Enkelte av implementasjonene tilsvarte emulatoren fra Sun, som vist i Figur 5.11, med gode muligheter for å lagre rettighetene en applikasjon skal ha til kommunikasjonsteknologier, mens andre ikke hadde noen mulighet for dette. Av denne grunn må en bruker aktivt tillate en J2ME applikasjon rettigheter for hver gang et kommunikasjonsmedie skal benyttes.

OTA fungerte derimot korrekt og likt på alle mobiltelefonene. Under installasjon blir en spurt om å beholde lagrede data, dersom dette eksisterer på mobiltelefonen fra tidligere versjoner av programmet.

En av de større utfordringene under utviklingen, var forskjeller i J2ME implementasjonene på de enkelte mobiltelefonene. Alle implementasjonene for de mobiltelefonene vi testet, hadde enkelte kjente problemer med noen av de støttede API'ene. En applikasjon utviklet etter MIDP standarden, vil derfor fort kunne feile når den benytter funksjoner og moduler som ikke er korrekt implementert i mobiltelefonens J2ME versjon. Dette var heller ikke mulig å teste ved å kjøre applikasjonene på de ulike PC emulatorene som finnes for hver mobiltelefon. Disse er så å si alltid basert på Sun sin emulator, og vil med unntak av UI, kjøre applikasjonene likt.

Applikasjonen som ble utviklet fungerte etter formålet, og er i stand til å benytte både GPRS og Bluetooth kommunikasjon mot PC applikasjonen. Ved bruk opplevde vi en forsinkelse mellom ett til to sekunder før J2ME fikk beskjed om at en Bluetooth tilkobling var vellykket eller ikke. Ved Bluetooth kommunikasjon, var forsinkelsen ved dataoverføring ikke merkbar, mens for GPRS lå den i overkant av to sekunders RTT[1]. Selv om demonstratoren utviklet i denne oppgaven ikke emulerer en komplett arkitektur i forhold til de løsningene vi har presentert, mener vi dette tilsvarer den forsinkelsen en normalt vil kunne oppleve ved å benytte GPRS kommunikasjon i disse løsningene.

[1] Round Trip Time. Tiden det tar å sende en melding fra en node til en annen, og tilbake igjen.

7 Drøfting

7.1 Innledning

Vi vil i dette kapitlet diskutere våre løsninger, drøftinger og konklusjoner fra tidligere i paperet. Kapitlet representerer våre tanker og meninger om de løsninger og resultater vi har funnet for applikasjonsutvikling for mobiltelefoner. I sammenheng med styring av SEAs Bluetooth nettverk fra mobilterminaler, var det viktig å kartlegge aktuelle kommunikasjonsteknologier, programmeringsspråk og applikasjonsplattformer på mobilterminaler.

Aktuelle kommunikasjonsteknologier delte vi i løsninger for kort og lang rekkevidde, og så på aktuelle kommunikasjonsmedier for hver av disse. I tillegg evaluerte vi høyere kommunikasjonslag som TCP, WAP/HTTP og Web Services, ettersom disse er aktuelle å benytte i en fjernstyringsløsning.

For best å kunne benytte en mobilterminal i en slik sammenheng, må vi ha mulighet for å utvikle og installere egen programvare på mobilterminalen. Vi evaluerte derfor en rekke av de mest aktuelle applikasjonsplattformene som finnes på markedet i dag. Disse undersøkte vi med tanke på brukervennlighet, og ikke minst med tanke på utviklingsmuligheter i forhold til ressursutnyttelse innen kommunikasjonsmedier, UI og plattformuavhengighet.

Til slutt satte vi disse vurderingene sammen og presenterte mulige løsninger med J2ME som applikasjonsplattform. Disse løsningene ble basert på ulike kommunikasjonsteknologier og mulige arkitekturer for lokal- og fjernstyring av SEAs Bluetooth nettverk. Vi vil i denne delen drøfte egenskapene til disse løsningene, for å kunne konkludere med hvilke muligheter som finnes for å benytte tredjepartsprogramvare på mobilterminaler i ulike kommunikasjonsløsninger.

7.2 Kommunikasjonsteknologier

Vi har tidligere, i Kapittel 2.1.8, vurdert 802.11 standardene og Bluetooth for lokalstyring av SEAs hjemmenett. Mens Bluetooth har fordeler innen strømforbruk, pris, oppkobling og fysisk størrelse, har WLAN fordeler innen båndbreddekapasitet og rekkevidde. Sikkerhetsmessig er begge teknologiene nokså sidestilte med hverandre, men det kan likevel nevnes at Bluetooth krypteringsmessig er et hakk bedre tilpasset små mobile enheter.

Rekkevidden til en Bluetooth enhet kan være opptil 100 meter, som også er rundt den maksimale rekkevidden for 802.11 enheter. Derimot vil de fleste Bluetooth implementasjoner ikke være beregnet for mer enn 10 meters rekkevidde med hensyn til strømforbruk. Slike hensyn til strømforbruk er ikke tatt i 802.11 standardene, og dette fører naturligvis til kortere batterilevetid for slike enheter. Dette er hovedårsaken til hvorfor WLAN er lite utbredt for mobiltelefoner.

Båndbredden for WLAN er vesentlig høyere enn for Bluetooth. Selv om dette er en fordel under overføring av mye data, er det ikke nødvendigvis en stor fordel for små mobile enheter. De fleste mobile applikasjoner er beregnet for atskillig lavere hastigheter enn hva Bluetooth tilbyr, og i SEA sitt tilfelle, vil det ikke være nødvendig med hastigheter på WLAN nivå.

GSM, GPRS, SMS og WAP er aktuelle teknologier for fjernstyring av SEAs hjemmenett. Hastighetsmessig har GPRS høyere båndbredde enn GSM. SMS og WAP kjører over GSM eller GPRS, og båndbredden er deretter.

GSM og SMS benytter globale telefonnummer for adressering. GPRS løsninger er normalt IP-baserte, og er derfor ikke adresserbare ved hjelp av telefonnummer fordi de ikke har kjennskap til lavere lag. De kan heller ikke adresseres ved hjelp av IP fordi det som regel opereres med dynamisk tildelte IP-adresser. Dette kan løses ved hjelp av en lokasjonsserver. En annen mulig løsning er å kjøre WAP over GPRS, og benytte WAP Push for å kunne adressere mobiltelefonen ved hjelp av dens telefonnummer.

Selv om WAP Push kan virke som en god løsning på adresseringsproblemet i GPRS, er det dårlig tilrettelagt for WAP Push hos de fleste mobiloperatører. WAP kan også benyttes over GSM eller SMS, men adresseringen må uansett løses ved hjelp av WAP Push. En WAP løsning er dessuten basert på HTTP, noe som implementasjonsmessig kan være tungvint for en del løsninger.

Prisingen av GSM, GPRS og SMS tjenestene er vidt forskjellig. Mens SMS har en fast enhetspris, prises GSM for tiden mobiltelefonen er tilkoblet nettverket, og GPRS for mengden data som overføres mellom mobiltelefon og nettverk. Prisingen av WAP tjenester er avhengig av underliggende teknologi.

Dersom en sender lite data, vil SMS være en enkel og rimelig løsning. Derimot vil det raskt bli dyrt dersom kommunikasjonsprotokollen mot hjemmenettet krever sending av flere SMS'er. En SMS løsning mot hjemmenettet bør derfor være statisk i form av at en gjør enkeltoperasjoner, som å hente eller sette status på enheter, istedenfor å sende fortløpende oppdateringer med informasjon om hjemmenettet til mobiltelefonen. En av fordelene med en slik løsning er at prisen blir den samme uansett mengden av data som sendes i en SMS[1].

Vi mener GPRS vil være en billigere løsning enn GSM dersom det ikke sendes for mye data over lang tid, se Figur 2.8. Sendes derimot mye data, vil GSM på sikt bli billigere enn GPRS.

[1] Mengden av data i en SMS kan ikke overskride 140 oktetter, eller 160 tegn.

Siden SMS har en upålitelig leveringstid, i tillegg til å være en kostbar løsning dersom en sender mye data, kan både GSM og GPRS oppfattes som billigere, raskere og mer pålitelig. Blant annet kan brukere av GPRS være fast oppkoblet uten å tape penger på det, og vil dermed spare tid på gjentatte oppkoblinger. Likevel kan SMS være et alternativ siden det er en enkel teknologi som er relativt lett å implementere, spesielt dersom en tidlig i utviklingen ønsker et fungerende system, om ikke annet, for test formål.

Sikkerhetsmessig er en løsning basert på GSM eller SMS god fordi både kryptering og autentisering skjer automatisk. GPRS krypterer derimot kun datatrafikken innad i mobilnettet, og kryptering og autentisering må implementeres på høyere lag. Selv om GPRS kompliserer implementeringen på grunn av dette, kan GPRS forenkle interaksjonen mot andre noder.

7.3 Applikasjonsplattformer

I Kapittel 2.2.9 drøftet vi de største og mest lovende plattformløsningene på mobilmarkedet. På basis av dette kan vi raskt utelukke et par applikasjonsplattformer som interessante i løsningssammenheng. Dette gjelder SavaJe, som ennå ikke er klar for implementasjon i mobiltelefoner, og MontaVista, som per i dag er for bagatellmessige å regne på mobiltelefonmarkedet, og dessuten ikke støtter Bluetooth. Denne situasjonen kan derimot raskt endre seg i fremtiden, som påpekt i Kapittel 2.2.3.

Ett av de viktigste kriteriene for valg av applikasjonsplattform, fra utviklers ståsted, er utvilsomt plattformuavhengighet. Selv om det kanskje vil være lettere å utvikle enkelte applikasjoner i et lavnivå språk, som har gode API'er for alle mobiltelefonens funksjoner, vil det kreve mye arbeid å få disse til å fungere på mobiltelefoner andre enn den de ble utviklet for.

Plattformuavhengighet har tre ulemper. For det første tar de fleste virtuelle maskiner stor plass i mobiltelefonens minne. Alle nyere mobiltelefoner er vanligvis godt utrustet minnemessig fordi de utstyres med minnekrevende multimediefunksjoner. For det andre krever en virtuell maskin mer av prosessoren, og dette senker programmenes potensielle hastighet. Vanligvis vil dette kun være et problem for ekstremt tunge programmer, og for spill med mye grafikk. For de fleste applikasjoner vil ikke brukeren merke noen hastighetsnedsettelse, og nye raskere løsninger er dessuten stadig under utvikling, se blant annet under Kapittel 2.2.3 om SavaJe OS. For det tredje kan mangel på direkte tilgang til hardware begrense tilgangen til mobiltelefonenes funksjonalitet. Derimot inkluderer de fleste språk, og da i høyeste grad J2ME, gode API'er for det viktigste av funksjonalitet. Blant annet får en i J2ME ikke tilgang til lavnivå nettverksfunksjoner som TCP sockets, men må programmere mot høyere lags protokoller. I de fleste tilfeller vil dette gjerne gjøre utviklingsprosessen enklere istedenfor å forvanske den.

Det er kun de tre språkene J2ME, VB og VB .NET som er i stand til å lage plattformuavhengige applikasjoner. VB er på vei ut til fordel for VB .NET. Av de to gjenværende applikasjonsplattformene, er det J2ME som støtter flest OS. I tillegg vil en være avhengig av å benytte AppForge Crossfire som utviklingsverktøy for å utvikle plattformuavhengige VB .NET løsninger.

Programmer skrevet for Mophun eller Brew kan sjelden betegnes som plattformuavhengige. I tillegg taler det til J2MEs fordel at programmer skrevet for J2ME også kan støtte en del Brew implementasjoner. Per i dag er det kun støtte for MIDP 1.0 i Brews implementasjon av J2ME, og dermed ikke støtte for Bluetooth.

Innlasting av applikasjoner er en annen viktig faktor. En god applikasjonsplattform må ha en eller annen form for et OTA installasjonssystem, som forenkler installasjonen av programmer for brukerne, og gjør det lettere for utviklerne å distribuere applikasjoner. Symbian, Mophun, Brew og J2ME har alle systemer for OTA innlasting av programvare. Symbians SiS gjør det mulig å laste ned og installere C++ programmer over blant annet WAP og HTTP. Systemene til Mophun og Brew inkluderer også et sentralt distribusjonssystem. Fra en utviklers ståsted, er den største fordelen med slike distribusjonssystemer at det forenkler salgsprosessen til applikasjonen. Pocket PC Phone Edition og Smartphone har tradisjonelle PC-lignende systemer, med nedlasting fra blant annet Internett, hvor de fungerer som normale noder og krever IP-basert kommunikasjon. Smartphone krever i tillegg sertifisering av programvare, noe som kan medføre litt ekstra arbeid for utvikler, men som er påkrevd av sikkerhetshensyn.

Applikasjonens brukergrensesnitt (UI) er viktig for brukerens persepsjon av applikasjonen. Applikasjonen bør være enkel å bruke, og det innebærer blant annet at brukeren må kunne dra kjensel på UI'et. Applikasjonsplattformen må derfor inneholde et sett med standard UI komponenter. Blant annet har J2ME og Brew slike innebygde komponenter, mens Mophun mangler standardiserte GUI komponenter. Også Pocket PC Phone Edition, Smartphone og Palm OS har standard UI komponenter. Ved å benytte forskjellige programmeringsteknikker, er det også mulig å tilpasse en applikasjons UI slik at det fungerer både på Pocket PC Phone Edition og Smartphone, men ellers er programmet begrenset til å kjøre på plattformen det er designet for.

Vanligvis tilbyr ikke lavnivå programmeringsspråk standardiserte UI komponenter. Dette er selvsagt opp til de enkelte plattformenes SDK'er å støtte. Ved heller å benytte et høynivå språk som tilbyr plattformuavhengighet, etter vår mening vil J2ME være det beste alternativet, vil en kunne stole på at VM'en tilpasser applikasjonen til mobiltelefonens UI.

7.4 Lokalstyring og kommunikasjon over kort rekkevidde

På grunn av at svært få av mobilterminalene som selges i dag støtter 802.11a/b/g standardene, tok vi tidlig et valg om å fokusere på en lokalstyringsløsning hvor Bluetooth benyttes. Ettersom SEA sitt nett er basert på Bluetooth, er det en ekstra kostnad å implementere GW'en som kreves hvis en annen kommunikasjonsteknologi benyttes. Vi vil derfor hovedsakelig drøfte hvilke muligheter som faktisk gis dersom en skal benytte Bluetooth i en applikasjon på en mobilterminal.

JSR-82 har åpnet for at mobilleverandørene kan tilby støtte for et standard sett med funksjoner og interfaces for å kontrollere Bluetooth kommunikasjon via Java. Gjennom å støtte standard profiler for å søke etter andre enheter og tjenester, samtidig som det tilbyr tilgang til de laveste kommunikasjonsprotokollene, bør det være få begrensninger i API'et i forhold til hvilke løsninger en kan benytte Bluetooth i J2ME.

Støtten for JSR-82 er begrenset på mobiltelefoner i dag. Med unntak av én mobiltelefon, er det kun mobiltelefoner basert på Symbian OS som har implementert API'et i J2ME. API'et var ferdig spesifisert i 2002, og det gis fremdeles ut nye mobiltelefoner som har både Bluetooth og J2ME, men ikke JSR-82. Samtidig er det få mobiltelefoner som i dag støtter MIDP 2.0. Denne innebærer en større implementasjonsendring, og ettersom det er vanlig å også støtte denne før en implementerer JSR-82, er det mulig at det er dette som hindrer leverandørene fra å implementere JSR-82 raskere. Alt tyder derfor på at så fort mobilleverandørene får ferdig MIDP 2.0 implementasjoner, vil JSR-82 støtte raskt følge etter, og bli en standard på alle mobiltelefoner som støtter Bluetooth.

Sikkerhet er støttet gjennom JSR-82 og gir mulighet for å kreve kryptering og autentisering for en tilkobling. Selve pairing'en og interaksjon med bruker er derimot styrt av BCC utenfor J2ME. Det viser seg derfor at det kan være eksempler hvor en bruker alltid må godkjenne og taste inn pairing kode når Bluetooth skal benyttes på mobilterminalen. Dette kan virke svært begrensende på hvor brukervennlig en applikasjon vil være i J2ME.

I forhold til SEAs problemstilling, skal JSR-82 fungere bra mot deres nett. Det er mulig å implementere egne protokoller over SPP støtten vi finner i JSR-82. Støtte for søk etter enheter og tjenester vil gjøre det mulig å kun presentere SEA enheter i et tilpasset UI. Problemer kombinert med lang søketid etter enheter, bør kunne begrenses ved at en benytter en egen installasjonsfase hvor adressen til SEA enheter lagres i mobiltelefonen. Dermed kreves kun søk kun ved installasjon av nye enheter. Ved tilkobling mot en SEA enhet vil en dermed oppnå rask respons på om en tilkobling er vellykket. Dette er et krav dersom en skal kunne tilby en løsning hvor applikasjonen automatisk foreslår å benytte et annet kommunikasjonsmedie, dersom en Bluetooth tilkobling ikke er mulig.

7.5 Mobilnett og fjernstyring

Grunnet utfordringer med adressering og sikkerhet når en benytter andre nettverk for å nå hjemmenettet, valgte vi å se på flere arkitekturer for fjernstyringsløsninger. Vi endte til slutt opp med tre hovedarkitekturer.

Grunnen til at en krever flere løsninger, er som forklart i Kapittel 7.2, der vi foreslår SMS kommunikasjon i mobilnettet kontra en større IP-nettverksstruktur med GPRS- eller GSM-data. Grunnet adresseringsproblemer, som gjør det vanskelig å benytte peer-to-peer kommunikasjon ved bruk av IP-nettverk og mobilkommunikasjon, så vi på en løsning med lokasjonsserver, og en annen løsning med en mer avansert proxy-gateway som all trafikken rutes gjennom. Sistnevnte løser ytterligere problemer med NAT rutere og IP-kommunikasjon via mobilnettet, som hindrer at kommunikasjon kan initialiseres mot mobilterminaler.

Det er stor forskjell i størrelse og kompleksitet ved implementasjon av de ulike løsningene. En SMS løsning vil kun sende SMS mellom en hjemme-GW og en mobiltelefon, mens en GPRS løsning vil kreve en svært stabil server til adresseoppslag eller videresending av meldinger. Implementasjonen av en lokasjonsserverløsning vil være noe enklere enn en proxy-gateway. Sistnevnte kan til motsetning utvides gradvis med funksjonalitet, og trenger derfor ikke å være verre å implementere dersom en kun ønsker samme funksjonalitet som en lokasjonsserverløsning. Funksjonalitetsmessig har derimot en proxy-gateway løsning potensial til å gi langt bedre muligheter for å styre hjemmenettet på en rekke enheter og ulike formater. Å tilby noe lignende i en lokasjonsserverløsning, der en GPRS-modul står for presentasjon av all data, vil bli dyrt i bruk og vanskelig å implementere.

I en SMS løsning har vi svært god skalerbarhet. Ettersom en utelukkende benytter mobilnettet slik det er i dag, vil det ikke være problemer med at antall brukere som benytter tjenesten øker. I en IP-nettverksløsning kan derimot kapasiteten inn mot en sentral server bli en faktor. Som beskrevet i Kapittel 5.3.2.1 gjelder dette spesielt en proxy-gateway som skal videresende all trafikk. Vi mener likevel at det ikke burde være en for tung prosess med dagens til dels kraftige Internett- og serverløsninger. Det burde også være enkelt å utvide med mer og bedre hardware, etter hvert som antallet brukere øker.

En av de større forskjellene for brukeren vil være differansen i latens for de ulike løsningene. SMS har lang sendingstid per melding og mulighet for flere timer forsinkelse mens en GPRS løsning vil basere seg på forbindelsesorientert kommunikasjon, og maks gi noen sekunders forsinkelse. Dersom en ikke kan holde en konstant TCP/IP tilkobling oppe, er en likevel avhengig av SMS for å starte kommunikasjonen, og vi får samme forsinkelsen som ved en SMS løsning ved oppstart av kommunikasjonen. For bruker vil uansett en GPRS løsning kunne ligne langt mer på lokalkommunikasjon, med mulighet for et interaktivt UI som oppdateres fortløpende etter hvert som brukeren navigerer. En tilsvarende SMS basert løsning vil kreve at brukeren aktivt henter informasjon om en og en enhet i hjemmenettet. Det vil da være vanskelig å tilby et identisk UI uavhengig om brukeren befinner seg i nærheten av hjemmenettet eller ikke. Spørsmålet om hvor viktig dette er for brukeren, ligger utenfor denne oppgaven, men formålet med en fjernstyringsløsning vil ofte være å slippe en svært tidskrevende manuell prosess. Forsinkelse ved fjernstyring mener vi derfor kanskje ikke er så farlig for brukeren. I sammenheng med alarmsystemer kan det derimot være et viktig moment.

Meldingsformatet ved smarthusløsninger kan gjøres svært komprimert, og dersom det i tillegg benyttes en binær protokoll over TCP/IP vil vi få svært små meldinger. Ettersom prissystemet i GPRS i dag er volumbasert, er det derfor forventet at løsninger basert på GPRS kommunikasjon vil koste mindre i bruk enn tilsvarende SMS løsninger. Vi mener dette gjelder selv om det gjerne vil være oftere oppdateringer av informasjon i en GPRS løsning.

Spesielt for styringssystemer der alarm er en del av funksjonaliteten, er både autentisering og kryptering viktig. I en SMS løsning vil en kunne benytte den sikkerheten som allerede er implementert i mobilnettet, mens for løsninger hvor en kobler til et IP-nettverk vil en være avhengig av å implementere autentisering og kryptering på høyere lag. Vi har forklart i hvilken sammenheng dette er nødvendig, og mener det finnes gode teknologier, som også er implementert for J2ME, som tilbyr den nødvendige sikkerheten.

Vi så også på implementasjonsmulighetene for alle løsningene i J2ME. Det finnes i dag spesifiserte API'er for både SMS-, TCP-, UDP-, HTTP- og Web Services-kommunikasjon. Med unntak av HTTP og Web Services er dette API'er som et begrenset antall mobiltelefoner støtter i dag ettersom de er valgfrie å implementere i J2ME. Vi har derimot ikke gjort noen formell undersøkelse på hvor mange mobiltelefoner som støtter disse API'ene i dag. Ut i fra de erfaringene vi har etter å ha testet ulike mobiltelefoner, mener vi likevel at mange mobiltelefonprodusenter begynner å legge til støtte for disse API'ene. Med TCP og UDP støtte på mobiltelefonen integreres den også bedre i dagens IP-nettverk. Slik vi ser det er dette svært ønsket i applikasjonsutvikling, ettersom WAP ofte kompliserer implementasjonen av nettverksløsninger. Også WMA API'et, som har eksistert lenge, begynner å bli vanlig å støtte. Ettersom MIDP 2.0 også spesifiserer støtte for HTTPS og SSL, burde det være mulig å enkelt kunne tilby sikker kommunikasjon i GPRS løsninger også. Gjennom MIDP-Push arkitekturen åpnes det i tillegg for at løsninger som krever å kunne motta nettverksforespørsler til enhver tid, ikke krever at en MIDlet kjører aktivt på mobiltelefonen.

Tabell 7.1: Sammenligning av kommunikasjonsteknologier benyttet på mobilterminalen.

Teknologi / Egenskap	SMS	TCP/IP	HTTP/WAP	Web Services
Foreslått Arkitektur	Ren SMS basert kommunikasjon	Lokasjonsserver Proxy-Gateway	Proxy-Gateway	Proxy Gateway
Overhead	Ikke relevant	Lite	Middels	Mye
Push/Pull arkitektur	Push/Pull	Push/Pull	Pull	Pull
Implementasjon	Enkel	Middels	Middels	Enkel
Sikkerhets-implementasjon	Enkel	Middels	Middels	Middels
Java2ME Støtte	Middels	Begrenset	Svært God	Ikke relevant

For SEA kan vi sammenligne løsningene og benyttet kommunikasjonsteknologi som Tabell 7.1 viser. Ved å benytte TCP/IP fra mobilterminalen vil vi kunne tilby et dynamisk UI, og bedre muligheter for alarmfunksjonalitet når kommunikasjonen først er satt opp. En løsning som benytter HTTP/Web Services trafikk fra mobilterminalen vil derimot kunne benyttes på alle J2ME mobilterminaler, men tilbyr begrensede muligheter for alarmfunksjonalitet, ettersom en aktivt må poll'e for å hente informasjon, eller implementere SMS funksjonalitet. En løsning utelukkende basert på SMS kommunikasjon tilbyr minst funksjonalitet, men er klart den enkleste å implementere. Ved å benytte en arkitektur med en proxy-gateway, som står som et mellomledd mellom hjemmenettet og mobiltelefonen, kan en få presentert informasjonen fra hjemmenettet på alle formater i Tabell 7.1.

7.6 Demonstrator

Vi hadde få muligheter for å teste en komplett løsning av de arkitekturer vi har foreslått i oppgaven. Derfor valgte vi å fokusere på å demonstrere at J2ME kan benyttes til å tilby et standardisert UI, uavhengig om en benytter mobiltelefonen som fjernstyringsenhet lokalt i Bluetooth nettet eller via andre nettverk.

Vi opplevde at brukergrensesnittet tilpasses mobiltelefonens UI og at dette fungerer bra på ulike mobiltelefoner. Så lenge en holder seg til standard UI elementer som standardisert J2ME, bør en bruker lett kunne sette seg inn i programmet, da det følger det grensesnittet han eller hun kjenner til. Derimot fikk vi varierende resultater vedrørende plattformuavhengigheten for de ulike J2ME implementasjonene. Enkelte forskjeller og feil i implementasjonene, gjør at applikasjoner ofte ikke vil fungere likt. Dette har trolig sammenheng med at MIDP 2.0 spesifikasjonen er svært ny, og at de implementasjonene vi benyttet er førstegenerasjonsutgave fra mobilprodusentene. Vi mener dette betyr at en som et minimum må teste at en applikasjon fungerer på en gitt mobiltelefon, før en kan si at applikasjonen støtter mobiltelefonen.

Demonstratoren viste også at det er mulig å tilby det samme UI uavhengig om en benytter GPRS eller Bluetooth som kommunikasjonsmedie. Forsinkelsen var merkbart lengre når en benytter GPRS, men vi mener likevel at forsinkelsen er akseptabel i en fjernstyringsløsning, og at samme UI kan benyttes for lokal- og fjernstyring.

OTA funksjonaliteten fungerte bra på alle mobilterminalene. Teknologien gir en enkel og brukervennlig måte for nedlasting av programvaren til en mobiltelefon. Hvordan dette fungerer i forhold til fakturering ved nedlasting av programvare, var noe vi ikke evaluerte i denne oppgaven.

7.7 Videre arbeid

Vi har i dette paperet hovedsakelig fokusert på J2ME implementasjoner på mobilterminaler med spesifikke OS. Vi tror dette definitivt vil være det som vil bli benyttet på de fleste mobilterminaler i fremtiden. I dag består derimot markedet av en rekke mobiltelefoner som kjører proprietær programvare med en egen implementasjon av J2ME. Et studie i hvor store forskjeller det er i implementasjonene av J2ME og de ulike API'er ville være aktuelt. Dette vil kunne gi et bedre bilde av hvor plattformuavhengige J2ME applikasjoner faktisk er.

Av samme grunn er det svært vanskelig å få en komplett oversikt over hvilke J2ME implementasjoner som støtter de valgfrie API'ene, som Bluetooth og TCP/IP. Vår erfaring har vært at dette varierer stort mellom ulike mobiltelefonleverandørene, og også mellom forskjellige mobiltelefoner. I forhold til å benytte disse valgfrie API'ene i en løsning, burde det foretas bedre undersøkelser på hvor stor del av mobiltelefonmarkedet som støtter de ulike API'ene.

Innen OTA er det en utfordring hvordan fakturering ved nedlasting av programvare skal utføres. Enkelte av plattformene har som kommentert i dette paperet komplette løsninger for dette. Likevel er det et felt hvor en har sett en rekke lite brukervennlige løsninger. På tross av dette blir det mer og mer vanlig å benytte mobiltelefonen i sammenheng med betalingssystemer, og det kunne derfor være interessant å kartlegge alternative løsninger på dette området.

Videre arbeid på de løsningene vi har presentert, ligger hovedsakelig i å kartlegge implementasjonskrav for hjemme-GW og lokasjonsserver/proxy-gateway. Spesielt innen sikkerhet må en vurdere i hvilken grad autentisering og kryptering kreves mot en sentral server. I tillegg burde det undersøkes hvilke serverteknologier, som Servlets, PHP, ASP og databasesystemer, som er best egnet for disse løsningene. Kombinert med dette, bør det foretas undersøkelser på hvor stor belasting en sentral server kan forventes å få i en slik løsning, og hvor sårbar den er i forhold til en peer-to-peer løsning ved feilsituasjoner. For bedriften vil det også være viktig å kartlegge eventuelle GPRS-moduler som kan benyttes som gateway mot hjemmenettet. Vi har i paperet kommentert at en kan benytte en vanlig mobilterminal til dette. Men det finnes en rekke moduler på markedet som er billigere og som kan være lettere å integrere med annen hardware.

Vi har i dette paperet hovedsakelig sett på dagens teknologi og muligheter. Såkalte 3G telefoner og mobilnett vil være langt mer fokusert på integrasjon og støtte for IP-nettverk. Kombinert med introduksjon av IPv6 ville det derfor være aktuelt å undersøke hvilke muligheter som gis ved fremtidige mobilnett for fjernstyring av smarthusløsninger, og om teknologiene vil løse problemer vi har påpekt i dagens systemer.

8 Konklusjon

I denne oppgaven har applikasjonsutviklingsmulighetene for mobilterminaler vært gjenstand for våre undersøkelser. Med fokus på kommunikasjonsteknologi og brukergrensesnitt, skulle vi foreslå løsninger for lokal- og fjernstyring av SEAs Bluetooth nettverk fra mobilterminaler. I tillegg var målsettingen å finne den mest passende plattformen for fremtidig utvikling av en applikasjon for disse løsningene.

Ved valg av kommunikasjonsteknologi og applikasjonsplattform, fokuserte vi på at en løsning måtte være plattformuavhengig, så lenge den gir støtte for de kommunikasjonsteknologiene en ønsker å benytte. Vi mener at Java fra Sun Microsystems, som tilbyr en plattformuavhengig løsning gjennom sin J2ME plattform, kan anbefales. J2ME tilbyr desidert best støtte blant dagens mobilterminaler, og har bred støtte blant produsenter av både mobiltelefoner og andre applikasjonsplattformer. Gjennom en mengde standardiserte, valgfrie API'er, er J2MEs ressursutnyttelse i dag på høyde med lavnivå applikasjonsplattformer.

Brukergrensesnittet i J2ME-applikasjoner tilpasses automatisk standardbrukergrensesnittet på mobiltelefonen applikasjonene kjøres på. Dersom utvikler kun benytter J2MEs standardkomponenter, og holder seg til de retningslinjer som vi har beskrevet i oppgaven, mener vi at applikasjonen vil være brukervennlig uavhengig av hvilken mobiltelefon den kjøres på. Demonstratoren har vist at dette stemmer bra i forhold til brukergrensesnitt, men grunnet feil på mange mobiltelefoner i J2ME implementasjonene av MIDP 2.0 API'er, må en applikasjon alltid testes før en kan si at den fungerer på en gitt mobiltelefon. Suns idé om at kode skrevet en gang, alltid vil fungere uten endringer på forskjellige mobiltelefoner, mener vi derfor sjelden stemmer for J2ME.

Bluetooth vil etter vår mening være det beste alternativet for kommunikasjon over kort rekkevidde. Dette valget er basert på SEAs krav. Vi tror også at en sameksistens mellom Bluetooth og WLAN på mobiltelefoner i fremtiden er et mer trolig scenario enn at WLAN vil fortrenge Bluetooth. Vi mener også, basert på drøfting og demonstrator, at det er riktig å velge en Java-løsning som støtter JSR-82. Derved oppnår en også plattformuavhengig Bluetooth støtte. Dette er oppnåelig så lenge mobilleverandørene følger spesifikasjonene i J2ME. En løsning basert på JSR-82 vil ha alt som skal til for å integrere mobiltelefoner med SEAs Bluetooth nett.

En løsning for styring av SEAs Bluetooth nett over mobilnettet, vil baseres på en GW i hjemmenettet, som er basert på mobilkommunikasjon, og en mobilterminal som kommuniserer mot denne. Flere av hindringene som oppstår idet det benyttes GPRS eller WAP, og det skal gå trafikk mellom to mobilterminaler, kan løses eller forenkles ved å benytte en lokasjonsserver eller proxy-gateway.

Vi mener det beste for SEA vil være å benytte en SMS løsning, dersom det er ønskelig å få opp en rask, enkel og funksjonelt tilstrekkelig løsning. Dersom det derimot er ønskelig med en mer fullstendig løsning, som vil tilby styring fra både PC- og WAP-nettlesere, vil vi anbefale en løsning basert på GPRS og en proxy-gateway, som tilkobles Internett eller et eget APN. Gjennom nye API'er i J2ME MIDP 2.0, vil en kunne støtte både alarmfunksjonalitet og sikkerhet i en GPRS-basert løsning.

Vi mener denne oppgaven har levert et godt overblikk i et uoversiktlig marked for applikasjonsutvikling til mobilterminaler. For SEA vil det i dag være fullt mulig å få til en kombinert Bluetooth, SMS og GPRS løsning, hvor det via J2ME tilbys et felles intuitivt brukergrensesnitt, selv om en GPRS løsning vil kreve at det utvikles en stabil proxy-gateway. For fremtiden gjenstår det å se hvordan 3G mobilnett, og større fokus på IP-telefoni, vil kunne forenkle og forbedre mulighetene for kommunikasjon mellom IP-baserte mobilterminaler.

Referanser

Ref. Nr.	Referanse
[1]	Roger Riggs, Antero Taivalsaari & Mark VandenBrink, *Programming Wireless Devices with the Java™ 2 Platform, Micro Edition*, Addison-Wesley, 2001, ISBN 0-201-74627-1
[2]	John W. Muchow, *Core J2ME™ Technology & MIDP*, Sun Microsystems Press, 2002, ISBN 0-13-066911-3
[3]	Kumar, Kline & Thompson, *Bluetooth Application Programming with the Java APIs*, Morgan Kaufmann Publishers, 2004, ISBN 1-55860-934-2
[4]	Bruce Hopkins & Ranjith Antony, *Bluetooth for Java*, Apress, 2003, ISBN 1-59059-078-3
[5]	Michael Juntao Yuan, *Enterprise J2ME. Developing Mobile Java Applications*, Pearson Education, 2004, ISBN 0-13-140530-6
[6]	Little Springs Design, *User Interface Design Guidelines for J2ME MIDP 2.0*, Little Springs Design, 2003
[7]	Frode Sørensen, *UMTS - mer og mer IP*, Kompendium i faget IKT2315, Høgskolen i Agder, september 2003
[8]	3GPP, *Security related network functions; 3rd Generation Partnership Project; Digital cellular telecommunications system (Phase 2+) (3GPP TS 03.20 V9.0.0 Release 2000)*
[9]	Wireless Application Protocol Forum, *Wireless Application Protocol Architecture Specification (WAP-210-WAPArch-20010712-a)*
[10]	Bluetooth SIG, *Bluetooth specification Version 1.2 [vol 2] (november 2003)*
[11]	The Parlay Group, *Parlay 4 - Parlay X Web Services Specification (Version 1.0)*
[12]	KK Tan, CY Soh & KN Wang, *Development of an Internet Home Control System (0-7803-7386-3/02 IEEE 2002)*, Department of Electrical and Computer Engineering, National University of Singapore, 2002
[13]	Hiroshi Kanma & Noboru Wakabayashi, Ritsuko Kanazawa & Hiromichi Ito, *Home Appliance Control System over Bluetooth with a Cellular Phone (0-7803-7721-4/03 IEEE 2003)*, Digital Media Systems R&D Division, Hitachi, 2003
[14]	Tomasz Keller, Rajmund Paczkowski & Józef Modelski, *Using Bluetooth in a System for Integrated Control of Home Digital Network Devices*, Institute of Radioelectronics, Warsaw University of Technology
[15]	Microsoft, *Microsoft Pocket PC 2003 SDK (Version 1.0)*
[16]	Microsoft, *Microsoft Smartphone 2003 SDK (Version 1.0)*
[17]	PalmSource, *Palm OS SDK: Version 5*

Ref. Nr.	Referanse	Aksessert
[18]	Geir Stian Bjåen & Erling Kaasin, *Security in GPRS, Master Thesis in Information and Communication Technology*, Høgskolen i Agder, mai 2001 http://www.siving.hia.no/ikt01/ikt6400/ekaasin/Master Thesis Web.htm	24.05.2004

[19] Charlotta Bååth & Joanna Kühn, *SMS over GPRS, A Comparison Between GSM and GPRS Architectures as Carriers for SMS and Between SMS and Other Protocols as Carriers of Short Messages over GPRS*, Royal Institute of Technology, april 2003 — 24.05.2004
http://www.d.kth.se/~d99-lba/doc/report-sms-gprs-cbaath-jkuhn.pdf

[20] Jyrki Oraskari, *Bluetooth versus WLAN IEEE 802.11x*, Helsinki University of Technology, oktober 2000 — 20.04.2004
http://www.hut.fi/~joraskur/BT2.pdf

[21] Nancy Cam-Winget, Tim Moore, Dorothy Stanley & Jesse Walker, *IEEE 802.11i Overview*, National Institute of Standards and Technology, desember 2002 — 20.04.2004
http://csrc.nist.gov/wireless/S10_802.11i Overview-jw1.pdf

[22] Thomas G. Xydis & Simon Blake-Wilson, *Security Comparison: Bluetooth™ Communications vs. 802.11*, Bluetooth Security Experts Group, 2002 — 29.05.2004
http://www.ccss.isi.edu/papers/xydis_bluetooth.pdf

[23] MontaVista Software, *Montavista Linux Consumer Electronics Edition 3.1 (rev 1.10)*, februar 2004 — 20.04.2004
http://www.mvista.com/dswp/cee_ds.pdf

[24] MontaVista Software, *Digital Airways* — 20.04.2004
http://www.mvista.com/partners/isv/digital-airways.html

[25] SavaJe Technologies, *SavaJe OS: Platform Overview - J2ME Personal Basis Profile & MIDP 2.0 Edition*, september 2003 — 20.04.2004
http://www.savaje.com/images/downloads/PlatformOverview-Final-September2003.pdf

[26] NetScreen Technologies, *GPRS Security Threats and Solutions*, mars 2002 — 26.05.2004
http://www.firewall-reviews.com/documents/ACFKM4dU8.pdf

[27] AppForge, *Crossfire, Simply the fastest way to develop mobile and wireless applications*, 2003 — 20.04.2004
http://www.appforge.com/products/enterprise/crossfire/crossfirespecsheet.pdf

[28] GSM World, *The website of the GSM Association* — 20.04.2004
http://www.gsmworld.com

[29] GSM World, *SMS (Short Message Service)* — 20.04.2004
http://www.gsmworld.com/technology/sms

[30] Netsize, *European SMS Guide*, februar 2003 — 20.04.2004
http://www.gsmworld.com/technology/sms/presentations/euro_sms_guide.pdf

[31] Ericsson, *WAP architecture overview*, november 2003 — 20.04.2004
http://www.ericsson.com/mobilityworld/sub/open/technologies/wap/about/wap_architecture_overview

[32] Symbian, *Symbian OS phones* — 21.04.2004
http://www.symbian.com/phones

[33] Qualcomm, *Brew Home* — 21.04.2004
http://brew.qualcomm.com/brew

[34] Qualcomm, *Brew Devices by Manufacturer* — 21.04.2004
http://www.qualcomm.com/brew/developer/resources/ad/devices_man.html

[35] BenHui.Net, *MIDP 2.0 Phones and PDAs*, april 2004 — 22.04.2004
http://benhui.net/midp2phonelist.html

[36] Atmel Corporation, *ZigBee - Wireless Standard Comparisons*, 2004 — 24.04.2004
http://www.atmel.com/products/zigbee

[37] Rococo Software, *Welcome to Rococo* — 06.02.2004
http://www.rococosoft.com

[38] Possio, *Welcome to Possio* — 27.04.2004
http://www.possio.com

[39] Pedro Amaro, *The Clash of Mobile Platforms: J2ME, ExEn, Mophun and WGE (Kapittel 5: Mophun)*, juni 2003 — 23.05.2004
http://www.midlet-review.com/index?content=articles/article_platformclash02

[40] Synergenix Interactive, *Welcome to Synergenix Interactive* — 23.05.2004
http://www.synergenix.se

[41] kSOAP, *Open Source SOAP for the kVM* — 29.05.2004
http://ksoap.enhydra.org

[42] Forum Nokia, *Multi-Player MIDP Game Programming (Version 1.0)*, oktober 2003 — 29.05.2004
http://nds1.forum.nokia.com/nnds/ForumDownloadServlet?id=3838

[43] Telenor Mobil, *CPA WAP - internettjenester på mobilen* — 29.05.2004
http://telenormobil.no/partner/tjenester/cpawap

[44] Kefk Network, *Bluetooth*, november 2003 — 29.05.2004
http://www.kefk.net/Hardware/Wireless/Bluetooth

[45] Sun Microsystems, *Java™ 2 Platform, Micro Edition*, 2002 — 30.05.2004
http://java.sun.com/j2me/docs/j2me-ds.pdf