Datavetenskap

Jonas Nilsson

Christian Däldborg

Box On-Line

Examensarbete, C-nivå

2002:06

Box On-Line

Jonas Nilsson

Christian Däldborg

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Jonas Nilsson

Christian Däldborg

Godkänd, 2002-06-04

Handledare: Robin Staxhammar

Examinator: Stefan Alfredsson

Sammanfattning

Examensarbetet är genomfört under våren 2002 på Telia Mobile AB i Karlstad. Uppgiften bestod av, en analytisk del och en konstruktionsdel, där vi skulle implementera en enkel demonstrationsprodukt av konceptet ”Box On-Line”, BOL.

Konceptet ”Box On-Line” innebär att man åstadkommer mobil trådlös Internetaccess för olika typer av utrustning. Detta görs genom att man ansluter utrustningen till sin BOL-enhet. Denna är tänkt att vara en liten ”svart låda” som är konstant uppkopplad via ett mobilnät.

Med detta koncept skall det på ett enkelt sätt gå att exempelvis skapa mobila väderstationer eller andra mobila tillämpningar. Produkten skall vara enkel att använda och billig att köpa.

I den analytiska delen undersökte vi vilka problem och möjligheter som finns med konceptet. Vi presenterar först en idealistisk vision av BOL-konceptet, och utifrån de problem som finns med denna har vi gjort en rad begränsningar för att nå en realistisk version.

Vi har undersökt fyra olika lösningar. Av dessa fyra lösningar anser vi att den lösning som kallas ”Certifierade drivrutiner”, beskriven i avsnitt 3.4, är den bästa. Denna lösning implementerade vi i vår demonstrationsprodukt. Genom utvärdering och tester av denna lösning har vi dragit slutsatsen att det är en bra lösning och att den går att utveckla vidare till en konsumentprodukt.

Box On-Line

Abstract

This bachelor project has been accomplished during Spring 2002 at Telia Mobile AB in Karlstad. The task consisted of two parts, one analytic part and one construction part, where we were to implement a simple prototype of "Box On-Line", BOL.

The concept "Box On-Line" means that you achieve a mobile wireless Internet access for different kinds of equipment. This is done by connecting the equipment to your BOL unit. This unit is intended to be a little "black box" which is constantly online through a mobile network.

With this concept it will be easy to create mobile weather stations for example, or other mobile applications. The product must be easy to use and cheap to buy.

In the analytic part we investigated what problems and possibilities there are with the concept. First we will introduce an idealistic vision of the BOL concept, and from the problems that occurred with this one, we have made several limitations to reach a realistic version.

We have examined four different solutions. Out of these four solutions we find the solution in chapter 3.4, called "Certified Drivers", the best. This solution we implemented in our prototype. Through evaluation and tests of this product we have concluded that it is a good solution capable of further development to a product for consumers.

Förord

Vi vill här först och främst tacka Telia Mobile Internet lab som gett oss möjlighet till examensarbete hos dem. Dessutom vill vi tacka vår handledare på Telia, Jan H Gustavsson, för all hjälp han bidragit med. Ett tack till Håkan Blomkvist för vägledning och idéer initialt. Dessutom ett stort tack till all övrig personal på Telia för er vilja att hjälpa till samt ert vänliga bemötande.

Ett tack till vår handledare på Karlstads universitet, Robin Staxhammar, för att han upprepade gånger läst vår uppsats och kommit med synpunkter och förslag.

Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Inledning

1.1 Bakgrund

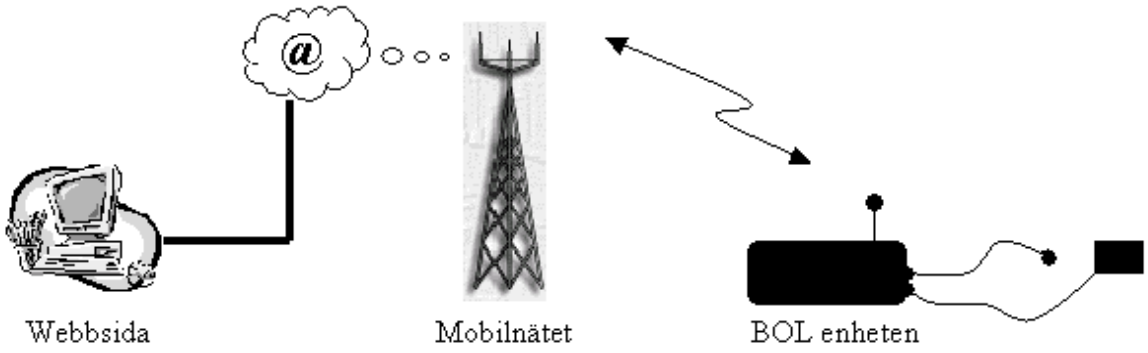
Datakommunikation över mobila trådlösa nät har inte nått de volymer som förutspåtts. En av anledningarna kan vara att användarna anser det krångligt att använda. En annan anledning kan vara bristen på tillämpningar. Att det finns så få tillämpningar beror i stor utsträckning på den relativt låga bandbredden. De mobiltelefoner som idag klarar av att hantera datatrafik har en bandbredd som ligger långt under vanlig modemhastighet. Det gör det svårt att hitta riktigt bra och intressanta användningsområden. Nya tekniker som gör att hastigheten på överföringen närmar sig modem, exempelvis UMTS (Universal Mobile Telefon System) även kallat 3G, är på ingång.

För att nå stora volymer med en ny teknik gäller att den antingen är enkel att använda och/eller att den är revolutionerande på något sätt. Mobil access till Internet har inte levt upp till något av detta. Det ligger i Telia Mobiles intresse att trafiken i mobilnäten ökar. För att åstadkomma det krävs nya intressanta tillämpningar för konsumenterna. Ett koncept på en tillämpning som därför skapats är ”Box On-Line”.

Konceptet ”Box On-Line”, BOL, innebär mer detaljerat att man med hjälp av en BOL får mobil Internetaccess för olika typer av utrustning. Tanken är att användaren köper en ”svart låda”, en BOL. BOLen är konstant uppkopplad mot en dator eller server via GPRS, General Packet Radio Service. Till denna kan användaren koppla in olika typer av utrustning som exempelvis en termometer, vindmätare, kamera eller en chatboard. Visionen är att det av användaren inte skall krävas någon, eller mycket enkel, konfiguration. Data kan sedan samlas in, skickas vidare för behandling, och exempelvis presenteras på en webbsida.

En användare skall kunna gå till butiken och köpa en BOL. Det skall även finnas olika tillbehör som han kan köpa till. Användaren skall också kunna använda sig av eventuella enheter som han kan tänkas ha tillgängliga redan. Användaren skall bara behöva ansluta sina enheter till BOLen. Genom att användaren sedan trycker på startknappen konfigureras BOLen, kopplar upp sig och börjar samla in data. BOLen skall också klara det omvända förhållandet att man kan ansluta utrustning för presentation av data eller styra utrustning. Det skall alltså gå att både samla in data och skicka ut data. Med detta menas att exempelvis en termometer samlar in data som av BOLen vidareförmedlas medan utdata är om man skickar

ut data till en enhet, exempelvis för att styra ett magnetiskt dörrstopp. Allt skall ske med automatik eller mycket enkel konfiguration.



Figur 1: Hur konceptet ”Box On-Line” fungerar

I dagsläget finns det på marknaden ingen färdig produkt att köpa. Mot denna bakgrund har Telia Mobile i Karlstad gett oss i uppdrag att utreda konceptet ”Box On-Line”. Arbetet är utfört under vårterminen 2002.

1.2 Mål

Målet med examensarbetet är att utreda huruvida konceptet ”Box On-Line” skulle kunna fungera i verkligheten. I arbetet ingår att undersöka vilka stora problem det finns och titta på möjliga lösningar.

Genom att studera olika övergripande lösningar skall vi presentera en lösning som vi tycker är bra. Dessutom skall vi enligt denna lösning implementera en begränsad demonstrationsprodukt. Detta för att se om vår lösningsmodell håller i realiteten.

Vi skall, då vi studerar olika lösningar, ta hänsyn till hur hela ”kedjan” kan tänkas fungera, från det att en enhet ansluts tills det att en drivrutin är installerad och körs på BOLen samt att kommunikationslänken är uppkopplad och fungerar.

1.3 Uppsatsens upplägg

I nästa kapitel beskriver vi hur en idealistisk vision av hur ”Box On-Line” skulle fungera. Därefter analyserar vi denna och delar upp de problem som finns i tre problemområden. Vissa av dessa problem ser vi som omöjliga att lösa, andra är lösbara. Detta ger oss en omformulering av den ideala visionen till en realistisk version av ”Box On-Line”.

I målet med examensarbetet ingår att undersöka flera olika alternativa till lösningar för ”Box On-Line”. De fyra lösningar som vi undersökte närmare finns beskrivna i kapitel 3. För varje lösning undersökte vi dess fördelar och nackdelar. Kapitlet avslutas med en sammanfattning av de olika lösningarna och hur väl de stämmer överens med den realistiska versionen.

För att läsaren skall förstå problemen och fördelarna med mobil kommunikation ger vi härnäst en översikt av hur mobiltelefonsystemet fungerar. Kapitlet inleds med en överblick av utvecklingen, därefter följer en beskrivning av vad det är som gör GPRS lämpat för BOL. Till sist ges en förklaring till varför hastigheten kan variera över mobiltelefonnätet.

Efter genomgången av mobiltelefonsystemet beskriver vi vår konstruktionslösning. Först ges en övergripande bild av hur vi valt att vår lösning skall fungera. Därefter beskrivs de avgränsningar som gjorts vad gäller demonstrationsprodukten.

Efter denna mer översiktliga bild av demonstrationsprodukten beskrivs vår demonstrationsprodukt implementationsspecifikt. Vi visar vår demomiljö, i vilken demonstrationsprodukten är utvecklad. Vi visar programflöden och protokollspecifikationer översiktligt. Vi beskriver det användargränssnitt som vi valt, webbsidan, och hur man genom denna kommunicerar med servern.

För att verifiera att demonstrationsprodukten fungerar tillfredställande har vi utfört en rad olika tester. Dessa är nämnda i kapitel 7 där vi också utvärderar utfallet av dessa. Uppsatsen avslutats med resultat och rekommendationer av arbetet, vilka erfarenheter och resultat som vi har fått av uppgiften, vad vi har lärt oss och vad som kunde ha gjorts annorlunda.

2 Problemställning

I detta kapitel beskriver vi den ideala visionen av konceptet ”Box On-Line”. Vi analyserar den och ser vilka problem som vi kan identifiera. Efter att ha analyserat problemen gör vi en omformulering av konceptet till en realistisk version.

2.1 Idealistisk vision

Avsnittet beskriver den ideala visionen av konceptet ”Box On-Line”. Den beskrivs på det sätt som Telia skulle vilja att produkten fungerar, om det inte finns några tekniska begränsningar.

Den ideala lösningen på konceptet ”Box On-Line” är att användaren inte behöver göra någon som helst konfigurering. När en användare startar en BOL-enhet kopplar den automatiskt upp sig mot en server. Om någon ny utrustning ansluts till BOLen så identifieras enheten och rätt drivrutin laddas till BOLen från en central server. Användaren behöver inte ange vilken utrustning denne har anslutit. När drivrutinen är installerad börjar BOLen kommunicera med en server, t.ex. en webbserver som användaren sedan kan kommunicera med.

För att en BOL skall bli en lyckad konsumentprodukt är det viktigt att den känns enkel att använda. Om man vill uppnå stora volymer är det också viktigt att den blir relativt billig att köpa. Visionen är också att befintliga ”hyllprodukter” skall kunna anslutas till BOL. Det vill säga att BOLen måste ha ett gränssnitt som är kompatibelt med befintliga produkter, exempelvis kan COM eller USB användas. Redan befintliga drivrutiner till dessa produkter skall kunna användas [3], [4]. Ett alternativ för att minska behovet av konfiguration är att använda sig av specialtillverkad kringutrustning istället för att använda befintliga hyllprodukter. Dessa skulle då kunna ha ett gemensamt gränssnitt för kommunikation med BOLen. Dessa enheter skulle dock sannolikt bli mycket dyrare.

2.2 Analys av den idealistiska visionen

I detta examensarbete har vi lagt fokus på BOL-enheten och hur den skall fungera. Hur skall konfigureringen skötas? Hur skall man sköta dataöverföringen? Hur skall gränssnittet mot

utrustningen se ut? Vid en analys av den ideala visionen framträder vissa områden där det finns stora problem. Vi har delat upp problemen i tre områden.

Hur skall gränssnittet mot utrustningen på BOLen se ut och hur skall man sköta konfigureringen på BOLen. Detta har vi kallat problemområde ”anslutning av enheter”, se avsnitt 2.2.1.

Vi ser också ett stort problem med överföringen mellan server och BOL. Hur skall kommunikationen mellan dessa se ut och hur löser man de problem som finns inom detta område? Detta har vi undersökt i avsnitt 2.2.2, ”överföring till server”.

I servern uppstår vissa problem som är gemensamma med BOL-enheten. Dessa rör konfiguration och den automatiska identifikationen av enheter. Vi har tittat närmare på problemen på serversidan i avsnitt 2.2.3, ”serversidan”.

Vi har antagit att servern är tillräckligt ”intelligent” för att kunna presentera information för användaren när vi har analyserat de tre problemområdena nedan.

2.2.1 Anslutning av enheter

Problemet här är vilket gränssnitt som skall användas mot utrustningen. Skall COM eller USB användas, eller kanske båda?

Hur skall man kunna identifiera enheterna automatiskt? Om man använder USB går detta med hjälp av den automatiska identifiering som finns inbyggt i USB, men hur skulle man kunna göra det om man använder COM? USB är i detta avseende bättre men det finns i idag mycket utrustning som använder COM-porten på datorn. Det skulle vara bra om användaren kunde använda även denna utrustning. Exempel på sådana enkla enheter, d.v.s. okomplicerade givare och ställare, är termometrar och dörrstopp. De flesta enheterna som är intressanta för BOLen är sådana enkla enheter, dessa finns inte med USB. USB kräver också mer systemresurser i BOLen för att fungera än vad COM gör. Med tanke på att BOLen skall vara så enkel och billig som möjligt är det bra om BOLen blir resurssnål.

Hur skall nedladdningen av rätt drivrutiner ske? Om man laddar ner dem från en central server automatiskt måste alla tänkbara drivrutiner ligga på denna. Det finns ytterligare två stora problem med drivrutinerna. Eftersom BOLen inte har någon möjlighet att direkt kommunicera med användaren måste de installeras utan att användaren behöver trycka på ”ok knappar” eller utföra annan handpåläggning. Om man granskar befintliga drivrutiner ser man att nästan alla kräver interaktion med användaren. Det andra stora problemet är att den data som drivrutinen transporterar mellan BOLen och enheten (indata och utdata) måste vara

tillgänglig för BOLen så att den kan förmedla den data som kan vara intressant för användaren till servern. Hur skall BOLen veta var den skall plocka den informationen från?

Förtydligat innebär detta att om användaren vill läsa av information från någon enhet så måste BOL-drivrutinen veta var informationen finns att hämta.

2.2.2 Överföring till server

Det finns i Telias GPRS-nät idag ingen möjlighet för en dator eller en server att etablera en GPRS-förbindelse till en mobiltelefon eller någon annan GPRS-mottagare. Initieringen måste alltid ske av telefonen. Man kan alltså inte nå en telefon som inte först begärt en uppkoppling. Detta eftersom telefonen har ett svart IP-nummer, d.v.s. dess IP-nummer är dynamiskt tilldelat och är därför inte publikt, det vill säga inte känt på Internet. Servern vet alltså inte vilket IP-nummer GPRS-enheten har förrän en uppkoppling har skett.

Man kan därför inte tilldela en BOL ett statiskt IP-nummer och sedan använda detta vid kommunikation via Internet. Problemet av ovanstående blir alltså att servern inte kan ”koppla upp” sig mot en BOL-enhet. Uppkopplingen måste ske från BOLen.

NAT-servern (Network Address Translation), som sköter kopplingen mellan telefonens svarta IP-nummer och dess publika IP-nummer måste få trafik inom ett visst tidsintervall. Annars släpper den adressöversättningen och man tappar kopplingen mellan IP-adresserna. Har man tappat kopplingen måste den öppnas av telefonsidan igen. Då kommer BOLen troligtvis att tilldelas en annan publik IP-adress.

Hur skall överföringen ske mellan klient (BOL) och server? Protokollet skall vara utformat för att passa all tänkbar utrustning. Det måste dessutom innehålla någon form av säkerhet så att bara ägaren av en BOL kan titta på informationen från denna, såvida användaren inte väljer att göra informationen publik. BOLen kanske skall agera övervakningsstation och då är det lämpligt om endast ägaren kan se den information som kommer från enheten. Är BOLen däremot utformad som en väderstation kanske användaren vill att vem som helst skall kunna se informationen. Ett annat problem inom detta område är hastigheten på överföringen via GPRS. Hastigheten är begränsad och kan dessutom variera kraftigt. Detta innebär att man får långa svarstider. Hur fungerar drivrutinerna med detta problem? Generellt sett skickar drivrutinerna ett kontinuerligt flöde av data, men BOLen måste begränsa trafikmängden efter vad som kan överföras. Strömmande media, exempelvis film från en webbkamera, blir således svårt att hantera.

När det gäller radiolänken förekommer det mycket störningar och fel i trafiken, på grund av bland annat radioskugga och överbelastning. Om BOLen är i rörelse kan det också bli fel då man byter från en basstation till en annan. Man kan tappa kontakten med BOL-enheten. Detta förekommer dock väldigt sällan. Om BOLen är i radioskugga mer än sex minuter ”släpper” NAT-servern IP-kopplingen och en ny uppkoppling måste initieras från BOLen.

Det finns felkorrigering i både GPRS-nätet och TCP-protokollet vilket gör att fel på högre nivå inträffar ytterst sällan, men man får dock räkna med långa och varierande svarstider. Det kan i värsta fall bli så långa uppehåll att GPRS-modulen tappar kontakten med nätet. Med det menas att kopplingen mellan BOL-enhetens ”svarta” IP och de publika IP som finns på Internet försvinner. Detta inträffar dock ytterst sällan på grund av de felkorrigeringar nämnda ovan. Den stora variationen av svarstider och hastigheten över GPRS beror främst på vilken belastning som finns på nätet just för tillfället.

2.2.3 Serversidan

Vi gjorde tidigare ett antagande att servern skall kunna presentera informationen för användaren, se avsnitt 2.2. För att servern skall kunna göra detta krävs det att den vet vad det är för data som den tar emot. Det är stor skillnad på om det kommer ett värde från en givare, en bild från en kamera eller en textfil från någon annan enhet. Detta måste servern veta för att kunna använda sig av informationen.

Den måste även klara av att ta emot information som skall skickas till någon av BOLens anslutna enheter på ett smidigt och enkelt sätt.

Serven måste också ha någon form av databas så att den kan hantera många anslutna BOL-enheter samtidigt.

2.2.4 Summering av de olika problemen

En kort summering av de problem som analyserats i avsnitt 2.2.1-2.2.3

- Vilket gränssnitt skall användas på BOLen mot enheterna?
- Hur skall problemet med olika drivrutiner på BOLen hanteras?
- Hur skall BOL-drivrutinen veta var den skall hämta informationen från en enhet? (När en användare har begärt att få data från exempelvis en givare)
- En förbindelse mellan Internet och en GPRS-telefon kan endast initieras från telefonen.
- NAT-servern kräver trafik inom ett visst tidsintervall för att kopplingen mellan telefonen och Internet inte skall försvinna.
- Hastigheten över GPRS är tämligen begränsad om man jämför med modem och LAN.
- Hur skall protokollet mellan BOL och server vara utformat?
- Hur skall servern veta vad för sorts data som BOL-enheten skickar?
- Hur skall informationen presenteras för användaren?

2.3 Realistisk version

Vi ser ingen möjlighet att lösa alla de problem som vi tagit upp i 2.2 så att den ideala visionen bibehålls. Detta innebär att vi har omformulerat den ideala visionen till en realistisk version som vi anser är möjlig att implementera. En stor ändring i visionen gäller den helautomatiska konfigurationen. På grund av de problem som vi beskrivit ovan kommer den sannolikt aldrig att kunna lösas, i alla fall inte med rimliga medel, framförallt inte inom detta examensarbete. Vi ser ingen möjlighet att kunna göra detta på något sätt. Därför måste vi här göra avsteg från den idealiska visionen. Antingen får vi bli mindre generella med vad för typ av kringutrustning som man kan koppla in eller så får en viss grad av konfiguration accepteras. Varje tillbehör/enhet som kopplas in i BOLen måste styras av sina speciella drivrutiner. Antingen lägger man dem i boxen eller i servern. Vilket av alternativen man än väljer så kommer man inte ifrån problemet med att installation och konfigurering av drivrutinen måste göras. Det är viktigt att konfigureringen av BOLen är så pass enkel att även en person med mindre datorvana kan använda den utan problem.

Visionen att man skall kunna koppla in nästan all tänkbar utrustning går att genomföra om man till dessa kräver speciella BOL-drivrutiner för varje enhet. Dock går det inte att lösa den automatiska konfigurationen och igenkänningen av enheter som ansluts, eftersom inte alla tänkbara enheter går att identifiera, om man inte använder sig av USB.

I den realistiska versionen måste man alltså göra vissa begränsningar av den ideala idén när det gäller konfigurationen. Användaren måste på något sätt ställa in eller ladda in drivrutiner. Detta kan antingen ske i BOLen eller i servern. Vilket alternativ man väljer ger olika fördelar och nackdelar. Om man väljer att installera drivrutinerna i BOLen måste det göras av användaren, vilket kan vara komplicerat och krångligt för en datorovan användare. Lägger man däremot installationen av drivrutinerna i servern kan installationen skötas av en anställd eller eventuellt av en tredje part. I nästföljande kapitel kommer vi att presentera olika konstruktionsförslag.

3 Olika konstruktionsförslag

I planeringsfasen undersökte vi ett antal möjliga vägar att gå för att lösa de problem som finns med konceptet Box On-Line, BOL. Nedan går vi igenom fyra modeller som vi undersökte, varav tre förkastades på grund av diverse olika nackdelar, eller att de helt enkelt inte passade in i visionen. Den modellen vi valde att bygga vidare på är beskriven nedan i avsnitt 3.4.

3.1 Certifierad kringutrustning

Det speciella med denna idé är att alla tänkbara enheter skall kommunicera med BOLen på ett och samma sätt oberoende av vad det är för enhet. I och med att alla enheter kommer att fungera på samma sätt mot BOLen blir det inga drivrutiner som skall installeras för användaren. BOLen används enbart som en koppling mot servern. Den har låg intelligens och utför inget annat arbete än att den sköter uppkopplingen och förmedlar data. Intelligensen delas i detta fall mellan enheterna och servern. BOLen blir bara en länk till GPRS/Internet, se figur 2.

Tanken med denna idé är att man lägger mycket av systemets intelligens i enheterna/kringutrustningen. Vi försöker med denna metod eliminera många av de problem som finns vid anslutning av enheter. Därav namnet certifierad kringutrustning.

Kringutrustningen måste vara specialtillverkad och ”BOL-certifierad” för att kunna anslutas till en BOL.

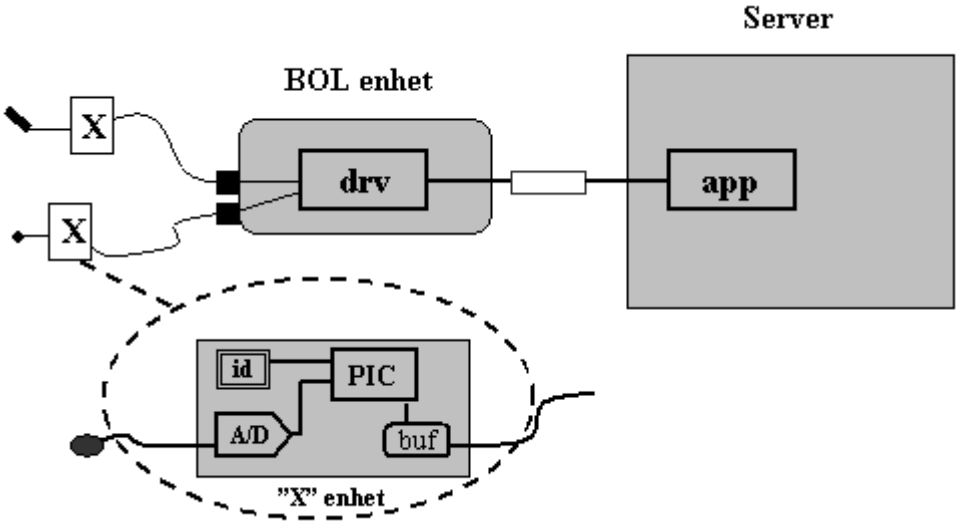
En enhet som är möjlig att ansluta skulle exempelvis kunna bestå av följande: En givare, eller vad det nu är som man vill koppla till, en liten krets med en Programmable Interrupt Controller (PIC) processor och eventuellt en A/D-omvandlare. I vissa fall måste kretsen även vara utrustad med ett buffertminne. I kretsen som finns i varje enhet hårdkodas ett ID-nummer in vid tillverkningen. Detta nummer används vid identifieringen av enheterna, se X i figur 2.

För att alla olika enheter skall kommunicera med BOLen på ett och samma sätt används ett eget protokoll. Protokollet kan läsas av PIC-processorn och ser likadant ut för alla tänkbara enheter.

All utrustning som man vill ansluta till en BOL måste således vara specialtillverkade för att klara specifikationen på protokollet. Ingen annan utrustning kan användas. Detta innebär att vi gör avsteg från den ideala visionen genom att användaren inte kan använda sig av befintliga hyllprodukter.

Anslutningen av enheterna görs med fördel till COM-porten för att komma bort från den komplexitet som USB kräver. Information om COM-porten är hämtad från [4] och [5].

I BOLen finns en generell drivrutin, Se *drv* i figur 2. Drivrutinen har två uppgifter. Att transportera datan från en COM-port över till servern, eller ta emot datan från servern och skicka ut den på rätt COM-port, samt att upprätthålla GPRS-förbindelsen med servern. Det som gör drivrutinen så generell är att den inte behöver bry sig om vad det är för data som den transporterar, eftersom alla enheter kommunicerar med BOLen på samma sätt med hjälp av protokollet och PIC-processorn



Figur 2: Översiktsbild, certifierad kringutrustning

Protokollet mellan BOL-enhet skulle kunna se ut enligt följande:

Id	Data

Id läggs på data från givaren. Varje Id blir unikt för varje sorts givare. Exempelvis kan en termometer ha Id 1, vindmätare Id 2 o.s.v. Man skulle exempelvis kunna tänka sig att de första 10 bitarna är Id, i sådana fall kan man använda 512 olika enheter.

Drivrutinen i BOLen tar emot ”paketet” och lägger på sitt eget BOL-Id/pin och skickar vidare till servern.

BOL id	Id	Data

Servern har sedan en lista med alla Id, enheter och vilka applikationer som är knutna till varandra.

Id	Enhet	Applikation
1	Termometer	App1
2	Vindmätare	App2

Om man sedan utvecklar en ny typ av enhet som kan anslutas till BOL så läggs dess Id till i servern av Telia eller annan part. Till varje ny enhet måste även en speciellt anpassad applikation skrivas. Servern vet sedan vad för någon enhet som skickat informationen och vilken applikation som skall startas. Applikationens uppgift blir att tolka den information som finns i datafältet i protokollet. Exempelvis för termometer kommer lägsta temperaturen att representeras som Id + 000000.... och den högsta som mätaren kan tolka som Id + 111111.... Applikationen översätter det mottagna värdet till en önskad representation, t.ex. 27^{o} C, som är begriplig för användaren.

3.1.1 Antaganden

Ett antagande är att det fungerar med ett separat protokoll mellan enheterna och BOLen utan att det krävs alltför omfattande intelligens på enheterna. Enligt de ytliga undersökningar vi genomfört verkar detta inte vara några större problem.

Ett annat antagande är att applikationen för en enhet som ligger på servern inte behöver data kontinuerligt. Det vill säga att enheten inte behöver skicka en konstant ström av data till servern. Då finns det stor risk för att GPRS-uppkopplingens begränsade bandbredd inte kommer att hinna överföra datan. En konstant ström av data är också olämpligt ur trafikkostnads hänseende. På grund av detta är det lämpligt om drivrutinerna är skrivna på så

sätt att de bara skickar nödvändig data till servern. Detta sker lämpligast på request basis, d.v.s. drivrutinen sänder ingen data förrän den fått en begäran om data från servern.

3.1.2 Fördelar

Användaren behöver inte installera några drivrutiner, varken i boxen, servern eller på den eventuella www sidan (måste istället göras av Telia eller tredjepart).

Genom att gränssnittet blir samma för all utrustning som sedan tolkas av servern läggs intelligensen i enheterna och på serversidan, vilket gör att BOLen kan göras mycket enkel och resurssnål. BOL-enheten kommer därför troligtvis att kunna göras mycket enkel och billig, eftersom det inte kommer att krävas så stora systemresurser.

3.1.3 Nackdelar

Den största och mest betydelsefulla nackdelen är att det inte går att ansluta befintliga produkter till BOLen.

Eftersom det inte finns färdiga enheter att ansluta, måste de sålunda utvecklas. Detta riskerar att bli ett så kallat moment 22. Eftersom det inte finns några enheter att köpa köper ingen en BOL, och eftersom ingen köper någon BOL vill inga företag utveckla några enheter. Dessutom krävs det specialskrivna drivrutiner både på serversidan och i enheten, som kostar mycket att utveckla.

En annan nackdel är att om det kommer nya tillbehör på marknaden krävs det ett visst utvecklingsarbete av Telias personal, eller tredjepart, innan utrustningen fungerar, eftersom servern måste uppdateras för varje ny enhet. Det är framför allt på grund av dessa nackdelar som vi anser att denna lösning inte är aktuell.

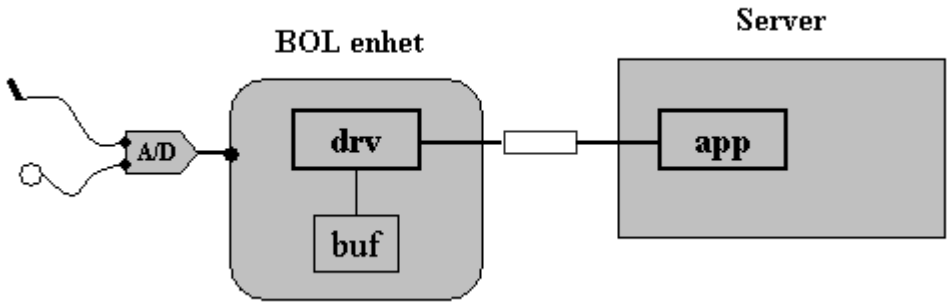
3.2 Analog kringutrustning

Det finns idag en stor mängd analoga enheter, exempelvis termometer, vindmätare och liknande kringutrustning, som kan vara intressanta att använda tillsammans med BOL. Det gemensamma med dessa enheter är att de drivs av likspänning och att de är analoga. För att kunna använda sig av denna typ av enheter är denna lösningsidé intressant.

Idén består i stora drag av att man har sin analoga enhet, givare eller ställare. Enheten ger en analog signal som måste konverteras till en digital signal för att kunna tolkas och

behandlas i BOLen. Detta görs med en A/D-omvandlare om man har en givare, eller en D/A-omvandlare om det rör sig om en ställare. När signalen är konverterad till en digital signal ansluter man denna till en COM-port på BOLen. Detta gör att man kan använda sig av alla analoga enheter som kan drivas av den likspänning som ligger på COM-porten, vanligtvis +12volt.

I BOLen finns en generell drivrutin, se drv i figur 3, som tar hand om den digitala signalen som kommer från A/D-omvandlaren in på COM-porten. Drivrutinen mellanlagrar den digitala signalen i ett buffertminne, buf i figur 3, på BOLen. Datan i bufferten skickas vidare till servern, antingen när det kommer en request från servern eller med ett bestämt tidsintervall. I servern presenteras datan för användaren på ett lämpligt sätt. Om det är det omvända fallet, det vill säga att data skall skickas ut till en ställare, så skickas datan från servern när användaren så vill. Datan lagras i BOLens buffertminne för att genast skickas ut på COM-porten till D/A-omvandlaren av den generella drivrutinen. Den generella drivrutinen bryr sig inte om vad det är för data som kommer från eller skall till COM-porten utan vidarebefordrar enbart datan. På så sätt läggs den mesta intelligensen i servern och då främst i applikationerna, se app i figur 3, som tar hand om datan och interagerar med användaren. Applikationerna som ligger på servern är speciellt utvecklade för varje enhet, d.v.s. de vet hur de skall representera det digitala värdet som kommer från BOLen på ett för användaren begripligt sätt. T.ex. 1001010 blir till en vindhastighet på 8m/s. Eller för det omvända fallet, representerar en för användaren begriplig temperatur på 22 grader Celsius, till en för ställaren begriplig representation 1110010.



Figur 3: Översiktsbild, analog kringutrustning

Den konfiguration användaren måste göra med denna lösning är att han måste tala om för servern vilken kringutrustning som sitter på vilken port på BOLen.

I servern finns en lista med information om BOLen och vilken kringutrustning som sitter på respektive port.

Port	Enhet	Applikation
1	Termometer typ 1	App1
2	Vindmätare typ 1	App2

Enheterna i listan är kopplade till en applikation som är specialskriven för just den enheten. Applikationer skrivs av Telia eller annan part och måste finnas i servern och vara kopplade till rätt enhet i listan för att en enhet skall kunna användas.

3.2.1 Antaganden

Endast analoga enheter är intressanta för BOL.

Den generella drivrutin som ligger på BOLen måste kunna ta emot all data som kommer på COM-portarna och mellanlagra denna på en buffert eller en fil. Den måste även klara av att skicka vidare information från bufferten till COM-porten för att kunna styra alla möjliga typer av analoga ställare.

Alla analoga enheter klarar av att drivas av den spänning som erhålls från COM porten.

3.2.2 Fördelar

Det finns redan idag ett stort antal analoga enheter att köpa. De flesta enheter som vi har sett är relativt billiga att köpa, exempelvis: spänningsmätare, termometer eller vindmätare. [8]

Själva BOL-enheten blir enkel och resurssnål att konstruera, tack vare att intelligensen läggs i applikationerna på serversidan istället för i BOLen.

3.2.3 Nackdelar

Vi låser oss till endast analoga enheter. All annan utrustning kommer inte att kunna användas såvida de inte modifieras till att fungera på samma sätt som det analoga gör.

Idén passar inte in i konceptet. BOLen skulle enbart bli en mobil mätstation eller styrstation, detta finns redan att köpa på marknaden.

Denna idé faller alltså på sin egen enkelhet och är därför inte aktuell att undersöka vidare.

3.3 Tunnling genom GPRS

USB har tillsammans med Windows ett plug ´n play system där Windows automatiskt identifierar enheter och installerar drivrutinerna till denna. Detta har vi gjort en modell för att försöka utnyttja.

Tanken är alltså att man skulle kunna utnyttja USBs automatiska identifiering för enheter som ansluts till BOL-enheten. Detta genom att ”tunnla” data via boxen direkt till servern. Med att tunnla menas att man ”kapslar” in data och gör transporten till BOLen transparent. Med andra ord så är det här ett sätt att lura servern att tro att utrustningen är ansluten på dess USB-port fast den egentligen sitter på BOLens USB-port, se figur 4.

Man kan säga att vi gör en ”förlängning” av USB-porten. På så sätt skulle systemet bli helt plug ´n play och användaren skulle inte behöva göra någon handpåläggning/konfiguration eftersom drivrutinerna skulle installeras automatiskt i servern. Man skulle med denna lösning också kunna använda sig av rena hyllprodukter.

På serversidan får man då problem med att omdirigera den data som kommer från en drivrutin till USB-enheten. Eftersom man vill använda sig av originaldrivrutiner måste man lura drivrutinen att enheten sitter ansluten på USB-porten, när den i själva verket sitter ansluten på BOLen. Man måste läsa av den data som skickas till en USB-port och istället skriva den aktuella datan till en socket.

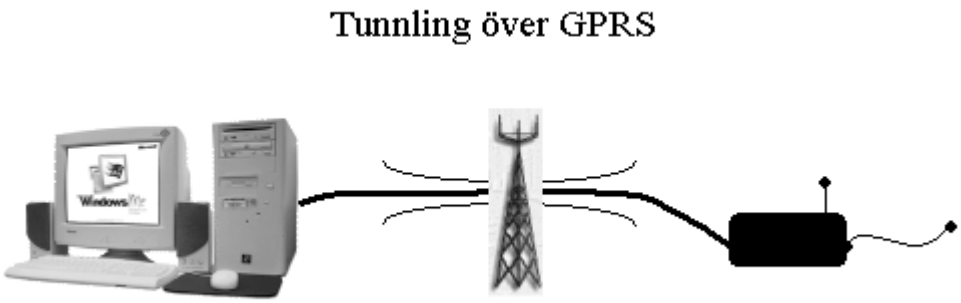
Ett annat problem är att maxhastigheten på en USB-port är 12Mbit.[3] Den teoretiska maxhastigheten på GPRS är ca 170 Kbit/s. Idag finns det inga telefoner/GPRS-enheter som klarar den hastigheten. I realiteten ligger den som bäst runt ca 30 Kbit/s, men ingen kapacitetsgaranti finns. Man måste ha någon form av flödeskontroll för att inte överbelasta överföringen. Denna flödeskontroll måste naturligtvis också finnas ute i BOL-enheten, eftersom kommunikationen är dubbelriktad.

Metoden med ”tunnling över GPRS” ger alltså två stora problem. Hur löser vi skillnaden i hastigheten och hur kan vi omdirigera USB-trafik. Detta är två stora viktiga problem som är mycket svårlösta.

Ett annat problem som vi inte vet svaret på är hur datorn reagerar på den långa svarstiden. Med det menar vi att om USB-enheten varit ansluten på datorns USB-port skulle svarstiderna handla om millisekunder, eller kortare, men då vi använder GPRS handlar det om sekunder. Hur reagerar USB och drivrutiner på att svarstiden blir avsevärt längre än väntat?

En annan intressant fråga är om det är möjligt att läsa av den trafik som går över en USB port. Antingen borde detta gå att lösa genom ren hårdvaruprogrammering, det vill säga att man läser av den trafik som kommer på USB-porten i servern. Informationen skickar man sedan till BOLen. När man får trafik på GPRS förbindelsen från BOLen skickar man denna till USB-porten på servern, detta så att applikationerna man använder tror att enheten är kopplad på serverns USB-port.

Det andra alternativet är någon form av specialtillverkad enhet på serverns USB. Denna skulle ha som uppgift att läsa av signalerna som kommer på porten och skicka tillbaka signalerna till BOL-serverprogrammet som kan skicka dessa till BOLen. Detta fungerar på motsvarande sätt åt andra hållet då det kommer information från BOL-enheten. Då skickar serverprogrammet informationen till den specialtillverkade USB-enheten. Denna skickar tillbaka signalerna på USB porten så att applikationerna luras att tro att enheten är ansluten på servern.



Figur 4: Översiktsbild, tunnling över GPRS

3.3.1 Antaganden

Ett antagande är att USB överhuvudtaget fungerar under förutsättning att hastigheten på en GPRS-överföring är varierande och dessutom låg. Hastigheten på en USB-kontakt är många gånger högre än vad man kan uppnå i de bästa fallen med GPRS.

Ett annat antagande är att det går att ”sniffa” USB trafik

3.3.2 Fördelar

Idén ger en väldigt enkel lösning för användaren. Den innebär automatisk identifiering och installation av drivrutiner på datorn. Befintliga produkter kan användas utan modifiering.

Till en USB-port kan man teoretiskt koppla 127 enheter.

3.3.3 Nackdelar

Hastigheten på GPRS är för låg för att kunna överföra några stora mängder data. Strömmande media, som exempelvis en webbkamera genererar, blir problem. Dessutom varierar hastigheten över GPRS-nätet väldigt mycket och kan ha lång round trip time, RTT, d.v.s. tiden det tar att få en signal att skickas iväg och för ett svar att komma tillbaka. Detta är inte bra eftersom att det finns många enheter till USB som skickar strömmande media, exempelvis kamera och mikrofon.

Problem uppstår med att omdirigera USB-trafiken så man lurar datorn att tro att utrustningen sitter på dess USB-port och inte på BOLen. USB kräver mer systemresurser än vad COM gör, vilket innebär att det blir svårare att tillverka BOLen på en begränsad hårdvara som gör den enkel och billig som slutprodukt. Det finns idag inte speciellt många enheter som kan vara intressanta för BOLen med USB-anslutning. De enheter som är enkla, exempelvis givare och ställare, har nästintill uteslutande COM-port som gränssnitt.

Dessa problem och framförallt problemet med den låga och varierande hastigheten på GPRS gjorde att vi valde bort den här lösningen.

3.4 Certifierade drivrutiner

Tanken med den här metoden är att det skall vara möjligt ansluta alla typer av befintlig kringutrustning, men med drivrutiner som är anpassade för BOL. Detta betyder att för att det skall gå att ansluta en befintlig enhet krävs det att det finns en BOL-certifierad drivrutin tillgänglig. På detta sätt bibehålls visionen på punkten om att tillgängliga hyllprodukter ska kunna användas, men med den modifikationen att det av någon, tillverkaren eller Telia, måste skrivas en speciell BOL-drivrutin. Detta för att all utrustning skall få ett gemensamt gränssnitt mot BOLen, vilket kraftigt förenklar hanteringen av datan i BOLen.

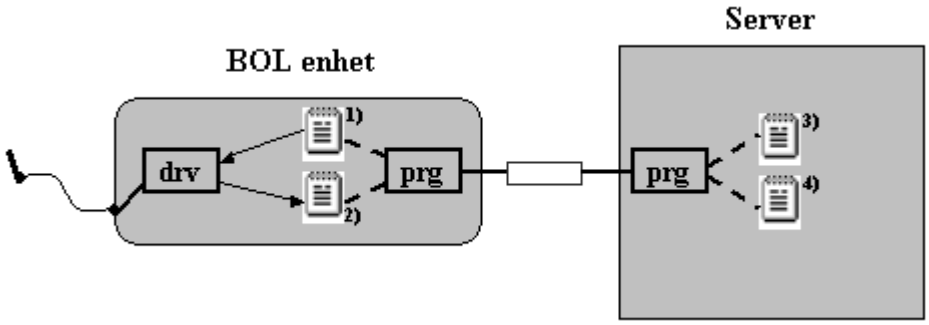
Det speciella med dessa certifierade BOL-drivrutiner är att de måste vara byggda på ett sådant sätt att kommunikationen mellan drivrutinerna och BOLen sker på ett förutbestämt och unisont sätt. Exempelvis arbetar de mot två filer. En fil för indata från enheten och en fil för utdata till enheten, se figur 5. Detta bör fungera tillfredsställande för de flesta typer av tillämpningar men kan bli problem för drivrutiner som normalt använder sig av strömmande

media, t.ex. ljud eller bild. Dessa typer av tillämpningar måste ändå modifieras kraftigt med anledning av GPRS begränsade bandbredd. På grund av att alla drivrutiner arbetar mot BOLen på samma sätt spelar det ingen roll vilken typ av port de använder sig av. Drivrutinerna eller drivrutinspaketet måste alltså bestå av ett exekverbart program som kan startas och stoppas i BOLen. Detta för att det skall vara möjligt att installera och starta respektive stoppa samt byta ut en drivrutin i BOLen.

I servern finns en databas över vilka enheter som finns tillgängliga, alltså de med certifierade drivrutiner. När en användare har anslutit en enhet till sin BOL anger han för servern vilken enhet det är och på vilken port den sitter. Servern letar då upp drivrutinen till enheten i sin databas och skickar den via GPRS till BOLen. När BOLen tagit emot den nya drivrutinen installeras denna på rätt plats och börjar sen jobba mot de två filerna beskrivna ovan. Om det tidigare fanns en drivrutin installerad på den aktuella porten stoppas denna först.

På detta sätt blir installationen av en ny enhet relativt enkel för användaren. Det han/hon behöver göra är att ansluta den nya enheten, komma ihåg vad det är för en enhet och vilken port den sitter på, meddela detta till servern varpå allt sker automatiskt. Användaren behöver endast göra enkel handpåläggning.

Med detta lösningsförslag behåller man visionen om att alla befintliga hyllprodukter skall kunna användas. Men med den modifikationen att en certifierad BOL-drivrutin behövs.



Figur 5: Överskiktsbild, certifierade drivrutiner

Exempel: Utdata som skall transporteras ut till en enhet läggs i fil 3. Datan skickas sedan över till BOLen som lägger datan i fil 1. Drivrutinen drv kan nu läsa av datan, behandla den och skicka ute den till enheten. Indata som fås från enheten lägger drivrutinen drv på fil 2. BOLen skickar datan till servern där den sparas på fil 4. Servern presenterar sedan datan i fil 4 på ett lämpligt sätt för användaren.

3.4.1 Antaganden

Vi antar att det med denna lösning kommer att gå att ansluta alla typer av enheter, bara de har certifierade BOL-drivrutiner. Detta eftersom BOLen som enhet inte bryr sig om vad som är anslutet på dess portar. Den jobbar bara mot filerna som beskrivs ovan. Vi antar också att det går att skriva dessa certifierade drivrutiner till alla möjliga enheter.

3.4.2 Fördelar

Befintliga produkter som finns på marknaden kan användas, om tillverkaren eller annan part skriver BOL-certifierade drivrutiner. BOLen går också att göra relativt enkel, eftersom den inte bryr sig om vad för slags data den transporterar. Installationen av en ny drivrutin blir relativt enkel att utföra för användaren. Endast enkel handpåläggning måste göras.

3.4.3 Nackdelar

En nackdel är att det behövs speciella drivrutiner för alla enheter. Detta innebär att tillverkarna av kringutrustning, eller tredje part, måste komplettera sina drivrutiner med en BOL-drivrutin. Detta kan bli dyrt om BOL försäljningen inte uppnår stora volymer.

En annan nackdel är att när en ny enhet finns tillgänglig på marknaden måste Telia, eller den som driver servern, lägga till drivrutinen i servern. Först då går den nya enheten att installera.

3.5 En jämförelse av konstruktionsförslagen

I detta avsnitt gör vi en jämförelse av huruvida de fyra konstruktionsförslagen löser de problem som konceptet Box On-Line innefattar. Vi har delat upp problemen i tre problemområden, ”Grad av konfiguration”, ”Tillgängliga enheter” och ”Genomförbarhet”, där vi ställer de olika förslagen mot varandra.

3.5.1 Grad av konfiguration

I konstruktionsförslagen ”certifierad kringutrustning”, avsnitt 3.1, och ”tunnling genom GRPS”, avsnitt 3.3, krävs ingen konfiguration av användaren på grund av den automatiska

identifieringen av enheter. Konfigurationen måste istället göras i servern. Dessutom har de två lösningsidéerna ett unisont gränssnitt mot BOLen, vilket gör hanteringen av information i BOLen enkel.

När det gäller förslagen ”analog kringutrustning”, avsnitt 3.2, och ”certifierade drivrutiner” avsnitt 3.4, krävs en viss handpåläggning av användaren. På grund av att det inte finns någon automatisk identifiering av enheterna måste användaren meddela servern vad för enhet han har anslutit till BOLen och till vilken port på BOLen han har anslutit enheten.

Samtliga förslag kräver en viss administration av de serveransvariga. I servern måste det finnas drivrutiner för samtliga enheter, som går att ansluta till BOLen, och applikationer kopplade till dessa för presentation av data för användaren.

Då en ny enhet släpps på marknaden måste en drivrutin och en passande applikation i servern anpassas/läggas till för den nya enheten.

3.5.2 Tillgängliga enheter

Det enda konstruktionsförslaget där man kan ansluta alla befintliga enheter som finns på marknaden är ”certifierade drivrutiner”, avsnitt 3.4, dock med den begränsningen att det måste finnas BOL-certifierade drivrutiner tillgängliga.

I förslaget ”tunnling genom GPRS”, avsnitt 3.3, kan alla befintliga enheter som har USB-gränssnitt användas. Enheter med något annat gränssnitt kan inte användas. Dock är det tveksamt huruvida alla USB-enheter är användbara på BOLen. Exempelvis enheter som skickar strömmande media, eller enheter som är beroende av snabba svarstider.

I förslaget ”certifierad kringutrustning”, avsnitt 3.1, kan alla certifierade enheter anslutas. Dessa certifierade enheter måste dock tillverkas och finns inte att få tag på i dagsläget. Nyutveckling av enheter är kostsamt.

Om förslaget ”analog kringutrustning”, avsnitt 3.2, väljs går det att ansluta alla tänkbara analoga enheter, därav namnet analog kringutrustning. Dock blir man begränsad till just de analoga enheter som passar till BOLen. Inga andra enheter går att använda med denna lösning.

3.5.3 Genomförbarhet

Om vi tittar på konstruktionsförslaget ”certifierad kringutrustning”, avsnitt 3.1, ser vi att det är genomförbart, i alla fall med en större budget. Eftersom det inte går att använda befintliga

enheter och det faktum att nya enheter måste utvecklas för att kunna användas med BOLen är förslaget inte aktuellt att granska noggrannare. Det är dock fullt möjligt att genomföra förslaget.

Förslaget ”analog kringutrustning”, avsnitt 3.2, verkar även det vara fullt möjligt att genomföra, men det faller bort på grund av att det endast går att ansluta analoga enheter. Användningsområdet för en BOL-enhet blir alltför begränsat för att förslaget skall vara aktuellt.

Efter att vi har undersökt förslaget ”tunnling genom GPRS”, avsnitt 3.3, har vi kommit fram till att det knappast är genomförbart i dagsläget. Detta på grund av de begränsningar som GPRS-nätet har idag. I framtiden med större bandbredd är dock förslaget intressant att utveckla, då det erbjuder en hel del fördelar. Se kapitel 4 för mer information om GPRS.

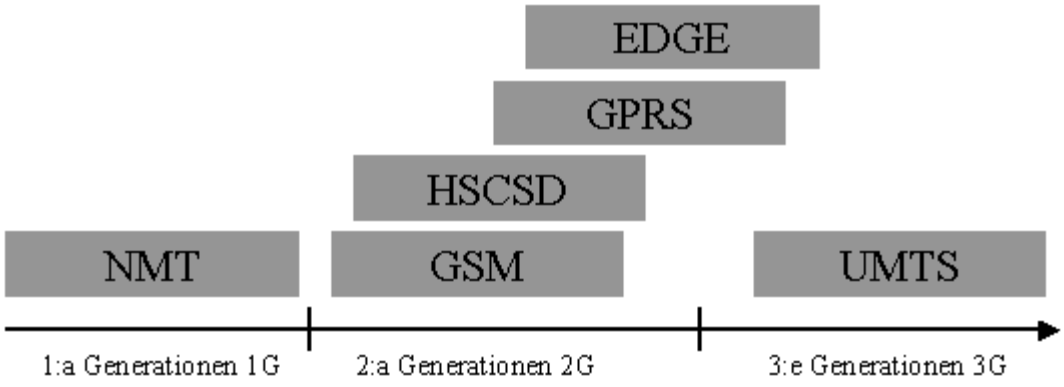
Det konstruktionsförslag som vi har valt att utveckla vidare är förslaget ”certifierade drivrutiner”, avsnitt 3.4. Det förslaget har vi valt därför att det är genomförbart inom ramen för vårt examensarbete och därför att visionen bibehålls i stora drag. Detta med en relativt enkel konfiguration och att alla befintliga hyllprodukter kan användas, dock med den modifikationen att certifierade BOL-drivrutiner måste skrivas.

4 Kort om hur mobila nätverk fungerar

I detta kapitel beskrivs mobila nätverk. Hur mobilnäten har utvecklats med tiden, hur de fungerar och vilka fördelar respektive nackdelar de har. Eftersom dataöverföringen sker via mobilnäten är det viktigt att förstå vilka faktorer som spelar in på hastigheten över förbindelsen och hur de olika näten fungerar. Detta gör att kapitlet är intressant för det här examensarbetet.

4.1 Utveckling av mobiltelefonnäten

I figur 6 visas en bild över hur utvecklingen av mobilnäten har skett. Föregångaren till dagens GSM-nät (Global System for Mobile communications), NMT (Nordic Mobile Telephony), var det första systemet som blev stort kommersiellt. Därför kallat 1:a generationens mobilnät, 1G. Det finns fortfarande i drift men är på väg att ersättas helt till förmån för nyare system. På dagens mobilnät finns det ett flertal påbyggnader och utvecklingar. I vårt examensarbete är det (General Packet Radio Service) som vi använder. EDGE (Enhanced Data rates for GSM Evolution) och GPRS är även kallat 2.5G då de ligger i gränslandet mellan de olika generationerna.



Figur 6: Schematisk bild över mobilnätens utveckling

Alla systemen som kallas 2G (och 2.5G) är egentligen vanligt GSM. Skillnaden är inte så stor i Europa. Alla varianter använder GSM-systemet för taltrafik. Det är endast för dataöverföring som de skiljer sig åt. HSCSD (High Speed Circuit Switched Data) är en kretskopplad teknik för att öka hastigheten på datatrafiken. GPRS är en paketförmedlad teknik för bättre delning

av resurserna i nätet, vilket ger en betydlig hastighetsökning. EDGE är ett system för att ytterligare öka hastigheten på GSM, upp till 384 Kbit/s, men kommer troligen inte att implementeras då man istället satsar på UMTS (Universal Mobile Telefon System) nätet som är en helt ny nätstruktur. [6]

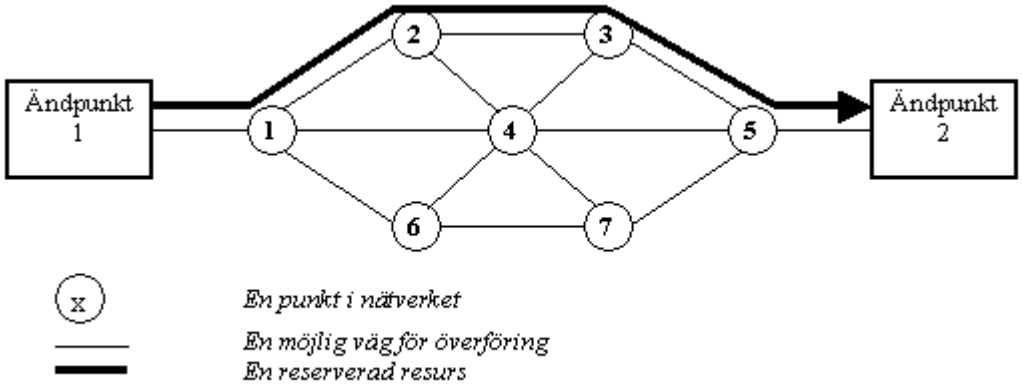
4.2 GSM/GPRS

Som figur 6 visar är GPRS en påbyggnad av GSM-nätet, för dataöverföring. Systemet introducerades 1999. Detta kommer på sikt att kompletteras med det nyare snabbare systemet UMTS, 3G. GSM är avsett för taltrafik och kretskopplad datatrafik, medan GPRS enbart är konstruerat för paketförmedlad datatrafik. GSM-nätet använder frekvensbanden 890-960 MHz och 1710-1880 MHz. I USA används 1850-1990 MHz. Dessa delas upp i 200 KHz kanaler. Hastigheten på GPRS är teoretiskt 171,2 kbit/s, men blir i praktiken mycket lägre. Den varierar också på grund av att många användare delar på samma resurser, på samma sätt som i ett LAN. Detta beror på att GPRS använder tidsluckor [1], se avsnitt 4.5. Förutom att GPRS erbjuder högre hastighet har det andra fördelar.
GPRS är paketförmedlande, GSM är kretskopplat. Detta innebär att man kan betala för mängden data som överförs i stället för att som i GSM betala per tidsenhet. Detta gör att GPRS passar bättre än GSM i konceptet Box On-Line.

Skillnaden blir att man kan vara konstant uppkopplad med en GPRS-mottagare utan att det kostar någonting. Användaren betalar bara för den mängd data som skickas, antalet bytes, och inte den tid som han/hon är uppkopplad.

4.3 Kretskopplade nät, CSD

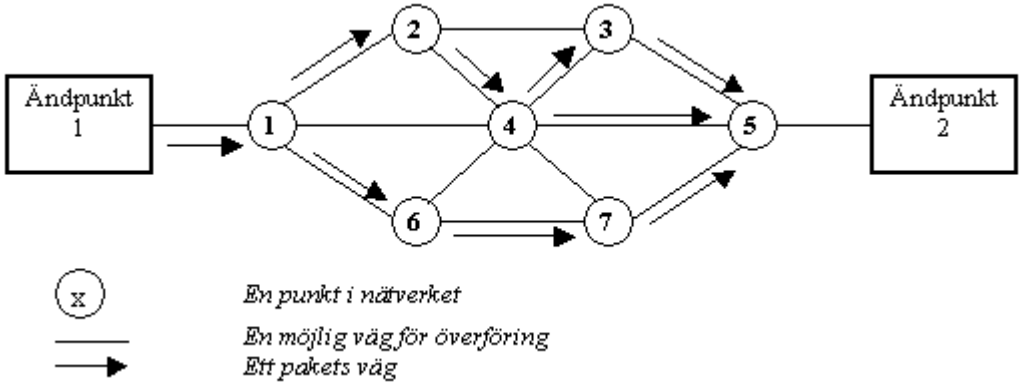
Kretskopplat, samma teknik som det vanliga telefonsystemet använder, innebär att man ”bokar” upp de resurser som krävs för samtalet hela vägen mellan de två ändpunkter som skall kommunicera. Man betalar sedan för tiden som man är uppkopplad, oavsett om kanalen används eller inte. Ingen annan kan använda deras resurs om den står oanvänd, exempelvis om ingen pratar för tillfället, se figur 7. Fördelen med detta system är att man är garanterad en minsta bandbredd på överföringen. Man vet också vilka konstanta fördröjningar som finns i systemet. Med detta menas att eftersom man har en ”bokad” resurs, så är RTT (Round Trip Time) konstant under hela tiden som man är uppkopplad.



Figur 7: Kretskopplat nät, CSD

4.4 Packetförmedlande nät, PSD

Paketförmedlande nät, exempelvis GPRS, fungerar som Internet. Data som skall överföras delas upp i paket som sedan skickas över nätet via lediga resurser. Inga resurser ”bokas” upp och nätet kan användas mycket mer effektivt. Man betalar oftast för mängden data som överförs via nätet. Se figur 8



Figur 8: Paketförmedlat nät

4.5 Tidsluckor i mobila system

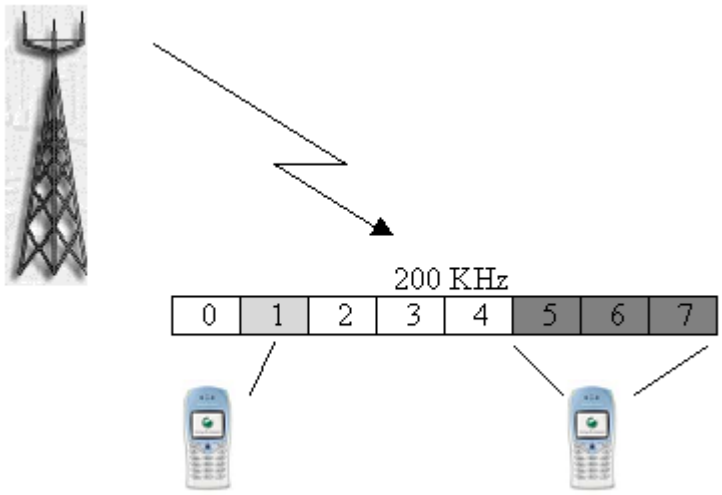
Som beskrivet ovan så delas GSM-nätet upp i ett antal 200 KHz band. Varje 200 KHz-band har åtta luckor, tidsintervall. Man låter alltså ”lucka ett” sända under ett visst tidsintervall. Sedan sänder ”lucka två” nästa intervall, därpå tre och så vidare. När alla åtta sänt börjar det om från början. Detta innebär att åtta telefoner kan dela på samma 200 KHz-band om alla använder taltrafik. Se figur 9. Om en telefon används för datatrafik kan fler än åtta telefoner användas per 200 KHz-band. Detta eftersom en telefon som sänder data inte behöver/får samma tidslucka i alla intervall. Exempelvis ”lucka ett” kan användas av en telefon första sändningen, nästa gång det är ”lucka etts” tur sänder en annan telefon.
Om en basstation bara använder ett band används en eller flera tidsluckor av mobilsystemet för signalering, dessa kan då inte användas kan användas av telefoner.

GSM använder en lucka för taltrafik. HSCSD och GPRS stödjer användandet av flera samtida ”luckor”.

TDMA (Time Division Multiple Access) är en teknik för uppdelning i tidsluckor. Vid låg belastning på nätet kan en användare av GPRS använda alla åtta ”luckor”, såvida telefonen och operatören stödjer det, och hastigheten blir därför hög. Om det däremot är normal belastning varierar antalet tillgängliga tidsluckor och hastigheten blir betydligt lägre än den teoretiska maxhastigheten.

Maxhastighet för en tidslucka är 21,4 kbit/s enligt GPRS definition [2]. I verkligheten är hastigheten operatörsberoende. Telia tillhandahåller 13.4 kbit/s per tidslucka i sitt GPRS-nät. Att hastigheten blir lägre beror på att operatören kan välja att använda olika kodning för trafiken. Denna kodning kallas CS, Coding Scheme [15]. CS finns i fyra olika varianter, CS 1 till CS 4. Telia använder, liksom nästan alla andra operatörer, CS 2. Om de istället valt CS 4 hade hastigheten varit 21.4 kbit/s. CS är metoden för att garantera att trafiken över radiolänken kommer fram, alltså graden av felkorrigering. Om en operatör använder CS 4 kräver det alltså hög täckningsgrad för att inte dataförlusterna skall bli för omfattande. Den teoretiska maxhastigheten en GPRS-telefon kan få, i Telias nät, blir då 13,4 kbit/s* 8 luckor = 107,2 kbit/s

Idag är mobiltelefonerna begränsade till maximalt fyra tidsluckor. En del GPRS-telefoner stödjer färre, (I dagsläget är därför den teoretiska maxhastigheten 53,6 kbit/s (13,4 kbit/s * 4 = 53,6 kbit/s)).



Figur 9: Tidsluckor i GSM / GPRS nät

5 Konstruktionslösning

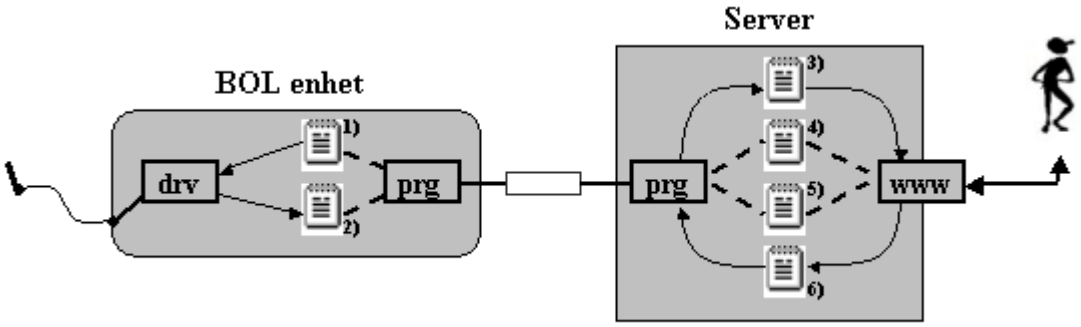
För att implementera vår demoprodukt valde vi det lösningsförslag som beskrivits i avsnitt 3.4, ”Certifierade drivrutiner”. Vi valde denna lösning för att den, enligt oss, på bästa sätt stämmer överens med visionen, samt är möjlig att implementera inom ramen för detta examensarbete. De punkter där vi anser att lösningen stämmer bra överens med visionen är framförallt att graden av konfiguration för användaren är låg. Dessutom kan befintliga produkter på marknaden användas om nya drivrutiner skrivs.

Från Telias sida finns det inga synpunkter beträffande det operativsystem eller vilket programspråk vi ska använda för vår demonstrationsprodukt. Vi har valt att använda oss av Linux som plattform. Det har vi gjort av flera anledningar. En viktig anledning är att det i Linux går att konfigurera kärnan. Detta innebär att man kan ta bort alla de funktioner som inte är aktuella för BOL-enheten. Ett exempel på denna ned skalning är att man tar bort alla de funktioner som har med bildskärmen och grafik att göra. På detta sätt blir operativsystemet som körs på BOL-enheten litet. Det finns färdiga varianter av Linux där installationen ryms på en diskett, 1.44 Mb[10].

Det finns inget motsvarande sätt att exempelvis modifiera Windows. Dessutom kräver de mindre versionerna av Linux mindre datorkraft, vilket gör att de går att köra på väldigt begränsad hårdvara[7]. Detta gör att enheten kan göras mindre och billigare än om man istället hade valt Windows, då den hade behövt vara en PC. Att den blir mindre gör den också mer lämpad för mobilt bruk.

Dessutom är Linux gratis vilket passar produkten väl då den i slutändan skall vara billig för konsumenten att köpa. Eftersom Linux är skrivet i C, kändes det naturligt att skriva vår källkod i C/C++.

I figur 10 nedan ges en övergripande bild av vår demonstrationsprodukt. I vår demonstrationsprodukt använder vi oss av en termometer som enhet på BOLen. Denna kallas Minipic och mäter temperaturen i grader Celsius, [11]. Termometern visas i figuren längst till vänster. Den kopplas till COM-porten på vår laptop som motsvarar BOL-enheten.



Figur 10: Övergripande bild av vår demonstrationsprodukt

Filerna 1,2,4 och 5 finns beskrivna i avsnitt 6.2. Filerna 3 och 6 är de filer som används för att styra servern från ett fristående program. Vi använder i vår demonstrationsprodukt www för interaktionen med användaren. Fil 3 heter bolidut.txt och fil 6 heter bolidin.txt och finns beskrivna i detalj i avsnitt 6.4

5.1 Avgränsning

Att realisera en ”Box On-Line” som en färdig konsumentprodukt är alldeles för omfattande för att rymmas i ett examensarbete. Vi har därför i samråd med handledare och uppdragshandledare gjort en rad olika begränsningar av uppgiften.

Demonstrationsprodukten, vår BOL, består av en vanlig bärbar PC. GPRS-modulen kommer att vara en vanlig mobiltelefon med GPRS-funktion. Vi undersöker inte vilka olika hårdvarulösningar som skulle kunna användas för BOLen.

Vi låter BOLen kommunicera med en central server. Servern som BOLen ansluter till implementeras inte på så sätt som en färdig konsumentprodukt skulle implementeras. I demonstrationsprodukten klarar vår server endast av att hantera en BOL-enhet. Fokus har lagts på hur BOL-enheten och dess gränssnitt skall fungera.

Den realisering som genomfördes är ett test på om vår lösningsmodell är genomförbar i stor skala. Till en början inriktade vi oss på att få BOLen att fungera med avseende på uppdatering av drivrutiner, konfigurering och att kommunikationen mellan klienten (BOLen) och servern fungerade.

I ett första steg skedde kommunikationen över ett LAN (Local Area Network). Därefter gjordes en påbyggnad för att kommunicera via GPRS istället för att använda ett LAN.

5.2 Detaljerad lösning

Vår demonstrationsprodukt implementerades först i en LAN-miljö, när denna fungerade ersatte vi LAN-miljön med en GPRS-uppkoppling. Vi använde mobiltelefonen som databärare. Detta fungerar på samma sätt eftersom vi i båda fallen kör med TCP som protokoll för dataöverföringen. Hastigheten blir dock betydligt lägre. TCP körs ”ovanpå” nätverkslagret, så applikationen som skickar data vet inte om på vilket sätt datan transporteras.

Vi har tidigare antagit att den enklaste lösningen för användaren är att BOLen ansluter till en central server för alla BOL-enheter. Användaren kommer sedan åt sin BOL via en webbsida. Denna sida är personlig och användaren får logga in för att komma åt sin BOL. Här kan användaren konfigurera sin enhet samt se insamlad information. Med denna metod eliminerar vi också risken för att användaren skickar felaktiga kommandon och drivrutiner till BOLen och på så vis orsakar fel. Detta eftersom användaren använder knappar och ”comboboxar” för att kommunicera med sin BOL istället för att skicka egna kommandon. Se avsnitt 6.5 och figur 23.

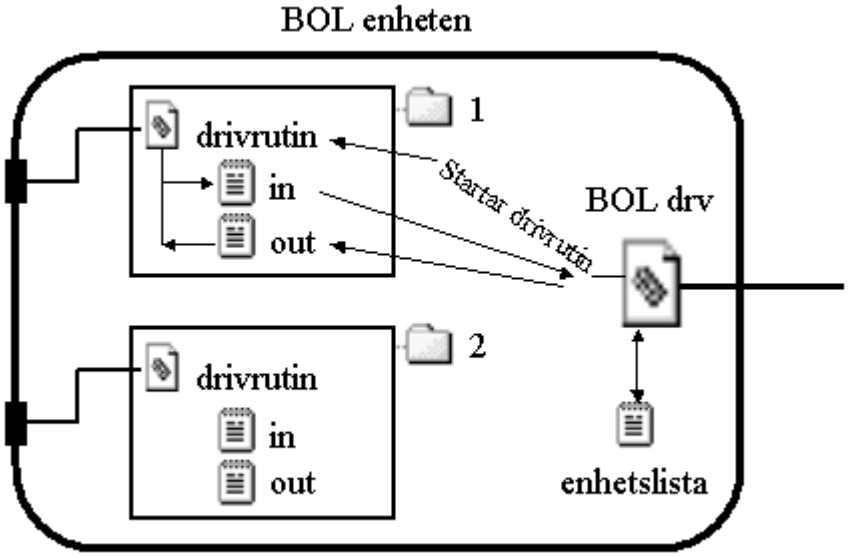
Vi har valt att konstruera den personliga BOL-sidan med hjälp av servlets, för detaljerad information se avsnitt 6.5, ”beskrivning av användargränssnittet”. Detta innebär att för varje ny enhet som skall göras tillgänglig för allmänheten måste det göras en servlet till denna. Detta innebär att tillverkaren, eller tredje part, för att använda vår lösning måste göra både en BOL-drivrutin och en servlet för att presentera informationen via webbsidan.

5.2.1 BOL-enheten

Hur själva BOL-enheten arbetar är beskrivet nedan, se figur 11.

Hjärtat i BOLen är BOL-drivrutinen, *BOLdrv* i figuren. Den har till uppgift att koppla upp BOLen mot servern och sedan sköta kommunikationen mellan BOLen och servern. Den har även ansvar för att stoppa gamla, installera nya och starta drivrutiner för tillkopplade enheter. De drivrutiner som körs i BOLen sparas i drivrutinslistan, *enhetslista*. Detta görs därför att BOLen skall veta vilka drivrutiner som skall startas upp vid eventuell omstart. BOL-drivrutinen ser också till att informationen i infilen, *in,* skickas till servern (om användaren skickat ett request) och att information från servern skrivs till utfilen, o*ut*. Om servern skickar en nedkopplingsbegäran till BOLen ser BOL-drivrutinen till att BOLen kopplar ner förbindelsen med servern och avslutas på ett kontrollerat sätt.

En drivrutin, *drivrutin*, har till uppgift att driva sin enhet och se till att den information som finns i utfilen kommer ut till enheten och den information som skall skickas till servern skrivs ner på infilen. Drivrutinen måste även vid start ta emot det portnummer, som enheten den jobbar mot är ansluten på, som inparameter. Ett anrop av en drivrutin i BOLen skulle kunna, om det sitter en enhet ansluten på port 3, se ut enligt följande: *”drivrutinsnamn 3”*



Figur 11: Översiktsbild av BOL-enheten

5.2.2 BOL-servern

En översikt av hur BOL-servern arbetar och kommunicerar mot fristående applikationer kan ses nedan i figur 12. I vår demonstrationsprodukt använder vi oss av www som gränssnitt mot användaren. Detta är dock inte nödvändigt. Vi har valt den lösningen i vår demonstrationsprodukt. Om man önskar att användargränssnittet skall vara något annat än www så går detta lätt att implementera mot servern. Se avsnitt 6.4 för detaljerad information om hur detta i sådana fall ska göras.

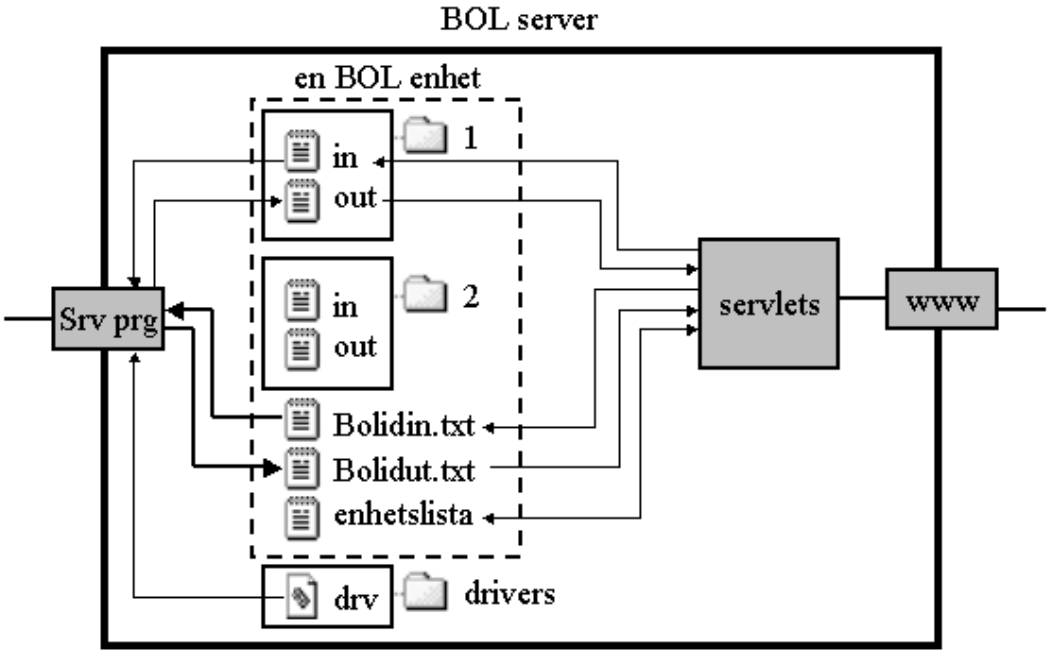
Bilden visar vilka filer som en ansluten BOL-enhet använder. Dessa visas inom det streckade området. Genom att följa flödet av pilarna ses att om man vill skicka ut data till en enhet på BOLen så lägger man informationen i filen *in* för motsvarande port.

Dessutom måste applikationen man använder skriva på filen *bolidin.txt*. Detta är en fil som används för att styra servern. När svaret har kommit från BOL-enheten skriver

serverprogrammet, *srv prg*, på filen bolidut.txt. Detaljerad information om dessa finns i avsnitt 6.4.

I filen *enhetslista* ligger en lista på vad som finns installerat på BOL-enheten. Denna fil läser servern in vid uppstart och skriver aktuell information då en uppdatering skett.

I katalogen *drivers* finns alla drivrutiner som kan installeras på BOLen. En drivrutin *drv* i katalogen *drivers*, se figur 12, skickas till BOLen och läggs som en *drivrutin* i BOLen, se figur 11.



Figur 12: Översiktsbild av BOL-servern

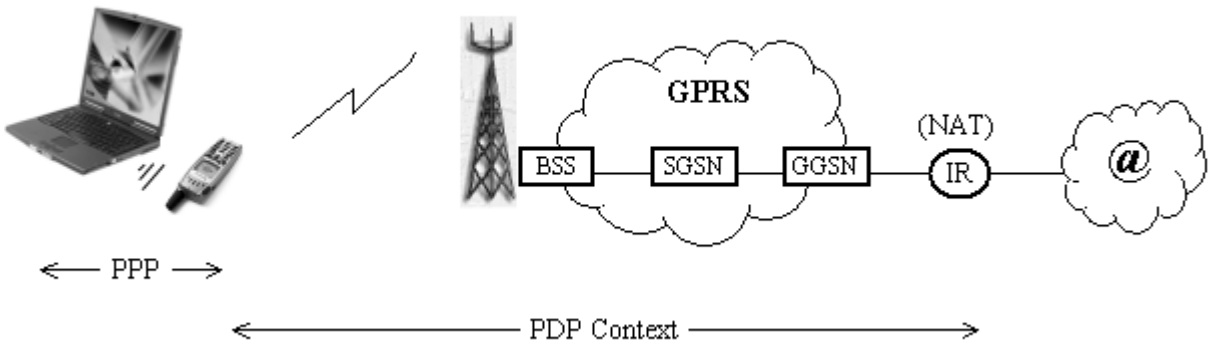
5.3 Etablera GPRS-förbindelse

Vi använder oss i detta examensarbete av en Ericsson R520m[14] som GPRS-modul, se figur 13. Den stödjer maximalt tre tidsluckor. Det innebär att den maximala hastigheten vi kan uppnå via GPRS-uppkopplingen, Telias nät, är 40,2 kbit/s (13,4 kbit/s* 3 luckor = 40,2 kbit/s). I realiteten är hastigheten som vi uppnår betydligt lägre.

Figur 13: Ericsson R520m

Som vi nämnde i avsnitt 5.1, ”avgränsningar”, består BOL-enheten av en bärbar dator, och en mobiltelefon, Ericsson R520m. För att kommunicera med telefonen från datorn använder vi IR-porten på telefonen. På datorn fungerar det på samma sätt som om man istället anslutit telefonen med kabel mot en COM-port.

För att etablera en GPRS-förbindelse sätter man först upp PPP-protokollet mellan datorn, som agerar BOL, och telefonen, se figur 14. Därefter initierar telefonen en PDP Context. Detta innebär bland annat att telefonen får ett privat IP-nummer av GPRS-nätet. NAT-servern sköter kopplingen mellan telefonens interna IP-nummer och dess publika IP-nummer. En PDP Context sträcker sig mellan telefonen och NAT-servern, GGSN i figuren är en gateway mot Internet.



Figur 14: Översiktsbild av GPRS-uppkoppling

5.3.1 AT-kommando

Attention (AT) kommandon är en uppsättning kommandon för att kommunicera med modem. Eftersom telefonen agerar modem använder man AT-kommandon för att skicka kommandon till telefonen.

Med AT-kommandon kan vi utföra alla önskade funktioner på telefonen såsom att ringa, skicka SMS och ändra i adressboken. När vi skall koppla upp en GPRS-förbindelse använder vi oss av en rad olika AT-kommandon. Dessa innehåller information som telefonen behöver för att PDP Context skall kunna sättas upp korrekt, exempelvis olika parametrar om nätet och information var och hur telefonen skall anslutas.

En lista över alla tillgängliga AT-kommandon som kan utföras på en Ericsson R520m finns på Ericssons hemsida [9]. Ett sätt för att enkelt testa olika AT-kommandon är att använda något program som kan skicka information över en COM-port. Vi använde programmet minicom på Linux när vi testade kommandon till telefonen.

För att demonstrationsprodukten skall bli så automatiserad som möjligt har vi skrivit ett script som etablerar GPRS-förbindelsen.

6 Implementation

I detta kapitel beskrivs den demomiljö som vi använt oss av i arbetet på Telia. Sedan beskrivs detaljerat hur vi implementerat vår demonstrationsprodukt. Gränssnittet mellan BOL-server och webbservern förklaras.

6.1 Beskrivning av vår demomiljö

BOL-enheten i vår demonstrationsprodukt består av en vanlig bärbar PC och en mobiltelefon, Ericsson R520m. Telefonen har agerat GPRS-modul i BOLen. Datorn i vår BOL och mobiltelefonen kommunicerar med hjälp av IrDa (Infrared Data Association). Servern består av en vanlig standard-PC och klarar endast av en ansluten BOL-enhet. Se figur 15

Vår demomiljö består av följande produkter:

Bärbar PC:
Dell Latitude C600
Processor: Intel PIII, 600MHz
RAM: 256 Mb
Hårddisk: 15 Gb
Ir port: SMC Ircc – snabb Ir-port
OS: Linux Mandrake 8.1 [12]

GPRS enhet:
Telefon, Ericsson R520m [14]

Testenhet:
Minipic Web Thermometer [11]

BOL-server, WWW-server:

Dell Optiplex, GXi

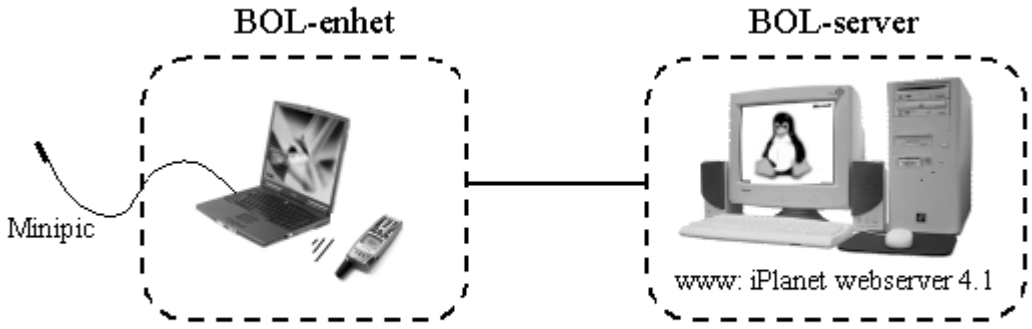
Processor: Intel PII, 200MHz

RAM: 32 Mb

Hårddisk: 1 Gb

OS: Linux Mandrake 8.1 [12]

WWW Server: iPlanet Web server 4.1 [13]



Figur 15: Bild över demonstrationsprodukten

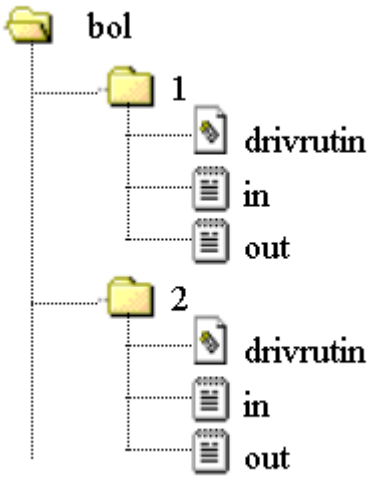
6.2 Specifikation

Som beskrivits tidigare i kapitel 5 använder sig BOL-enheten av två filer för att kommunicera med en enhet. Dessa filer är ”**in**” och ”**out**” och visas i figur 16, jämför med figur 11. Filerna ligger under en katalog som har samma namn som porten enheten är ansluten till. Även drivrutinen för enheten ligger i denna katalog, se figur 16 nedan.

På ”**in**” skrivs den data som kommer från enheten ansluten på BOL. Exempelvis, om Minipic, enheten i vår demonstrationsprodukt, är installerad så lagrar drivrutinen temperaturen från enheten på denna fil. När BOLen får en ”begäran om att hämta data”, en request, från servern läser den i denna fil och skickar innehållet som svar.

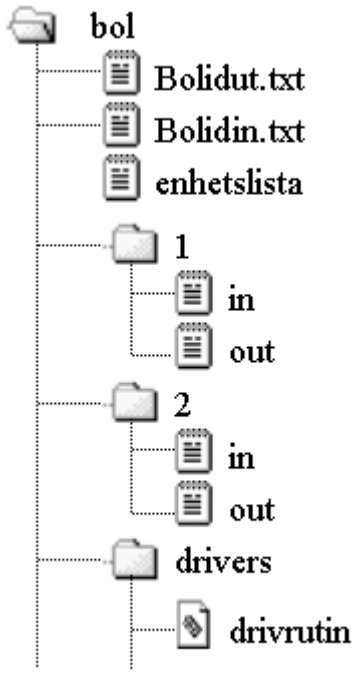
I ”**out**” filen finns det innehåll som användaren vill skicka till den aktuella enheten som är ansluten till BOLen. Drivrutinen läser sedan av denna fil och skickar innehållet till enheten.

Exempelvis om enheten är en resultattavla och användaren vill skicka ut text skickar servern resultatet till ”**out**” där drivrutinen läser av innehållet och matar ut detta till resultattavlan.



Figur 16: Katalogstrukturen på BOL-enheten

Katalogstrukturen på servern är motsvarande den som finns på BOLen plus ytterligare ett par filer. Jämför BOL-server, figur 17 och BOL, figur 16. Det som skiljer dem åt är filerna ”**bolidin.txt**” och ”**bolidut.txt**” samt ”**enhetslista**”, jämför med figur 12. Bolidut.txt och bolidin.txt är de filer som används för att styra servern från andra program. I vårt fall styr vi servern från vår webbsida med hjälp av servlets, se figur 12. Se avsnitt 6.5 för information om www sidan. Enhetslista är en fil som servern läser in vid uppstart. Den innehåller information om vad som är installerat på BOLen. Alla filer är vanliga textfiler, vilket innebär att du kan styra servern genom att skriva i textfilerna. Se avsnitt 6.4 för mer detaljerad information. Dessutom innehåller servern en katalog, drivers, där alla drivrutiner som kan installeras på en BOL ligger.



Figur 17: Katalogstrukturen på BOL-servern

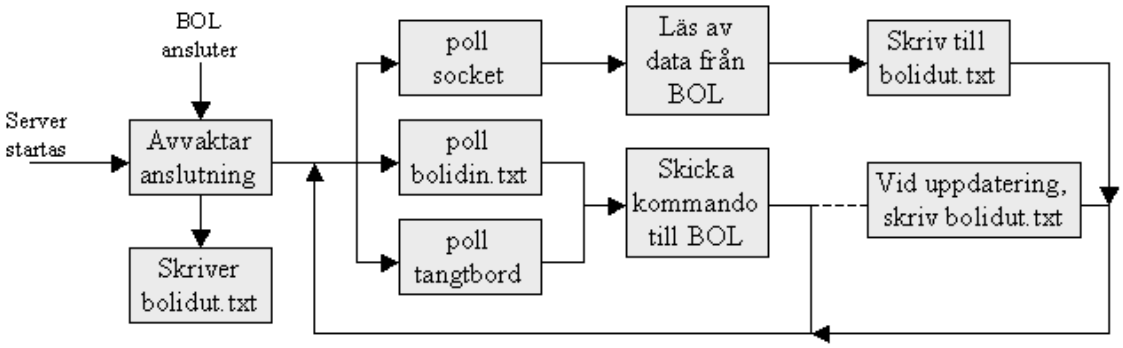
6.2.1 Översiktliga programflöden

BOL-server:

I figur 18 nedan visas en övergripande bild av hur BOL-servern i vår demonstrationsprodukt fungerar. Figuren visar det översiktliga flödet som vår BOL-server använder.

Vid start av servern läses enhetslista in, detta är ej med i figuren. Denna används för att servern skall veta vilka enheter som är installerade på BOLen. Se även figur 17. Då en användare valt att skicka en uppdatering, det vill säga att lägga till/ta bort en drivrutin för någon enhet som han/hon har anslutit till sin BOL, skriver serverprogrammet den aktuella enhetslistan på filen, ej med i figuren. Protokollet för överföringen mellan BOL och server är TCP. Ovanpå det kör vi vårt BOL-protokoll. Se avsnitt 6.3.

Det centrala i servern är de tre rutor som kallas ”poll socket”, ”poll bolidin.txt” och ”poll tangentbord”. Att man pollar innebär att man hela tiden väntar på att ”innehållet” skall ha förändrats. Det vill säga att om man pollar på exempelvis tangentbordet så ger det ett positivt utslag om en tangent har blivit nedtryckt. På detta sätt kan man kontrollera ett antal händelser samtidigt. Endast en poll-händelse kan vara sann i ett givet ögonblick.



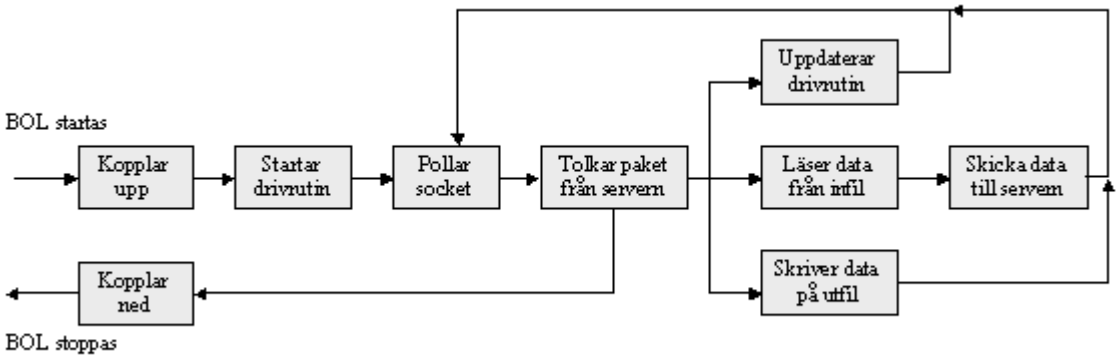
Figur 18: Övergripande bild av programflödet i BOL-servern

BOL-enhet:

På samma sätt som vi beskrivit programflödet för servern i figur 18 har vi nedan beskrivit programflödet för BOL-enhetens drivrutin. Se figur 19.

Det första som händer när BOLen startas är att BOLen kopplar upp en GPRS-förbindelse till servern. När uppkopplingen är klar och fungerar startar BOLen de eventuella drivrutiner som ligger lagrade i drivrutinslistan. När detta är gjort är BOLen klar att användas och BOL-drivrutinen börjar att polla den inkommande socketen för att läsa in och tolka de paket som kommer från servern. Paketen som kommer från servern kan delas in i fyra grupper:

- Uppdateringar av drivrutiner, installation eller avinstallation av drivrutiner.
- Användaren har skickat en begäran om att hämta data från någon enhet, Get request. Datan läses från infil.
- Begäran att skicka data, Put request, från servern/användaren. Datan i paketet skrivs ner på en utfil som drivrutinen till enheten läser av.
- Nedkopplingsbegäran från servern. BOLen skriver ner drivrutinslistan på fil, kopplar ner GPRS-förbindelsen och avslutar sig själv.



Figur 19: Övergripande bild av programflödet i BOL-enheten

6.3 Protokollet mellan BOLen och servern

Kommunikationen mellan BOL-enheten och servern sker via ett protokoll som vi definierat enligt figur 20. Det underliggande transportprotokollet är TCP.

Protokoll Specifikation

SizeOf	Bol Id	ComPort	ControlField	Data

Figur 20: Protokollspecifikationen mellan server och BOL

SizeOf anger storleken på datafältet som skall överföras.

Bol Id är ett Id nummer från den aktuella BOL-enheten.

ComPort anger vilken COM port som är den som paketet avser.

ControlField talar om vilken typ av paket som skickas. ControlField kan ha en rad olika värden, se tabell nedan.

Data är fältet där eventuell data lagras. Den lagras som en sträng av varierande längd.

Exempel: Så här ser det paketet ut som servern skickar till BOLen när användaren gör ett GET request på port 1:

SizeOf = 0

Bol Id = Den aktuella BOLens id

ComPort = 1

ControlField = 2

Data = ””

PUT svaret som BOL-enheten skickar tillbaka skulle kunna se ut så här:

SizeOf = 12

Bol Id = Den aktuella BOLens id

ComPort = 1

ControlField = 3

Data = ”09.53 = 18.9”

Möjliga värden på ControlField		
Namn	*Värde*	*Förklaring*
UPDATE	0	Paketet är en uppdatering
KILL	1	Avinstallerar en drivrutin
GET	2	Servern skickar en request till BOL*
PUT	3	Anger att data skickas till mottagarsidan**
KA	4	Keep Alive skickas för att upprätthålla kontakten
BYE	5	Servern skickar en nedkoppling till BOL
ERROR	6	BOL skickar felmeddelande***

Tabell 1: Olika värden som ControlField kan anta

* *Ett GET kan bara skickas från servern. BOL svarar med att skicka ett PUT paket med data.*

** *Om ett PUT skickas från servern innehåller det data till en utenhet på någon COM-port på BOLen. Om BOLen skickar ett PUT så är det ett svar på en request (GET) från servern*

*** *ERROR kommer bara från BOL till server. Datafältet innehåller felmeddelandet.*

6.4 Kommunikation mellan server och servlet

Vi skapar interaktionen med användaren via en personlig BOL-webbsida. Dessa sidor skapar vi med hjälp av ett antal olika servlets, se figur 12. Dessa behöver kunna kommunicera med BOL-servern för att användaren skall kunna styra sin ”Box On-Line”. Se avsnitt 6.5 för en beskrivning av webbsidan. Kommunikationen sker genom två filer, ”**bolidin.txt**” och ”**bolidut.txt**”, där bolid är det id-nummer som den aktuella BOLen har, exempelvis 1024. Det innebär att filerna i exemplet skulle heta 1024in.txt och 1024ut.txt. Information om var de ligger i katalogstrukturen finns under avsnitt 6.2

Vi har valt att sköta interaktionen med användaren med hjälp av en webbsida, men man kan lika gärna göra detta med något annat program som är specialskrivet för ändamålet. Enda kravet är att applikationen skriver och läser på filerna på det sätt som är beskrivet nedan.

Liksom alla andra filer vi använder oss av är även dessa två filer rena textfiler. Varje rad har en speciell betydelse enligt exemplet nedan. Kommandoraden i filerna talar om för server/servlet vilken typ av kommando som filen anger.

De olika värden som kommandoraden i filerna ovan kan anta visas nedan. Dessa värden är **inte** desamma som ControlField i avsnitt 6.3.

Kommandoraden olika värden

0	Update	Uppdatering av drivrutin
1	Get	Begäran om information
2	Put	Skicka information till BOL
3	Kill	Ta bort drivrutin på BOL
4	Error	Felmeddelande
5	Msg	Meddelande från BOL/server
6	Bye	Nedkoppling av BOL

Nedan finns en specifikation på bolidin.txt, Den används i vår demoprodukt av en servlet för att styra servern. Bolid är här 1024.

1024in.txt

Kommandorad	Kommando till BOL
Portnummer	Vilken port som är aktuell
Ev. Sökväg	Sökväg till drivrutin

Exempel 1:	**Exempel 2:**
1	0
1	2
	/bol/drivers/Minipic

I exempel 1 skickas en begäran om information från BOLen, ett GET kommando skickas till servern på port 1. I detta fallet är raden om sökväg tom.

I exempel 2 skickas en uppdatering till BOLen. Uppdateringen skall installeras på port 2. I detta fall kommer drivrutinen Minipic att skickas.

Nedan visas en specifikation av filen bolidut.txt. Den används av servern för att skicka information till omvärlden, i vår demoprodukt till en servlet. Även här är bolid 1024.

1024ut.txt

Password	Lösenord till den aktuella BOLen
Kommandorad	Svar till användaren
Meddelande	Meddelande från BOL/Server
Port1	Vad som finns installerat på port 1
Port2	Vad som finns installerat på port 2

Exempel 1:	**Exempel 2:**
Boxonline	Boxonline
1	4
Svar mottagit	Kunde inte starta drivrutin
Ledig	Minipic
Minipic	Ledig

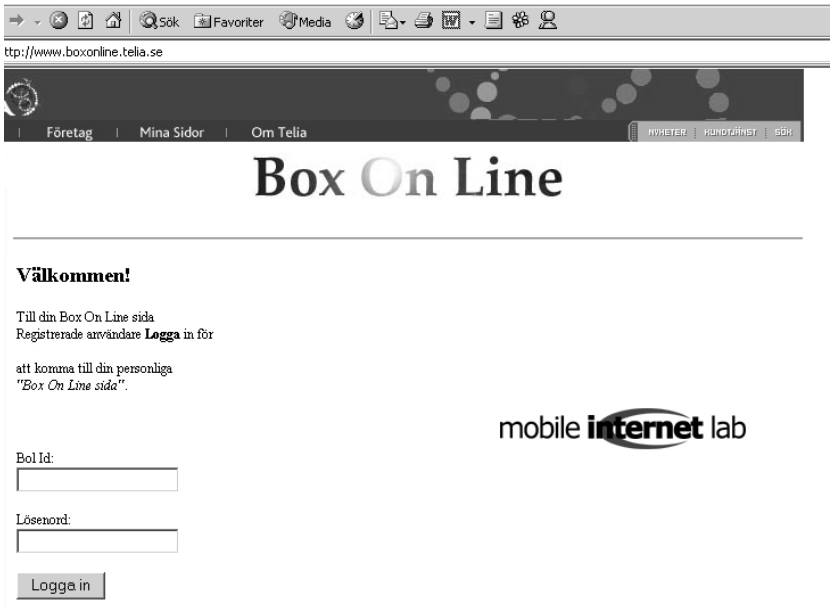
I exempel 1 talar servern om för webbsidan att lösenordet till den aktuella BOLen är ”Boxonline”, webbsidan får sedan kontrollera att användaren har aktuellt lösenord. Servern har mottagit svaret på en GET-begäran från BOLen. Meddelandet i detta fall betyder inte så mycket, men visas i ett textfält på webbsidan. Dessutom säger servern att BOLen har port 1 ledig och Minipic installerad på port 2.

I exempel 2 ovan är lösenordet ”Boxonline”. Det har blivit något fel i BOLen. Drivrutinen kunde inte installeras av någon anledning. Meddelandet visar detta för användaren.

6.5 Beskrivning av användargränssnittet

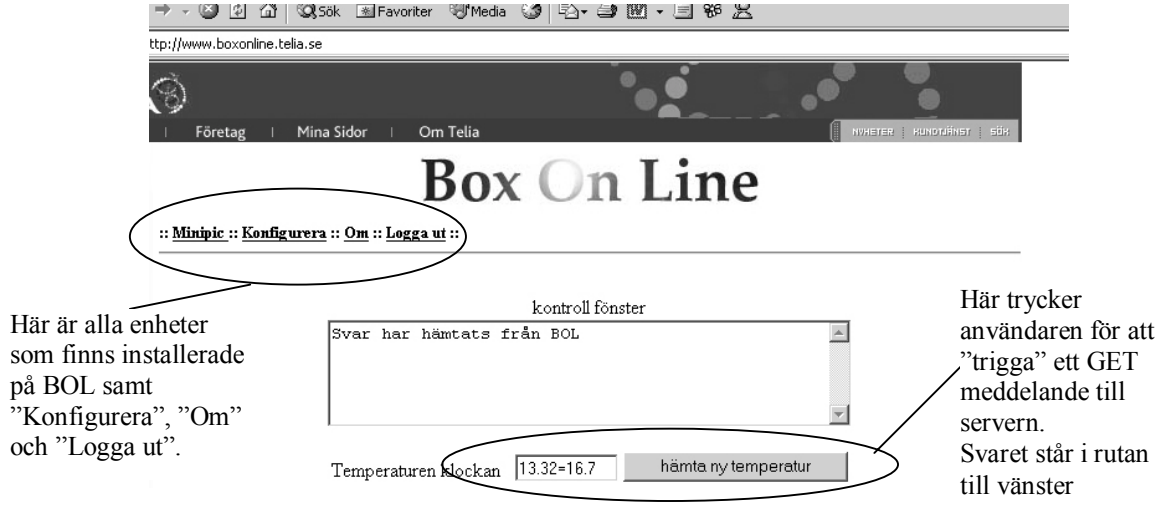
Vi valde en webbsida som gränssnitt mot användaren. Detta för att inte låsa oss till Linux som operativsystem. Webbservern i vår demomiljö är ”iPlanet Webserver 4.1” [13]. Denna har inbyggt stöd för servlets.

Användaren som köpt sin BOL får reda på dess id-nummer och ett lösenord. Detta använder han för att logga in på sin personliga BOL-sida. Utseendet kan du se i figur 21. För att generera sidan använder vi oss av en inloggningsservlet.



Figur 21: Inloggning till personlig BOL-sida

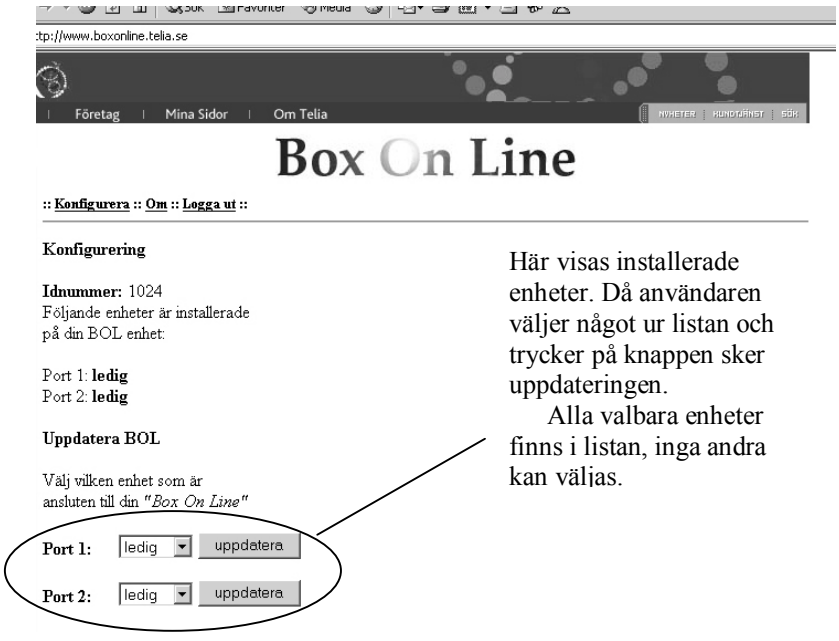
När användaren loggat in kommer han/hon till en sida liknande den i figur 22. I menyraden visas alla enheter som finns installerade. Om inga enheter finns installerade så finns där endast valen ”Konfigurera”, ”Om” och ”Logga ut”. Se förklaring till menyn i figur 22. Menyn skapas av en meny-servlet. Om användaren installerat en enhet, exempelvis Minipic, blir den valbar i menyn. Då användaren trycker på ”Minipic” laddas en minipic-servlet. Det är den sida som användaren använder då han/hon vill läsa av temperaturen vid sin BOL. Knappen skickar en begäran om information, GET, till servern. När svaret kommit tillbaka från BOLen visas det i rutan till vänster om knappen. Eventuella meddelanden från server/BOL visas i ”kontrollfönster”, se figur 22.



Figur 22: Användandet av Minipic

Då användaren skall installera en ny enhet, eller ta bort en befintlig, så går han/hon in på sin konfigurationssida, se figur 23. Där visas vilket id-nummer som BOLen har samt vilka enheter som för tillfället körs på BOLen.

Då användaren önskar ändra väljer han/hon önskad enhet ur listan och trycker på uppdatera knappen. Alla enheter som kan installeras på BOLen finns i listan. Användaren har inte någon möjlighet att själv skicka egna drivrutiner till sin BOL.



Figur 23: Konfiguration av ”Box On-Line”

7 Testning

Eftersom interaktionen med användaren sker via ett webbgränssnitt kan inte felaktig information skickas till BOLen. Webbsidan ger alltså användaren begränsade möjligheter till kommunikation med BOLen. På grund av webbsidan och det faktum att vi endast har en enhet att testa på vår BOL är antalet testfall begränsade. De testfall som vi kan genomföra är

Lägga till en enhet
Hämta ett värde från en enhet
Ta bort en enhet

Dessa testfall kan varieras inbördes i ordning och alterneras mellan port 1 och port 2. Testfallen har testats ett stort antal gånger i varierande ordning och i samtliga fall med lyckat resultat. Utomstående personer har också testat vår demonstrationsprodukt. Inte i något fall får vi några fel. Med utgångspunkt från ovanstående tester hävdar vi att vår lösning och implementation fungerar utan problem, med hänsyn till de problem som beskrivs nedan.

Ett problem finns med fallet att lägga till en enhet. Nämligen att vi inte avvaktar något svar från BOLen vid uppdateringen. När en användare tryckt på uppdateringsknappen skickas ett uppdateringskommando till BOLen samtidigt som huvudwebbsidan laddas. På huvudsidan kan användaren trycka på de olika enheterna, ”Konfigurera”, ”Om” och ”Logga ut”. Detta innebär att användaren kan trycka på exempelvis ”hämta värde” innan drivrutinen hunnit installeras på BOLen. Detta kan, beroende på när användaren trycker, medföra fel. Problemet kan lätt åtgärdas genom att man väntar på en ACK från BOLen. Detta har vi dock inte hunnit implementera i vår demonstrationsprodukt.

Användaren kan också ”skapa” ett fel om han/hon lägger till en enhet på BOLen som inte finns ansluten i verkligheten. Servern visar webbsidan för den aktuella enheten, då kan användaren kommunicera med en enhet som inte är ansluten. Servern avvaktar svar från BOLen, men eftersom inget sådant kommer hänger servleten sig i avvaktan på svar. Detta kan avhjälpas med en timeout på servern. Detta har vi heller inte hunnit implementera i vår demonstrationsprodukt.

Ett annat test som vi genomfört är att låta BOLen vara online under längre tid. BOLen skall skicka ett ”keep alive”, KA, meddelande var tredje minut som BOLen är inaktiv. Detta

för att NAT-servern inte skall tappa kopplingen mellan telefonens privata IP-adress och dess publika IP-adress. Tiden som NAT-servern behåller kopplingen, om ingen trafik förekommer, är satt till sex minuter i Telias mobiltelefonnät.

Sammanställning av ”Keep Alive” test

Test nr	Tid Online (timmar)	Lyckades
1	**16**	**JA**
2	**18**	**JA**
3	**109**	**JA**

Tabell 2: Sammanställning av ”Keep Alive” test

Det finns andra problem som uppstått då vi överfört vår programlösning från ett LAN till ett GPRS-nät.

Första problemet med att använda GPRS-nätet är att när vi kör vårt uppkopplingsscript brukar nästan alltid det första försöket misslyckas medan det andra alltid lyckas. Varför detta fenomen inträffar vet vi inte idag.

Det andra problemet är att tiden det tar att skicka ett paket över GPRS är så mycket längre än tiden det tar att skicka ett motsvarande paket över ett LAN. Detta innebar att vi var tvungna att modifiera funktionen för nedkoppling av BOLen för att den skulle fungera över GPRS. Kommandot, BYE, hinner inte skickas iväg över GPRS-förbindelsen innan servern kopplar ned socketen. Detta innebär att klienten får ett felmeddelande istället för ett nedkopplingskommando. Detta problem åtgärdades genom att lägga in en fördröjning på åtta sekunder i servern. På detta sätt hinner data skickas över socketen och nedkopplingen sker på ett korrekt sätt.

8 Resultat och rekommendationer

För att uppnå huvudmålet med examensarbetet har vi undersökt flera olika metoder för att lösa uppgiften. Dessa metoder har vi sedan analyserat för att kunna välja ut den mest lämpade. Utifrån den valda metoden implementerade vi vår demonstrationsprodukt. Detta för att verifiera att vår valda lösningsidén fungerade.

I kapitel 3 beskriver vi de fyra olika konstruktionslösningar som vi undersökte. Av dessa var det endast ”certifierade drivrutiner”, avsnitt 3.4, vi fann möjlig att implementera. Vi anser dock att lösningsförslaget ”Tunnling genom GPRS”, avsnitt 3.3, är intressant att undersöka vidare. Speciellt då överföringshastigheten i det nya mobilnätet UMTS är högre än vad GPRS kan erbjuda idag. De övriga två förslagen ”Certifierad kringutrustning” och ”Analog kringutrustning” är inte intressanta att undersöka vidare, då de passar dåligt in i den realistiska visionen, se avsnitt 3.1.3 och 3.2.3. Det finns säkert andra enklare och bättre metoder för att lösa uppgiften men inom ramen för detta examensarbete såg vi inga andra alternativ.

Enligt testresultaten i kapitel 7 hävdar vi att vår demonstrationsprodukt fungerar utan problem, enda problemet som kvarstår är att det oftast krävs två uppkopplingsförsök med GPRS. Detta är dock inget stort problem och mot bakgrund av de övrigt lyckade testfallen drar vi slutsatsen att konstruktionslösningen, ”certifierade drivrutiner”, går att genomföra i större skala ända till en färdig konsumentprodukt. Idén går dock att utveckla vidare med andra lösningar. Vi valde att implementera vår demonstrationsprodukt på det sätt att alla BOL-enheter ansluter till en stor central server. Istället kan man välja lösningen att BOL-enheten ansluter sig mot användarens dator, med alla drivrutiner lagrad på en central server. Med detta menas att när användaren vill uppdatera sin BOL med en ny drivrutin så hämtas drivrutinen från en central server istället för att användaren har filen på sin dator. Gränssnittet mot användaren behöver som nämnts tidigare inte vara en webbsida.

För att vår demonstrationsprodukt skulle fungera utan problem behövde vi modifiera nedkopplingsfunktionen så att servern väntar ett antal sekunder innan den stänger socketen. Detta problem väntade vi oss inte eftersom det inte existerade när vi använde oss av ett LAN. Dataöverföring via GPRS gav betydligt längre RTT-tider än vad som var väntat. Vi hade väntat oss att hastigheten skulle vara ganska stabil runt 30 kbit/s. Detta innebär att vi räknade med hastigheter som under vissa tidsperioder var snabbare och ibland långsammare, men vi hade inte räknat med att den skulle vara konstant lägre.

Java är ett plattformsoberoende programspråk, det vill säga samma program kan köras på olika operativsystem såsom Linux och Windows. För vår demonstrationsprodukt innebär det att om vi hade valt Java som programspråk istället för C/C++ hade vi varit mer flexibla med valet av operativsystem. Nu valde vi från början Linux som operativsystem och C/C++ som programspråk därmed kunde vi enbart använda Linux, både på servern och på BOL-enheten. Med Java hade vi på ett enkelt sätt kunnat använda Linux på BOLen och något annat operativsystem på servern om vi så önskat, exempelvis Windows. Detta gör att vi hade valt Java istället för C/C++ om vi skulle göra om uppgiften. Detta blir speciellt viktigt om man tänker sig att BOLen istället skulle ansluta mot en användares dator. För vanliga datoranvändare är Windows nästintill standard i dagsläget.

9 Slutsatser

Vi har i detta examensarbete, på 10 poäng, arbetat halvtid under tjugo veckor. Arbetet har under hela tiden varit utvecklande. Vi har av arbetet fått kunskap om hur man kommunicerar med en telefon via AT-kommandon. Dessutom har vi lärt oss hur GPRS fungerar, exempelvis hur man kopplar upp och kommunicerar. Vi har också studerat för och nackdelar med detta system. Dessutom har vi utökat våra tidigare kunskaper om klient/server programmering.

Vi har lärt oss hur man planerar och genomför en förstudie för en produkt. I och med att Telia gav oss fria händer, med avseende på slutproduktens utseende och funktionalitet, för projektet så har vägen mot slutmålet inte varit spikrak. Konceptet har varit inne på flera olika sidospår, beskrivna i kapitel 3. Enligt oss har vi kommit fram till den bästa lösningen inom ramen för den tid som examensarbetet förfogar över.

Svårast har varit att i planeringsfasen bestämma oss för vilken lösning som vi skulle lägga energin på och undersöka noggrant. Det gäller att överblicka en lösning för att se möjligheter och problem med denna, helst redan i förstudien så att man inte lägger för mycket tid på en idé som är omöjlig att genomföra. Att med relativt begränsade kunskaper kunna ta beslut som är viktiga för slutprodukten. Hade man haft en redan specificerad uppgift att genomföra hade man inte haft dessa problem.

Tidsåtgången för de olika momenten kan grovt uppskattas som att de tagit en tredjedel vardera. En tredjedel för planering och undersökningar, en tredjedel för implementation av vår demonstrationsprodukt och en tredjedel för att skriva rapporten.

Vi är nöjda med resultatet av vårt examensarbete, såväl rapporten som demonstrationsprodukten.

Referenser

[1] http://www.gsmworld.com/technology/gprs/intro.shtml , 2002-02-13

[2] http://se.allwap.com/gprs/ , 2002-03-09

[3] http://www.usb.org/ , 2002-01-20

[4] http://www.arcelect.com/rs232.htm , 2002-01-20

[5] http://www.howstuffworks.com/serial-port.htm , 2002-01-20

[6] http://www.cs.kau.se/~johang/dk2/access2pg.pdf, 2002-03-12

[7] http://www.linux.org/hardware/index.html, 2002-03-12

[8] http://www.elfa.se/se/ 2002-03-12

[9] www.ericsson.com/mobilityworld/developerszonedown/downloads/docs/r520/r520_at_commands.pdf

[10] http://www.math.psu.edu/morris/deptpage/distrib.html, 2002-04-23

[11] http://www.rlkonsult.se, 2002-04-23

[12] http://www.linux-mandrake.com/en, 2002-04-23

[13] http://wwws.sun.com/software/download/inter_ecom.html, 2002-04-23

[14] http://www.sonyericsson.com/se/, 2002-04-23

[15] Bates, J, Regis, Professional GPRS, McGraw-Hill, 2002

IrDa och GPRS för uppkopplingen

http://irda.sourceforge.net/ 2002-03-08

http://www.ee.oulu.fi/~mcfrisk/linux/gprs/files/INSTALL.gprs 2002-03-08

http://gatling.ikk.sztaki.hu/~kissg/gsm/at+c.html 2002-03-08

Programmeringsreferenser

Osborne McGraw – Hill, C The complete reference, second edition, shildt, 1990

S. Monk, Timothy, Serial Communications, SAMS Publishing, 1993

Prata, Stephen, Avancerad C, Pagina Förlags AB, 1987

Kochan, G, Stephen och Wood, H, Patrik, UNIX – Shell programming, Hayden books, 1989

Negus, Christopher, Red Hat Linux 7 Bible, IDG books, 2000

A Förkortningar

A/D Analog -> digital omvandlare

AT Attention

BOL Box On-Line

COM Communications port

CSD Circuit Switched Data

CS Coding Scheme

D/A Digital -> Analog omvandlare

EDGE Enhanced Data rates for GSM Evolution 384kb/s

GPRS General packet radio service

GSM Global System for Mobile communications

HSCSD High Speed Circuit Switched Data

IrDa Infrared Data Association

LAN Local Area Network

NAT Network Address Translation

NMT Nordic Mobile Telephony

PDU Protocol Description Unit

PIC Programmable Interupt Controller

PSD Packet Switched Data

RTT Round Trip Time

UMTS Universal Mobile Telefon System

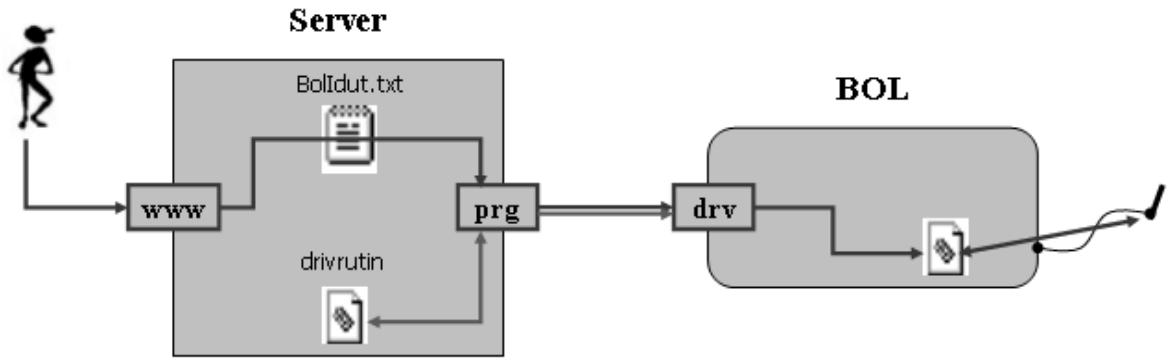
USB Universal Serial Bus

WWW World Wide Web

B Trafikfall

B.1 Uppdatering

Här visas kommunikationsgången då en användare valt att uppdatera en drivrutin.
En användare väljer drivrutin ur listan på webbsidan och trycker ”uppdatera”. Via BolIdut.txt når servleten BOL-servern. Denna hämtar drivrutinen och skickar över denna till BOL-enheten. BOL-enheten installerar drivrutinen mot vald port och startar den.



B.2 Hämta värde

Här visas kommunikationsgången då en användare vill läsa av sin enhet på BOLen. Användaren trycker på ”hämta” knappen på webbsidan. Via BolIdut.txt når servleten BOL-servern. Denna skickar en request begäran till aktuell BOL-enhet. BOL-enheten läser av innehållet i ”in-filen” för aktuell enhet.

Svaret skickas tillbaka till BOL-servern. Informationen lagras på filen ”in” för den aktuella porten. Samtidigt skriver BOL-servern på filen BolIdin.txt och når på det sättet servleten. Servleten vet nu att det finns ett svar och hämtar detta från filen ”in” i rätt katalog (katalogen motsvarar porten på BOLen som enheten är ansluten mot). Därefter presenteras informationen på ett lämpligt sätt för användaren.