NTNU
Det skapende universitet

Brukerdrevet backup-system for nomadiske PC-er

Tore Mauset

Master i datateknikk
Oppgaven levert: Juni 2006
Hovedveileder: Anders Christensen, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Oppgavetekst

På bakgrunn av eksisterende komponenter; designe og implementere et brukerdrevet (bruker initierer backup og restore) backup-system for brukere av bærbare PC-er ved IDI (studenter og ansatte), som har lav administrativ overhead,lav brukerterskel og kryptert overføring.

Oppgaven gitt: 20. januar 2006
Hovedveileder: Anders Christensen, IDI

Sammendrag

God og oppdatert backup er viktig for å unngå tapt arbeidsinnsats når uhellet er ute og data forsvinner eller korrupteres. Flere og flere benytter bærbare PC-er i jobben. Bærbare maskiner er mer utsatt for datatap i form av støtskader og tyveri enn stasjonære, samtidig som deres nomadiske natur gjør at de lett faller utenfor tradisjonelle backup-systemer.

En undersøkelse utført blant ansatte ved IDI viser at å tilby en brukervennlig løsning for backup av bærbare PC-er er ønsket og vil forbedre de ansattes backup-frekvens.

På grunnlag av denne undersøkelsen og ønsker fra Teknisk gruppe ved IDI blir eksisterende backup-teknologi vurdert og et brukerdrevet backup-system basert på verktøyet rdiff-backup designet og implementert.

rdiff-backup viser seg å ha problemer med at metadata lagret på serveren lett korrupteres ved uforutsette hendelser i backup-prosessen og at å kjøre det under Cygwin i Windows ikke er fullt funksjonelt og stabilt.

Til sist påpekes punkter som må forbedres i systemet før det eventuelt kan settes i drift. Det anbefales også å se på andre mulige løsninger, spesielt trekkes backup-systemet Box Backup frem som et alternativ å vurdere i senere arbeid.

Forord

Denne rapporten er resultatet av masteroppgaven min ved Norges teknisk-naturvitenskapelige universitet, institutt for datateknikk og informasjonsvitenskap (IDI).

Oppgaven har latt meg utforske nye og spennende teknologier og tilnærminger til backup-tjenester i dagens raskt endrende nettverksmiljøer.

Jeg vil rette en stor takk til min veileder ved instituttet Anders Christensen og Arne Dag Fidjestøl ved Teknisk gruppe, IDI for gode idéer og konstruktiv tilbakemelding underveis i prosjektet.

Trondheim, 23.06.2006

Tore Mauset

Innhold

Figurer

Tabeller

Del 1

Innledning

I denne delen beskrives kort bakgrunnen for prosjektet, hvorfor det gjennomføres og hvordan dette skjer. Rapporten videre består av disse delene:

- Del 2 presenterer metode og innfallsvinkel for oppgaven.
- Del 3 presenterer brukerundersøkelsen og diskuterer resultatet av denne.
- Del 4 inneholder kravspesifikasjonen for systemet.
- Del 5 presenterer og vurderer eksisterende teknologi opp mot kravspesifikasjonen.
- Del 6 tar for seg videre teknologivalg og design av systemet.
- Del 7 forklarer utvalgte deler av implementasjonen.
- Del 8 presenterer observasjoner av det ferdig systemet og resultater i forhold til kravspesifikasjonen.
- Del 9 oppsummerer arbeidet og trekker konklusjoner basert på resultatene.
- Del 10 presenterer tanker omkring videre arbeid innen feltet.

1.1 Bakgrunn

Det finnes mange og godt kjente grunner til hvorfor man skal ta backup av dataene sine[21]. Brukere kan ved en feiltagelse slette eller overskrive viktige data, lagringsmedia feiler, brann eller vannlekkasjer ødelegger datamaskiner og utenforstående endrer eller ødelegger data gjennom elektroniske innbrudd og virus. God og oppdatert backup er derfor essensielt for å unngå merarbeid og merkostnader som resultat av datatap. Ikke alle brukere tenker over disse farene og mangler derfor tilstrekkelig backup av dataene sine.

Flere og flere av brukerne ved IDI benytter bærbare PC-er som eneste eller primære arbeidsmaskin. Bærbare PC-er har høyere risiko for datatap enn stasjonære PC-er, da de lettere utsettes for støtskader og tyveri. Tradisjonelt har backup-systemer forventet å finne klientene på faste adresser til faste tidspunkt. Med nomadiske PC-er er ikke dette lenger mulig, selv med tilordning av faste adresser kan man ikke sikkert si når den enkelte maskin vil være tilkoblet nettet. Derfor må nye løsninger tas i bruk, der klienten styrer backup og restore. IDI har i dag ingen løsning for backup av nomadiske PC-er.

1.2 Mål

Målet med oppgaven er å utvikle en løsning for backup av bærbare PC-er for å redusere risikoen for datatap for brukere ved IDI.

Dette er formulert i følgende oppgavetekst:

> På bakgrunn av eksisterende komponenter; designe og implementere et brukerdrevet (bruker initierer backup og restore) backup-system for brukere av bærbare PC-er ved IDI (studenter og ansatte), som har lav administrativ overhead, lav brukerterskel og kryptert overføring.

1.3 Begrensninger

Tidsrammene for masteroppgaven begrenser hvor mye som kan implementeres. Det skal derfor kun utvikles en kommandolinjebasert prototype av klienten for å teste teknologien. Serveren skal være klar for å settes i produksjon.

Del 2

Metode

Det vil først gjennomføres en spørreundersøkelse blant de ansatte ved IDI, som aktuelle brukere av systemet. Denne undersøkelsen utformes med tanke på å kartlegge hvilke krav brukerne vil stille til et backup-system, spørsmålene og resultatene finnes i del 3.

Parallelt med at spørreundersøkelsen gjennomføres vil aktuell teknologi kartlegges. Kataloger for åpen programvare, som Freshmeat[1] og SourceForge[2], vil være gode kilder til programmer som bør vurderes. Informasjon om de forskjellige alternativene vil først og fremst samles fra de respektive programmenes hjemmesider på internett, men også uavhengige kilder kan trekkes inn.

Resultatet fra spørreundersøkelsen vil si noe om behovet for en sånn tjeneste og danne grunnlaget for kravspesifikasjonen. Denne vil utarbeides i samarbeid med Teknisk gruppe ved IDI. Kravspesifikasjonen angir rammene for og kravene til programvaren som skal utvikles.

Deretter vil den eksisterende teknologien vurderes opp mot punktene i kravspesifikasjonen og scenarioet programvaren skal benyttes i. Når en eksisterende løsning er valgt som grunnlag for det ferdige backup-systemet vil andre teknologi- og designvalg gjøres. Dette omfatter programmeringsspråk og eventuelle andre behov, som database og kommunikasjon.

Ut fra designet implementeres så systemet basert på de utvalgte teknologiene. Koden dokumenteres med Javadoc[16] og det skrives brukerveiledning til backup-systemet.

Avsluttende testes det ferdige systemet mot kravspesifikasjonen og oppgaveteksten. Denne testen danner grunnlag for resultatet og dermed også konklusjonen.

[1] http://freshmeat.net/
[2] http://sourceforge.net/

Ut fra resultatene og erfaringer underveis skrives til sist forslag til videre arbeid innen området.

Del 3

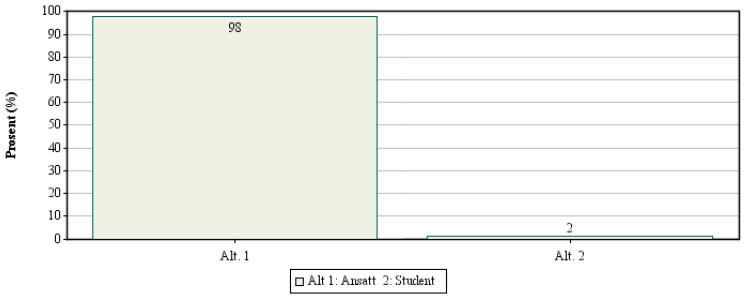
Brukerundersøkelse

For å ha et bedre grunnlag for videre design- og teknologivalg har det blitt gjennomført en brukerundersøkelse blant ansatte ved IDI. Undersøkelsen ble gjennomført med spørreskjema på web og var åpen en uke. På denne tiden svarte 62 personer på undersøkelsen, hvilket er i overkant av en tredjedel av de ansatte ved instituttet.

Under presenteres først spørsmålene fra undersøkelsen sammen med resultatene, deretter observasjoner på grunnlag av disse. Alle besvarelsene finnes i vedlegg B.

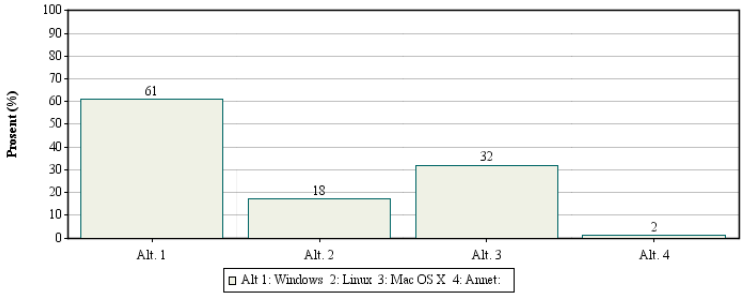
3.1 Spørsmål og resultater

1. **Hva er din tilknytning til IDI?**
    - Ansatt: 61 (98%)
    - Student: 1 (2%)



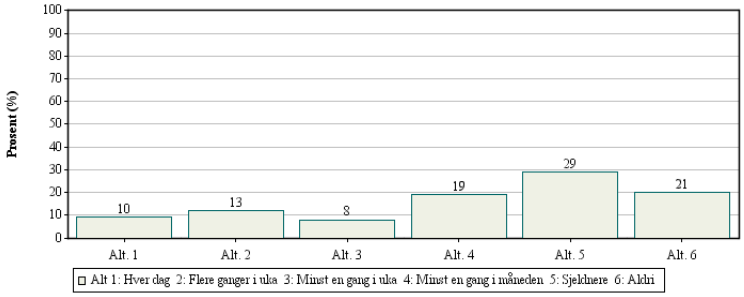
Invitasjon til undersøkelsen ble kun sendt ut til ansatte, men også studenter hadde mulighet til å svare på den.

2. **Hvilke(t) operativsystem bruker du på din bærbare PC?**
    - Windows: 38 (61%)
    - Linux: 11 (18%)
    - Mac OS X: 20 (32%)
    - Annet: 1 (2%)



Den totale summen av svar utgjør her mer enn 100%, da enkelte har flere operativsystem på sin bærbare maskin, eller flere maskiner.

3. **Hvor ofte tar du backup av data på din bærbare PC?**
    - Hver dag: 6 (10%)
    - Flere ganger i uka: 8 (13%)
    - Minst en gang i uka: 5 (8%)
    - Minst en gang i måneden: 12 (19%)
    - Sjeldnere: 18 (29%)
    - Aldri: 13 (21%)



4. **Hvis du ikke svarte “Aldri” på forrige spørsmål, hvordan tas denne backupen?**

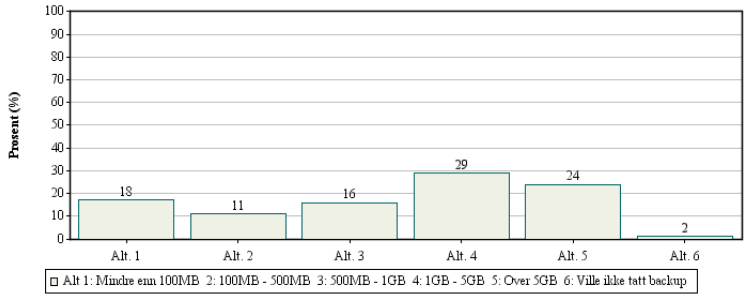
   Svarene deler seg i tre kategorier:

   - Versjonskontroll med CVS[27] eller SVN[26] mot IDIs server.
   - Mer eller mindre regelmessig kopiering av viktige data til ekstern harddisk, CD/DVD eller hjemmeområde ved IDI.
   - Tar ikke backup fordi alt arbeid foregår opp mot hjemmeområde ved IDI.

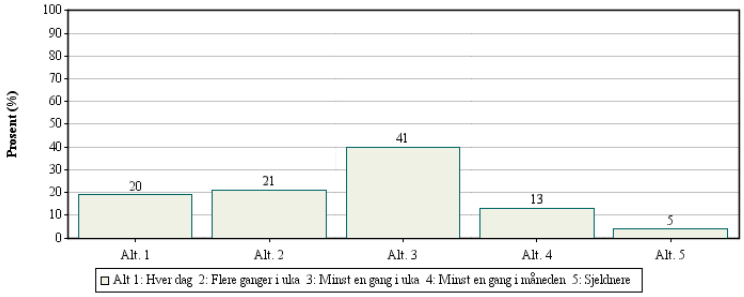
Spørsmålene herfra og ned gjelder din antatte bruk dersom IDI hadde tilbydd en tjeneste for backup av bærbare PC-er.

5. **Hvor mye data ville du tatt backup av fra din bærbare PC?**

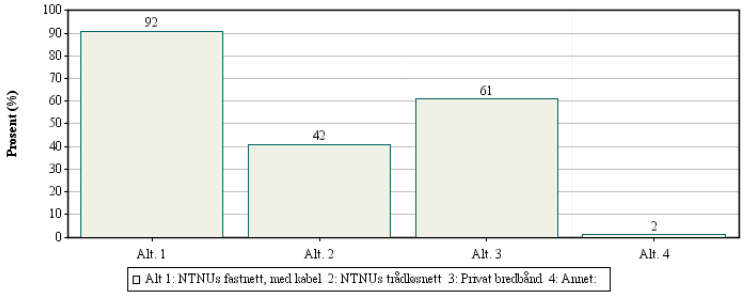
   - Mindre enn 100MB: 11 (18%)
   - 100MB - 500MB: 7 (11%)
   - 500MB - 1GB: 10 (16%)
   - 1GB - 5GB: 18 (29%)
   - Over 5GB: 15 (24%)
   - Ville ikke tatt backup: 1 (2%)



6. **Hvor ofte ville du tatt backup?**
    - Hver dag: 12 (20%)
    - Flere ganger i uka: 13 (21%)
    - Minst en gang i uka: 25 (41%)
    - Minst en gang i måneden: 8 (13%)
    - Sjeldnere: 3 (5%)



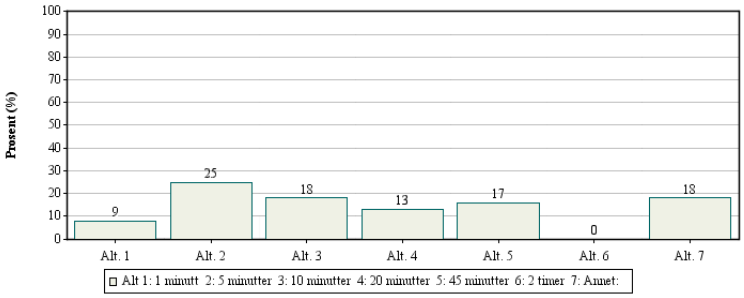
7. **Hva slags internettilknytning ville du benyttet? (Du kan velge flere)**
    - NTNUs fastnett, med kabel: 57 (92%)
    - NTNUs trådløsnett: 26 (42%)
    - Privat bredbånd: 38 (61%)
    - Annet: 1 (2%)



Den totale summen av svar utgjør her mer enn 100%, da en bærbar maskin kan tas med og brukes på flere forskjellige steder.

8. **Hvor lang tid mener du det ville være akseptabelt å vente mens en backup kjører?**

    - 1 minutt: 5 (9%)
    - 5 minutter: 15 (25%)
    - 10 minutter: 11 (18%)
    - 20 minutter: 8 (13%)
    - 45 minutter: 10 (17%)
    - 2 timer: 0 (0%)
    - Annet: 11 (18%)

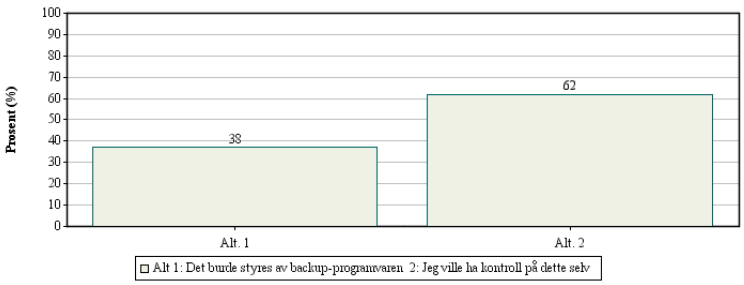


Svarene under "Annet" havner i to kategorier:

- Backup tar den tiden det tar, avhengig av mengden data som skal overføres.
- Tidsbruk er uvesentlig dersom backup kan foregå om natten.

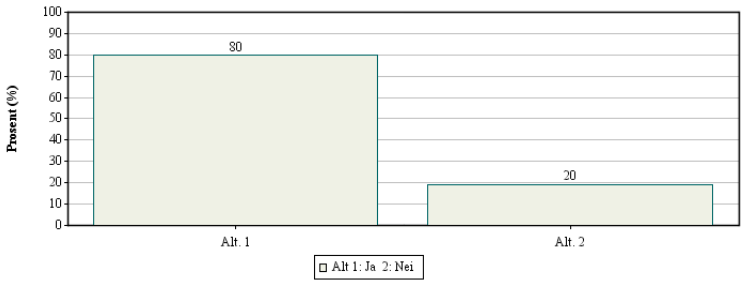
9. **Burde backup utføres på faste tidspunkt, ved tilkobling til NTNUs nett e.l. uten at du selv måtte gjøre noe, eller ville du ønsket å kontrollere dette selv?**

    - Det burde styres av backup-programvaren: 23 (38%)
    - Jeg ville ha kontroll på dette selv: 38 (62%)



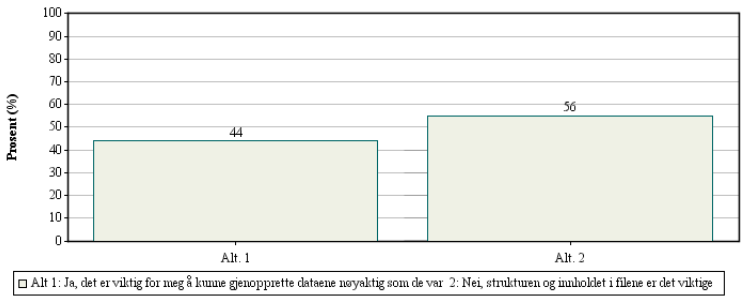
10. **Ville du benyttet tjenesten dersom dataene dine ble lagret ukryptert på backup-server, og administrator dermed i teorien hadde tilgang til dem?**

    - Ja: 49 (80%)
    - Nei: 12 (20%)



11. **Ville det vært viktig for deg at det ble tatt backup av filrettigheter?**

    - Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var: 27 (44%)
    - Nei, strukturen og innholdet i filene er det viktige: 34 (56%)



12. **Eventuelle kommentarer**
    Hovedpunktene som kommer frem her er:

    - God konfigurerbarhet av hva det skal tas backup av og når dette skal gjøres er viktig.
    - Systemet bør benytte inkrementell backup.
    - “Godt tiltak, dette er et tilbud jeg har savnet.”
    - “Jeg klarer å styre backup selv og ser ikke behovet for en sånn tjeneste.”

3.2 Observasjoner

Det mest utbredte operativsystemet er Windows, men undersøkelsen viser at tilsammen 52% av de spurte også benytter andre operativsystem. De tre som utmerker seg og dermed må støttes av programvaren er Windows, Mac OS X og Linux.

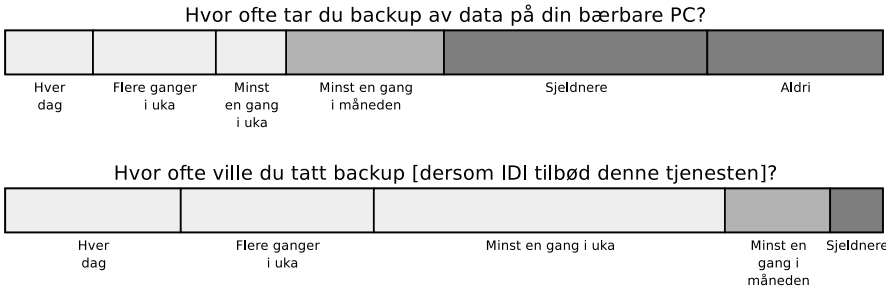
Mengden data de spurte ønsker å ta backup av varierer, men over halvparten, 52%, svarer at de ønsker å ta backup av mer enn 1GB data. Med så store datamengder bør den initielle backupen utføres på fastnett tilknyttet NTNU, noe 92% svarer at de kan tenke seg å bruke for å ta backup.

80% av de spurte ville benyttet tjenesten selv om dataene ikke ble lagret kryptert på serveren. Det kan spekuleres i at en del av de resterende 20% likevel vil benytte en tjeneste uten kryptert lagring dersom det forklares at dataene vil være like sikre som om de ble lagret på hjemmeområdet ved IDI.

Til spørsmålet “Ville det vært viktig for deg at det ble tatt backup av filrettigheter?” svarer 44% “Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var”. Dette er en høyere andel enn forventet, da de fleste kun har én bruker på sin bærbare maskin og derfor ikke har behov for å kontrollere aksess-kontroll-lister (eng. access control list, ACL). I etterkant antas svaralternativene å ha vært noe ledende, da de fleste naturlig nok ønsker å få tilbake dataene sin “nøyaktig som de var”.

Av de som tar backup hver dag eller flere ganger i uke benytter de fleste et versjonskontrollsystem mot IDIs server. Av de som tar backup minst en gang i uka eller sjeldnere kopierer de fleste data de anser som viktige til ekstern harddisk, CD/DVD eller hjemmeområde ved IDI, mens noen av de som aldri tar backup oppgir at de alltid jobber opp mot hjemmeområdet.

Nesten to tredjedeler av de spurte ønsker å kontrollere når backup skal foregå selv, den resterende tredjedelen mener dette kan være opp til programvaren.



Figur 3.1: Teoretisk endring i backup-frekvens ved innføring av backup-system.

Hvor ofte man bør ta backup er avhengig av hvor mye arbeid som utføres på PC-en og hvor ofte dataene endres. Her klassifiseres det at det er ofte nok å ta backup minst en gang i uka og for sjelden å ta backup sjeldnere enn en gang i måneden. De som svarer at de tar backup minst en gang i måneden faller imellom disse to gruppene, nylig arbeid kan gå tapt, men eldre data vil finnes på backup-serveren. Med disse grensene svarer 31% at de allerede tar backup ofte nok, mens 50% svarer at de tar backup for sjelden. Videre svarer 82% at de ville tatt backup ofte nok hvis de kunne benytte seg av en backup-tjeneste ved IDI, mot bare 5% som fortsatt vil ta backup for sjelden, se figur 3.1.

Når en ser på enkeltbesvarelser svarer 42 av 62 at de kommer til å ta backup oftere med en slik tjeneste enn de gjør i dag. Dette er økning blant 68% av de spurte.

Disse endringene er en god indikasjon på at tjenesten er ønsket og at den vil benyttes og dermed minske risikoen for stort tap av arbeidsinnsats som følge av korrupsjon eller tap av data på bærbare maskiner.

Del 4

Kravspesifikasjon

Denne kravspesifikasjonen er utarbeidet i samarbeid med Teknisk gruppe med IDI, NTNU med grunnlag i brukerundersøkelsen beskrevet i del 3. Kravspesifikasjonen angir rammer for og krav til programvaren som skal utvikles i dette prosjektet.

4.1 Omfang

Systemet skal:
Krav 1. *inneholde serverprogramvare for en sentral backup-server.*
Krav 2. *inneholde prototype for klientprogramvare for Windows, Linux og Mac OS X som lar brukeren koble til serveren for å sikkerhetskopiere og gjenopprette data.*

4.2 Autentisering

Systemet skal:
Krav 3. *autentisere tilkoblinger til backup-tjenesten med kryptert innlogging.*
Krav 4. *kun gi hver enkelt bruker oversikt over og tilgang til sine egne filer.*

4.3 Backup og restore

Systemet skal:
Krav 5. *kryptere alle data som overføres mellom klient og server.*
Krav 6. *tilby brukeren backup og restore av enkeltfiler.*
Krav 7. *tilby brukeren backup og restore av hele katalogstrukturer.*
Krav 8. *la brukeren starte backup manuelt.*

Krav 9. *kunne starte backup automatisk på brukerspesifiserte tidspunkt.*

Systemet bør:
Krav 10. *forsøke å minimalisere mengden overførte data.*

4.4 Lagring

System bør:
Krav 11. *kryptere alle sikkerhetskopierte data som lagres på serveren for å sikre disse ved en eventuell kompromittering av servermaskinen.*
Krav 12. *lagre data på serveren på en måte som begrenser den nødvendige lagringsplassen.*

4.5 Dokumentasjon

Systemet skal:
Krav 13. *leveres med brukerveiledning og kodedokumentasjon.*

Del 5

Eksisterende teknologi

I denne delen presenteres og evalueres eksisterende løsninger som kan benyttes som grunnlag for videre arbeid.

5.1 Amanda

Amanda[4] utfører serverinitiert backup av flere klientmaskiner til en sentral server og videre ut på tape eller optisk media. Dette forutsetter at alle klientene kan nås på faste adresser.

Backup-prosessen styres av den sentrale serveren og foregår ved hjelp av dump[12] eller GNU tar[28] for *nix-klienter og Samba[25] for Windows-klienter. Backup over Samba forutsetter at klientene deler alle data som skal sikkerhetskopieres med full lesetilgang og kan ikke ta backup av åpne filer i Windows.

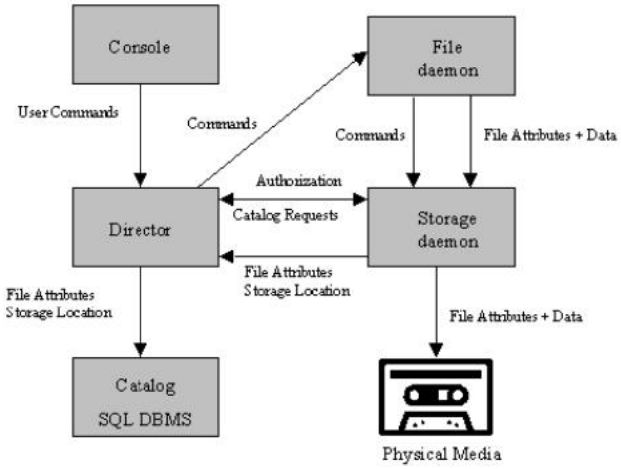
Amanda tilbyr ingen brukerstyrt gjenoppretting av data, kun root/administrator har mulighet til dette.

Systemet støtter komprimering som kan det skje på klient eller server. Kerberos[17] støttes for autentisering for klienter og kryptering av sikkerhetskopierte data.

5.2 Bacula

Bacula[9] utfører serverinitiert backup av flere klientmaskiner til en sentral server og videre ut på tape, harddisk eller optisk media. Dette forutsetter at alle klientene kan nås på faste adresser.

Bacula styres av en sentral *Director daemon*. På hver maskin det skal tas backup fra kjører en *Filer daemon* som sender data til en sentral *Storage daemon*.



Figur 5.1: Interaksjon mellom komponenter i Bacula. Illustrasjon fra brukermanualen.

Overføringen kan kan autentiseres og krypteres med TLS. Sikkerhetskopierte data indekseres i en *Catalog database*. Disse komponentene kan kjøre på samme maskin eller spredd ut på flere maskiner. Prosessen kan styres og overvåkes fra en konsoll som kobler seg til Director daemon. Bacula benytter Volume Shadow Copy Service[29] i Windows XP og Windows 2003 for å ta backup av låste filer.

Gjenoppretting av data krever tilgang til Baculas konsoll, noe som også gir tilgang til alle andre sikkerhetskopierte data.

5.3 BackupPC

BackupPC[8] utfører serverinitiert backup av flere klientmaskiner til harddisk på en sentral server. Dette forutsetter i utgangspunktet at alle klientene kan nås på faste adresser, men BackupPC kan også forsøke å finne klienter basert på NetBios-navn.

For Windows-klienter kan BackupPC benytte Samba[25] eller rsync. Bruk av rsync krever at også Cygwin[11] er installert på klientmaskinen. Backup over Samba forutsetter at klientene deler alle data som skal sikkerhetskopieres med full lesetilgang. For Linux-klienter benyttes rsync eller GNU tar[28].

BackupPC støtter komprimering av data og smart sammenslåing av filer på serveren. Dette innebærer at en fil som sikkerhetskopieres av flere brukere kun lagres én gang på serveren.

BackupPC tilbyr brukerne et webgrensesnitt for gjenoppretting av filer. Filene kan

lastes ned i et komprimert arkiv eller lastes direkte opp til klienten over Samba. Gjenoppretting over Samba forutsetter at BackupPC har full skrivetilgang til den aktuelle nettverksstasjonen.

5.4 Box Backup

Box Backup[6] benytter en klient som med tilfeldige intervall våkner og sjekker etter endringer blant filene den skal ta backup av. Backup kan også initieres manuelt av klienten.

Klient autentiseres med sertifikat signert av serveren og dataene krypteres med en privat nøkkel. Dette innebærer at det må finnes en separat kopi av denne nøkkelen utenfor klientmaskinen da dataene ikke kan gjenopprettes uten den. Man kan dessverre ikke gå ut fra at alle brukere faktisk vil ha en sånn kopi tilgjengelig og man løper derfor en risiko ved en slik løsning.

I likhet med rsync-baserte løsninger sender Box Backup kun de faktiske endringene i filene til serveren, se 5.9. Metainformasjon som ACLer og Mac OS X resource forks lagres.

5.5 Duplicity

Duplicity[13] utfører i likhet med rdiff-backup (se 5.8 klientinitiert backup basert på rsync-algoritmen[3] og støtter flere forskjellige overføringsprotokoller som ssh, ftp og rsync.

Backup foregår ved at at endringene pakkes med GNU tar[28] og krypteres med GnuPG før de overføres til serveren. Denne krypteringen fordrer at man har en separat kopi av den private nøkkelen for å kunne gjenopprette data dersom den lokale kopiene går tapt. Dette er en risikomoment, da sannsynligheten er stor for at ikke alle brukerne vil følge opp dette.

5.6 NasBackup

NasBackup[19] utfører klientinitiert backup av Windows-klienter til harddisk på en sentral server.

Backup initieres fra et eget GUI, den underliggende transporten foregår med rsync, se 5.9. Gjenoppretting foregår over Samba, FTP eller HTTP med vanlige verktøy for disse protokollene.

5.7 P2PBackup

P2PBackup benytter Peer-to-peer-teknologi for å ta backup. Klientene kobler til en sentral server som distribuerer dataene. Dersom klienten gjør 100GB tilgjengelig som lagringsplass for andre kan du bruke tilsammen 100GB på de andre klientene. Brukeren bestemmer selv hvordan denne plassen skal brukes. Data kan lagres mer enn en gang i nettverket, så 100GB kan f.eks. brukes til å lagre fem kopier av 20GB. Bakgrunnen for dette er å øke sannsynligheten for at alle de sikkerhetskopierte dataene er tilgjengelige selv om ikke alle klientene er koblet til serveren.

5.8 rdiff-backup

rdiff-backup[7] er likt Duplicity (se 5.5), men lager en fullstendig ukryptert kopi av dataene og filstrukturen på serveren i tillegg til at differansen til eldre versjoner lagres. Programmet bruker rsync-algoritmen[3] for å minimere mengden overført og lagret data.

rdiff-backup lagrer også metainformasjon som ACLer og Mac OS X resource forks selv om filsystemet det tas backup til ikke støtter dette.

5.9 rsync

rsync[23] er et verktøy for å synkronisere filer over nett og har gitt navn til rsync-algoritmen[3].

Denne algoritmen overfører kun endringer i hver enkelt fil i stedet for hele filen hver gang den er endret. Dette gjøres ved at serveren deler filen i blokker og generer to sjekksummer for hver blokk; en svak 32-bit “rullende” sjekksum (eng. “rolling” checksum) og en sterk 128-bit MD4 sjekksum. Sjekksummene overføres til klienten, som søker etter identiske blokker i sin versjon av filen ved hjelp av den svake sjekksummen. Klienten sender så instruksjoner til serveren for hvordan den skal bygge en kopi av filen som ligger på klienten. Disse instruksjonene består av referanser til blokker den allerede har, og dataene i de blokkene den ikke har.

Ettersom rsync er utviklet for å synkronisere filer og kataloger lagrer det ingen versjonsdata på serveren. Kun siste versjon det ble tatt backup av vil være tilgjengelig.

Det eksisterer en implementasjon av rsync skrevet i Python[22], rsync.py[24], som muliggjør bruk av rsync på Windows uten Cygwin. Denne versjonen støtter derimot ikke funksjonene sjekksummer, rettigheter, eierskap, grupper, operasjoner

på tvers av filsystemer, ssh, autentisering og kryptering som alle finnes i den opprinnelige implementasjonen. rsync.py utelukkes derfor fra videre vurdering.

5.10 Capivara

Capivara[10] er ingen backupløsning, men en filbehandler med støtte for synkronisering mot en annen maskin over nettverk.

Programmet er skrevet i Java og dermed uavhengig av operativsystem. Synkroniseringen støtter kryptert SFTP-overføring av filer. Det kan derfor være interessant å benytte komponenter fra Capivara i en OS-uavhengig klient, dersom SFTP velges som overføringsprotokoll.

5.11 Sammenligning

I tabell 5.1 sammenlignes egenskaper ved de eksisterende systemene.

	Amanda	Bacula	BackupPC	Box Backup	Duplicity	NasBackup	P2PBackup	rdiff-backup	rsync
Klientinitiert backup				x	x	x	x	x	x
Klientinitiert gjenoppretting			x	x	x	x	x	x	x
Inkrementell backup	x	x	x	x	x	x		x	
Kryptert overføring fra alle klient-OS		x	x[1]	x	x[1]		x	x[1]	x[1]
Overfører og lagrer kun endringer				x	x	x		x	x
Kryptering av data på klientsiden	x			x	x	x	x		
Komprimert lagring	x	x	x			x			
"Pooling" av identiske filer			x						
Støtter Windows-klienter	x[2]	x	x[1,2]	x[1,3]	x[1]	x[1]	x	x[1]	x[1]
Støtter VSS[4]		x							
Støtter Linux-klienter	x	x	x	x	x			x	x
Støtter Mac OS X-klienter	x	x	x	x	x			x	x
Støtter Mac OS X resource forks	x[5]		x[5]	x				x	
Støtter Extended Attributes				x				x	

Tabell 5.1: Sammenligning av eksisterende backup-løsninger.

[1] Med Cygwin.
[2] Kan hente data vha Samba.
[3] En klient for Windows finnes, men er uferdig.
[4] Volume Shadow Copy Service. Lar Windows ta backup av låste filer, se [29].
[5] Med Xtar[30].

5.12 Evaluering

Amanda og Bacula er utviklet for å brukes i et nettverksmiljø der administrator har kontroll på alle maskinene og kan styre når de skal tas backup av og gjenopprettes. Begge sender backup over nettverk og ut på tape. Selv om de er gamle og velprøvde løsninger vil de ikke egne seg for en klientdrevet løsning.

BackupPC, Box Backup, NasBackup og P2PBackup er fra den tiden der harddisker har blitt så billige at de kan erstatte tapene. BackupPC er også lagd med den forutsetning at administrator har oversikt over alle maskinene og styrer når det skal

tas backup av dem. Dette gjør den i utgangspunktet til en uaktuell løsning, men om transportmekanismen endres til å være klientdrevet. De tre andre er klientinitierte løsninger der brukeren har kontroll på både backup og gjenoppretting. NasBackup og P2PBackup er løsninger kun for Windows-klienter og dekker derfor ikke krav 2 presentert i 4.

rsync, rdiff-backup og Duplicity baserer seg alle på rsync-algoritmen[3] som kun overfører endringer innad i hver enkelt fil istedet for hele filen. De er utviklet for *nix-plattformer, men kan benyttes i Windows sammen med Cygwin. Ingen av de er komplette backup-løsninger, men teknologien kan danne grunnlaget for et backup-system. rsync mangler støtte for å hente ut gamle versjoner av filer og er derfor ikke noe godt alternativ.

Backup over Samba tar ikke vare på alle filattributter i Windows, som aksess-kontroll-lister. Å måtte ha data delt med full lesetilgang (og skrivetilgang for direkte gjenoppretting med BackupPC) er en sikkerhetsrisiko da Samba-protokollen kun støtter kryptert autentisering, ikke kryptert overføring. Det må derfor benyttes en annen overføringsprotokoll.

De fire alternativene som gjenstår er da BackupPC, Box Backup, Duplicity og rdiff-backup. BackupPC må skrives om en del for å kunne bruke i et klientdrevet backup-system, overfører ikke bare endringene i hver enkelt fil og støtter ikke Extended Attributes i filsystemet. Til tross for mekanismen for “pooling” av like filer på serveren kommer BackupPC derfor til kort.

Box Backup og Duplicity baserer seg begge på en privat nøkkel for kryptering. Dette er en risiko, da tap av denne denne nøkkelen medfører at alle sikkerhets-kopierte data blir ubrukelige. Faren for at ikke alle brukere vil passe på å ha en egen kopi av nøkkelen anses som stor. Løsningene vil derfor ikke benyttes her.

rdiff-backup baserer seg på klientinitiert backup og restore, støtter kryptert overføring over ssh/scp, støtter de aktuelle operativsystemene[1] og backup av de fleste filattributter uavhengig av filsystemet på serveren. Det vil derfor brukes som grunnlag for backup-systemet som skal utvikles.

[1] Med Cygwin.

Del 6

Systemet

I denne delen presenteres først teknologivalg gjort for systemet. Deretter design av systemet og dets komponenter. Til slutt forklares implementasjonen av hver enkelt komponent.

6.1 Teknologivalg

Noe som er med å definere rammene for det videre designet er teknologien som skal benyttes i systemet. rdiff-backup er allerede valgt som underliggende teknologi, her presenteres kort andre nøkkelkomponenter og hvorfor de er valgt.

6.1.1 Programmeringsspråk

Klienten skal kunne kjøre på Windows, Linux og Mac OS X. Det finnes flere programmeringsspråk dette kan løses i, som Java[2], Perl[20] og Python[22]. Av disse er Java det klart mest utbredte, spesielt på Windows-plattformen. Klienten vil derfor utvikles i Java 2 SE 5.0. For serveren gjelder ikke de samme kravene til plattformer den skal kjøre på, og kommunikasjonen mellom klient og server (se 6.1.2) er uavhengig av programmeringsspråk. Det er derfor nokså fritt hva denne implementeres i, men Java 2 SE 5.0 velges også til den for å holde systemets kode mest mulig ryddig og konsekvent.

Java er et høynivå, objektorientert og portabelt programmeringsspråk[14]. Java-kode kompileres til byte-kode (eng. bytecode), som er maskinspråket til Java Virtual Machine (JVM). JVM-en er skrevet spesielt for den plattformen den kjører på, dermed er den kompilerte Java-koden plattformuavhengig[1].

[1] Med unntak av enkelte operativsystemspesifikke kall.

6.1.2 Kommunikasjon

For kommunikasjon mellom klient og server vil systemet benytte Apache XML-RPC 2.0[1], en implementasjon av Extensible Markup Language over Remote Procedure Call [15].

Ved å utføre XML-RPC-kall mot en XML-RPC-server som kjører på backup-serveren returneres alle data som svar på kall fra klienten. Systemet omgår dermed problemer forbundet med NAT[2] og VPN[3]. XML-RPC-kall foregår over http, og XML-RPC-implementasjonen støtter SSL for sikker kommunikasjon.

En ulempe med XML-RPC er at det ikke opprettes en permanent kommunikasjons-kanal mellom klient og server. Alle meldinger må derfor autentiseres separat.

6.1.3 Database

Systemet benytter en enkel database (se vedlegg C.2.4) for lagring av informasjon om brukernes filer. Serveren implementeres for å koble til en MySQL-database[18], men kan lett modifiseres til å benytte en annen databasemotor som støttes av Java.

6.2 Design

Overordnet består systemet av klienten *BackupClient* (omtales videre i denne delen som "klienten"), rdiff-backup på klienten, klientens filsystem, serveren *BackupServer* (omtales videre i denne delen som "serveren"), rdiff-backup på serveren, serverens filsystem og serverens database, se figur 6.1. Det skraverte området markerer hvilke komponenter som skal utvikles i dette prosjektet.

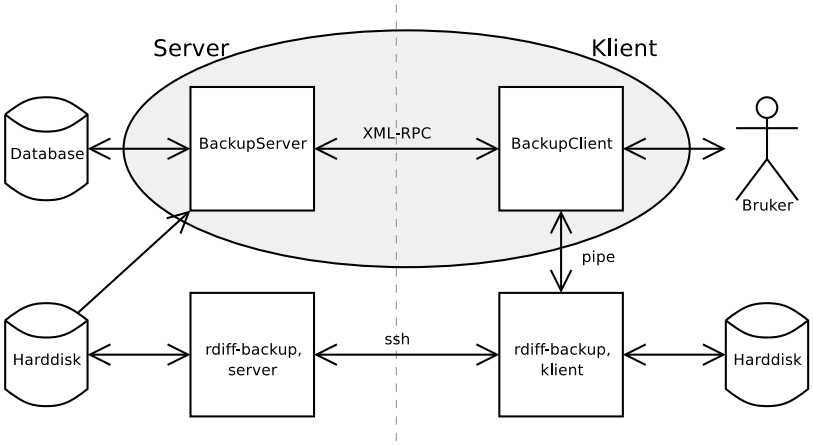
6.2.1 Kommunikasjon

Her presenteres XML-RPC-kall fra klienten til serveren.

Autentisering og Last Seen

Hver gang klienten skal utføre et XML-RPC-kall på serveren genererer den en *Response* fra en lagret *Next Challenge*. *Response* og *User* sendes som autentisering sammen med eventuelle andre parametere. Serveren sjekker først om autentisering-en er korrekt. Hvis den er det utføres operasjoner på serveren i henhold til kallet og klientens Last Seen-oppføring på serveren oppdateres. Så genererer serveren en

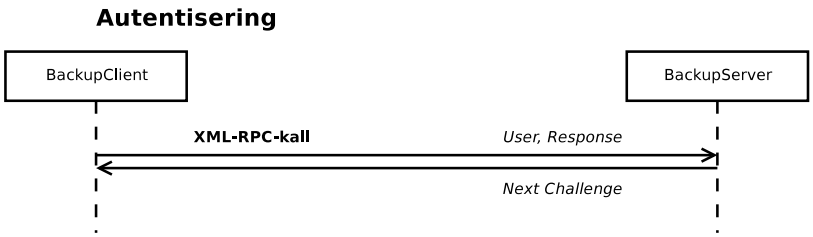
[2] Network Address Translation
[3] Virtual Private Network



Figur 6.1: Overordnet systemdesign.

Next Challenge som returneres til klienten sammen med eventuelle andre resultater. *Next Challenge* lagres på klienten til neste kall skal utføres. Se figur 6.2

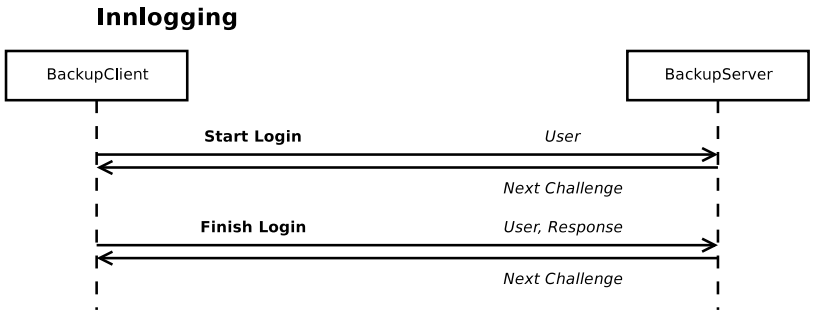
Disse operasjonene regnes som implisitt for alle kall fra klienten og utelukkes derfor fra beskrivelsen av de enkelte kallene, med unntak av Login og Logout.



Figur 6.2: Autentisering av XML-RPC-kall.

Login

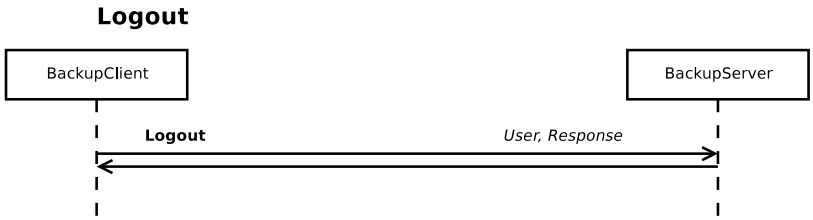
Ved innlogging oppretter klienten først en XML-RPC-forbindelse til serveren. Den kaller så **Start Login** på serveren med *User*. Serveren returnerer en *Next Challenge* til klienten. Klienten genererer en *Response* ut fra *Next Challenge* og kaller **Finish Login** på serveren med *User* og *Response*. Hvis *Response* er korrekt registreres brukeren som innlogget og en ny *Next Challenge* returneres til klienten. Se figur 6.3.



Figur 6.3: Innloggingssekvens.

Logout

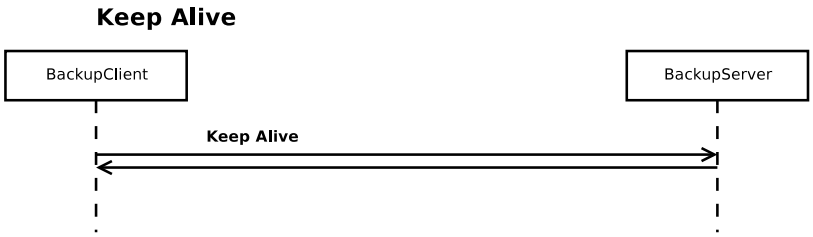
Ved utlogging kaller klienten **Logout** på serveren med parameterne *User* og *Response*, se figur 6.4. Dette kallet returnerer ingenting til klienten.



Figur 6.4: Utloggingssekvens.

Keep Alive

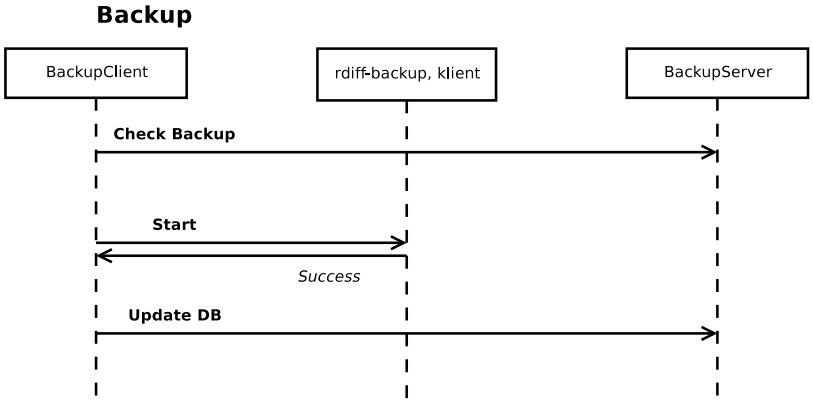
Med faste intervall sender klienten et **Keep Alive**-kall til serveren, se figur 6.5. Dette oppdaterer klientens innslag i Last Seen-listen på serveren.



Figur 6.5: Keep Alive-kall fra klient til server

Backup

Ved oppstart av backup kaller først klienten **Check Backup** (se 6.2.2) på serveren for å sikre at ikke backupen på serveren er korrupt, se figur 6.6. Deretter starter klienten sin lokale rdiff-backup, rapporterer fremdriften fra denne til brukergrensesnittet og venter på at rdiff-backup skal avslutte. Dersom backupen var vellykket kaller klienten så **Update DB** (se 6.2.2) på serveren.



Figur 6.6: Kall fra klient til server ved gjennomføring av backup.

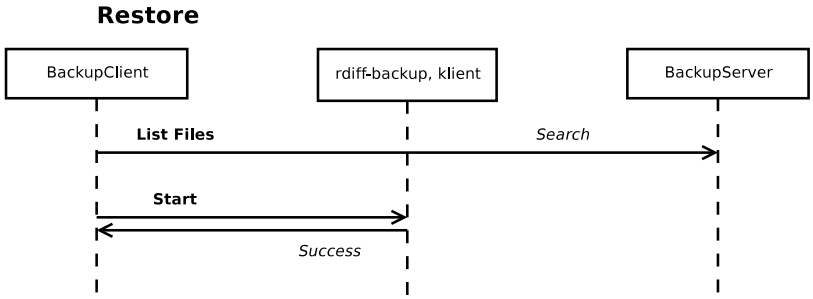
List Files

Når brukeren ber om en liste over tilgjengelige filer og versjoner av disse kaller klienten **List Files** på serveren med eventuell søkestreng *Search*. Serveren returnerer en liste *Result* med filnavn og versjoner av disse. Se figur 6.7.

Figur 6.7: Kall fra klient til server ved henting av backup-database.

Restore

Ved restore kaller klienten først **List Files** på serveren med *Search* lik filen eller katalogen som skal gjenopprettes. Dette gjøres for å sikre at en bruker som bruker kommandolinjegrensesnittet ikke har forespurt et mål det ikke finnes backup av. Dersom målet eksisterer på serveren starter klienten sin lokale rdiff-backup og rapporterer fremdriften til brukergrensesnittet. Se figur 6.8.



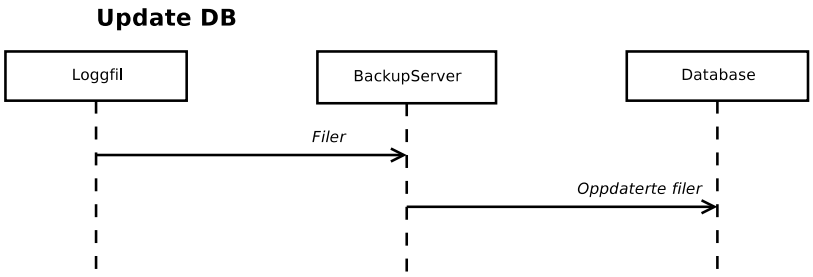
Figur 6.8: Kall fra klient til server ved gjennomføring av restore.

6.2.2 Server

Her presenteres kort prosesser internt på serveren.

Update DB

I prosessen **Update DB** analyserer serveren loggen fra rdiff-backup og legger informasjon om oppdaterte filer inn i databasen, se figur 6.9



Figur 6.9: Oppdatering av databasen.

Check Backup

Med faste intervall kjører serveren prosessen **Check Backup** for hver registrerte bruker av backup-systemet. Denne prosessen sjekker om metadataene i backupene har blitt korrupte (som følge av feilet backup) og forsøker å korrigere dette.

Check Connections

Hvert minutt kjører serveren prosessen **Check Connections**. Denne sjekker alle innslagene i Last Seen-listen på serveren. Klienter som ikke har kontaktet serveren i løpet av de siste tre minuttene antas frakoblet og logges ut fra serveren.

Del 7

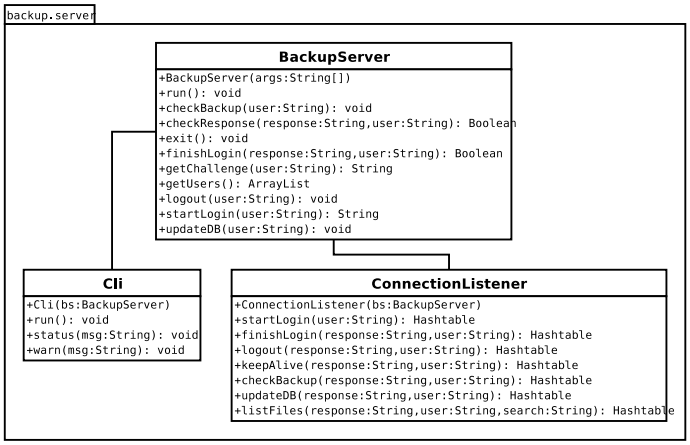
Implementasjon

Her presenteres og forklares utvalgte deler av implementasjonen.

7.1 Server

Figur 7.1 viser klassediagram for pakken backup.server. Pakken består av disse klassene:

- **BackupServer** – Hovedprogrammet.
- **Cli** – Kommandolinjebasert brukergrensesnitt.
- **ConnectionListener** – XML-RPC-lytter.



Figur 7.1: Klassediagram for server.

7.1.1 BackupServer

Ved oppstart av serveren startes XML-RPC-serveren med tilhørende Connection-Listener,

```
91      server = new WebServer(port);
        server.addHandler("BackupServer", new ConnectionListener(this));
        server.start();
```

og driver for databasetilgangen lastes.

```
101     try {
            Class.forName("com.mysql.jdbc.Driver");
            dbUrl = "jdbc:mysql://" + dbHost + ":" + dbPort + "/" + dbName;
        }
```

En separat tråd kontrollerer hendelser som forekommer med faste intervall:

```
125 public void run() {
        .
        .
        while (running) {
            Thread.sleep(1000);
            .
            .
                mins++;

                // Check for lost connections every minutes
                checkConnections();

                // Check for aborted backups
                if (mins \% 30 == 0) {
                    checkBackups();
```

Dersom det blir behov for flere faste hendelser kan de lett legges til i samme tråd.

Serveren har en metode for å generere en ny *challenge* og lagre korrekt response til senere sjekk. *challenge* er en tilfeldig streng med tilfeldig lengde. Korrekt *response* er en SHA-1-hash av passordhashen til brukeren konkatenert med *challenge*.

```
173 public String getChallenge(String user) {
            // Generate random seed for password challenge
            String challenge = Long.toString(Math.abs(r.nextLong()), 36);

            String tmp = getPWHash(user) + challenge;
            currentChallenges.put(user, hash(tmp));

            return challenge;
        }
```

Metoden som sjekker om en *response* er korrekt oppdaterer også brukererens innslag i serverens *lastSeen*-liste med tidspunktet meldingen ble mottatt.

```
190  public Boolean checkResponse(String response, String user) {
         .
         if (response.equals(currentChallenges.get(user))) {
             .
                 lastSeen.put(user, Calendar.getInstance());
                 .
                 return true;
```

Formatet på hver linje i loggfilen som rdiff-backup genererer er:

```
Filename Changed SourceSize MirrorSize IncrementSize
```

Dette medfører et problem for analyseringen, da filnavn kan inneholde mellomrom. Løsningen er å analysere felt for felt bakfra på hver linje:

```
393      while ((line = in.readLine()) != null) {
             int tmp;
             tmp = line.lastIndexOf(" ") + 1;
             String incrementSize = line.substring(tmp, line.length());
             line = line.substring(0, tmp - 1);
             tmp = line.lastIndexOf(" ") + 1;
             String mirrorSize = line.substring(tmp, line.length());.
             .
             .
```

Dersom *Changed* er *1* og *SourceSize* ikke er *NA* betyr det at linjen representerer en fil/katalog som har er ny eller endret og denne legges derfor inn i databasen:

```
409          if (changed.equals("1") && !sourceSize.equals("NA")
                     && !filename.equals(".")) {
                 insertDB(root + filename, user, time);
```

7.1.2 ConnectionListener

Alle metoder i XML-RPC-handleren er public og dermed tilgjengelig for klienten. Med unntak av *startLogin* som starter en ny login-prosess tar alle metodene inn brukernavn og respons på forrige challenge. Hvis autentiseringer er korrekt utføres tilhørende operasjoner på serveren og en Hashtable med ny challenge og eventuelle resultater returneres til klienten:

```
 93  public Hashtable checkBackup(String response, String user) {
         Hashtable<String, String> result = new Hashtable<String, String>();
         if (bs.checkResponse(response, user)) {
             bs.checkBackup(user);
             String challenge = bs.getChallenge(user);
             result.put("challenge", challenge);
         }
         return result;
     }
```

7.1.3 Cli

Kommandolinjegrensesnittet starter en separat tråd som leser linjer fra tastaturet:

```
32   public void run() {
         try {
                 InputStream inputstream = System.in;
                 InputStreamReader inputstreamreader = new InputStreamReader(
                         inputstream);
                 BufferedReader bufferedreader = new BufferedReader(
                         inputstreamreader);
                 .
                 .
                 while (running) {
                     string = bufferedreader.readLine();
```

Cli har også metoder som lar serveren skrive statusmeldinger og advarsler til konsollet:

```
65   public void status(String msg) {
```

og

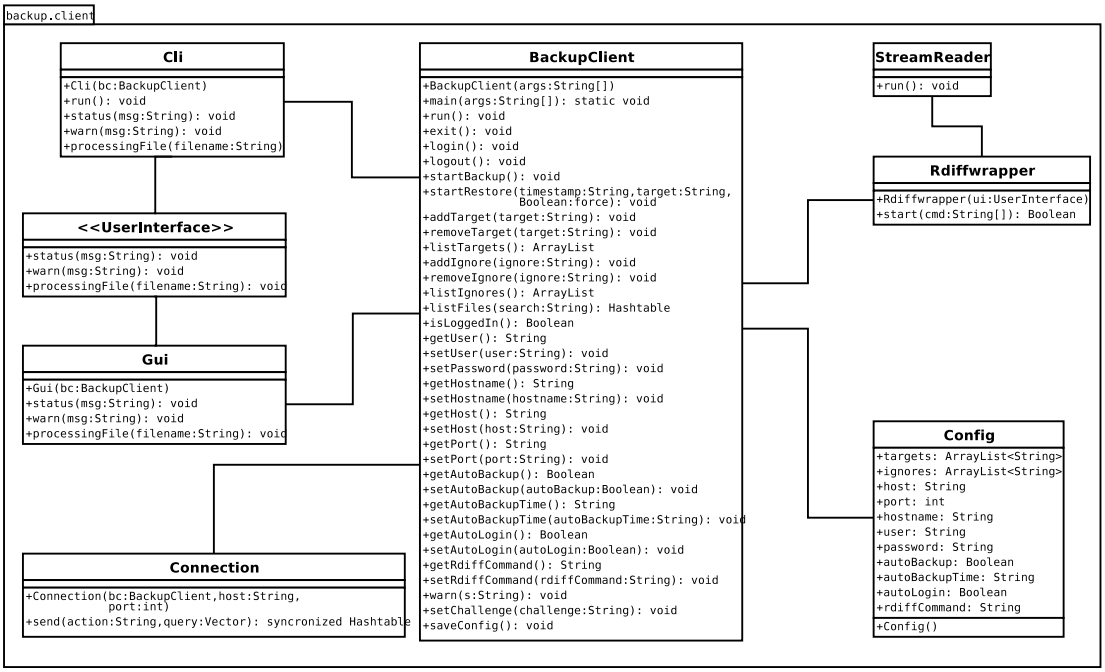
```
81   public void warn(String msg) {
```

7.2 Klient

Figur 7.2 viser klassediagram for pakken backup.client. Pakken består av disse klassene:

- **BackupClient** – Hovedprogrammet.
- **UserInterface** – Generelt interface som brukergrensesnittene må implementere.
- **Cli** – Kommandolinjebasert brukergrensesnitt.
- **Gui** – Grafisk brukergrensenitt.[1]
- **Config** – Konfigurasjonsobjekt for lagring til disk.
- **Connection** – XML-RPC-kobling mot serveren.
- **Rdiffwrapper** – Wrapper omkring rdiff-backup.

[1] Det grafiske brukergrensesnittet er ikke implementert i prototypen.



Figur 7.2: Klassediagram for klient.

7.2.1 BackupClient

Ved oppstart oppretter klienten et brukergrensesnitt i henhold til eventuelle kommandolinjeparametre:

```
48      if (args.length > 0 && args[0].equals("-g")) {
            ui = new Gui(this);
        }
        else {
            ui = new Cli(this);
        }
```

Klienten starter i likhet med serveren en separat tråd for å styre faste hendelser:

```
82  public void run() {
        .
        .
            while (running) {
                Thread.sleep(1000);
                    .
                    .
                        mins++;

                        // Send keep-alive signal to server every minute
```

```
if (isLoggedIn()) {
    keepAlive();
}

// Start scheduled backup
if (autoBackup && autoBackupTime.equals(now)) {
    startBackup();
```

Ved login oppretter klienten først en tilkobling til serveren,

```
142     conn = new Connection(this, host, port);
```

deretter klargjøres parametere,

```
144     Vector<String> query = new Vector<String>();
        query.add(user);
```

og startLogin på serveren kalles.

```
148     conn.send("startLogin", query);
```

Når denne returnerer klargjøres nye parametere,

```
150     query.clear();
        query.add(hash(hash(password) + nextChallenge));
        query.add(user);
```

finishLogin kalles og resultat lagres.

```
155     String success = (String)conn.send("finishLogin", query).get("success");
```

Ved oppstart av backup sjekkes det at ikke backup eller restore allerede pågår, at det er spesifisert mål for backup og at brukeren er logget inn. Deretter kalles checkBackup på serveren for å sikre at metadataene ikke er korrupte.

```
209     conn.send("checkBackup", query);
```

Deretter genereres rdiff-backup-kommandoene i henhold til operativsystem på klienten og mål og unntak for backup og de kalles sekvensielt med Rdiffwrapper:

```
211     ArrayList cmds = new ArrayList();
        if (osName.startsWith("Linux") || osName.startsWith("Mac")) {
            cmds = buildBackupCmdUnix();
        }
        else if (osName.startsWith("Windows")) {
            cmds = buildBackupCmdWin();
        }

        Boolean success = true;

        for (int i = 0; i < cmds.size(); i++) {
```

```
            String[] cmd = (String[])cmds.get(i);

            // Tell Rdiffwrapper to start and break out of loop if it fails.
            if (!rdiff.start(cmd)) {
                success = false;
                break;
            }
        }
```

Implementasjonen av restore har mange likheter med backup. Først sjekkes det at ingen annen backup eller restore pågår og at brukeren er innlogget.

Deretter kalles listFiles for å bekrefte at den forespurte filen/katalogen finnes i backupen.

```
272         Hashtable result = listFiles(target);
```

Dersom resultat inneholder treff er det et gyldig mål og restore kan gjennomføres:

```
273         if (!result.isEmpty()) {
```

rdiff-backup-kommando genereres utfra operativsystem og mål for restore og kommandoen kalles med Rdiffwrapper.

BackupClient inneholder mange metoder som lar brukergrensesnittet hente eller lagre informasjon. Disse er trivielle og trenger ikke nærmere forklaring her.

Lagring av konfigurasjonen foregår ved at de aktuelle variablene synkroniseres til Config-objekt og dette serialseres til fil. Lasting av konfigurasjon er den motsatte operasjonen; et Config-objekt leses fra fil og variablene leses inn fra dette.

7.2.2 UserInterface

Inneholder kun tomme metodedeklarasjoner for metoder som klienten forventer å finne i brukergrensesnittet for tilbakemelding til brukeren.

7.2.3 Cli

Kommandolinjegrensesnittet implementerer metodene fra UserInterface med enkel skriving til konsollet.

Klassen har i likhet med serverens grensesnitt en separat tråd som leser linjer fra tastaturet.

```
39  public void run() {
        .
        .
```

```
            while (running) {
                string = bufferedreader.readLine();
```

7.2.4 Gui

Denne klassen gjør ingenting annet enn å implementere UserInterface med tomme metoder og å skrive en melding om at GUI ikke er implementert.

7.2.5 Config

Denne klassen replikerer alle konfigurasjonsvalgene fra BackupServer og implementerer Serializable for å kunne skrives til fil.

7.2.6 Connection

Når BackupClient oppretter en ny Connection lager denne en ny XmlRpcClient:

```
27      String server_url = "http://" + host + ":" + Integer.toString(port);

            try {
                server = new XmlRpcClient(server_url);
```

Ved XML-RPC-kall å serveren lagres resultat i en Hashtable, den inkluderte challenge lagres for å autentisere neste kall og resultat returneres til BackupClient:

```
44  public synchronized Hashtable send(String action, Vector query) {
        try {
            // Call the server, and get our result.
            Hashtable result = (Hashtable)server.execute("BackupServer."
                    + action, query);

            // Record next authentification challenge
            String challenge = (String)result.remove("challenge");
            bc.setChallenge(challenge);

            return result;
        }
```

7.2.7 Rdiffwrapper

Rdiffwrapper implementerer en egen StreamReader-klasse for å lese stdout og stderr fra rdiff-backup:

```
20  StreamReader(InputStream is, UserInterface ui) {
            this.is = is;
            .
```

```
        .
public void run() {
    try {
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);
            String line = null;
            while ((line = br.readLine()) != null) {
                if (line.startsWith("Processing changed file")) {
                .
                .
                else if (line.startsWith("Starting restore of")) {
                .
                .
                else if (line.endsWith("specify --force to overwrite.")) {
```

Når Rdiffwrapper kaller rdiff-backup lager den først et nytt Runtime-miljø for rdiff-backup å kjøre i

```
86      rt = Runtime.getRuntime();
```

deretter eksekveres kommandoen i dette miljøet,

```
Process proc = rt.exec(cmd);
```

og stdout og stderr tilordnes hver sin StreamReader.

```
StreamReader errors = new StreamReader(proc.getErrorStream(), ui);
StreamReader output = new StreamReader(proc.getInputStream(), ui);
errors.start();
output.start();
```

Rdiffwrapper venter så på at prosessen skal avslutte:

```
int exitVal = proc.waitFor();
```

Del 8

Observasjoner og resultater

I denne delen presenteres observasjoner av det ferdige systemet og resultater av testing.

8.1 Installasjon av systemet

Oppgaven spesifiserer at systemet skal ha en lav brukerterskel. Dette innebærer at installasjonen må kunne foregå enkelt på alle operativsystem.

Å lage en enkel pakke for klient og server viser seg vanskelig, da noen av bibliotekene i XML-RPC-implementasjonen må bygges på den enkelte maskin og dette krever det Java-baserte byggeverktøyet Apache Ant[5]. For serveren er ikke dette noe stort problem, da installasjonen på den uansett er en engangsjobb og vil utføres av noen med god kompetanse innen området. Det går derimot ikke an å kreve dette av vanlige brukere.

Installasjon av rdiff-backup i Windows går greit for vanlige brukere med en ferdig pakke med automatisk installasjon. Solutions First har lagd en uoffisiell pakke[1] som har blitt benyttet i testingen av Windows-klienten. Dersom en sånn pakke ikke kan benyttes må Cygwin, librsync og rdiff-backup installeres manuelt. Dette vil medføre at installasjonen blir for vanskelig for mange brukere.

8.2 rdiff-backup

I løpet av arbeidet med backup-systemet har rdiff-backup blitt testet mye. Det er stort sett en god løsning, men lider under noen svakheter, større og mindre.

[1] http://solutionsfirst.com.au/ dave/backup/

Ved restore av eksisterende kataloger har ikke rdiff-backup noen måte å gjenopprette kun endrede og/eller ikke-eksisterende filer; enten gjenopprettes ingen ting, eller så slettes den eksisterende katalogen med alt innhold før den gjenopprettes til det spesifiserte tidspunktet. Slettingen av katalogen kan omgås ved å gjenopprette kun enkeltfiler, men dette kan bli betraktelig mer jobb enn om den kunne gjenopprette en hel katalog.

Et større problem er tilstanden til metadataene på serveren. Ved vellykket backup er alt som det skal være. Dersom backupen avbrytes eller feiler av andre årsaker blir metadataene korrupte. Dette kan i de fleste tilfellene korrigeres ved å slette filer fra metadataene som spesifisert i FAQ-en til rdiff-backup[2] og sånn gå tilbake til det forrige settet metadata. Serveren implementerer denne fiksen i sin checkBackup-metode og i de fleste tilfeller er dette nok til å få metadataene tilbake til en tidligere fungerende tilstand.

I testing av systemet med simulering av nettverksfeil har det dog ved to anledninger lyktes å få metadataene i en tilstand der dette ikke har vært nok. Å forsøke å reprodusere feilen har ikke lyktes. Det antas derfor at metadataene i løpet av backupen er i en tilstand der eventuell stopp i backup-prosessen gjør at det ikke vil være nok å gå tilbake til det forrige settet metadata. Å undersøke dette nærmere går ut over rammene for denne oppgaven og vil innebære en grundig gjennomgang av kildekoden til rdiff-backup.

8.3 BackupClient

Backup-systemet har vært utviklet og testet med tildels gode resultater på Linux. Backup har vært gjennomført på Windows-klient med rdiff-backup for Windows installert fra Solutions First sin uoffisielle pakke. Ved forsøk på restore i Windows feiler rdiff-backup med denne installasjonen. Klienten har ikke vært testet på Mac OS X, men operativsystemets Unix-base gjør det meget sannsynlig at det vil fungere som på Linux eventuelt behøve kun mindre endringer.

Det har ikke lyktes å la prototypen av BackupClient kommunisere med SSH-innloggingen som rdiff-backup benytter til serveren. Dette resulterer i at passordet må skrives inn manuelt for hver gang rdiff-backup eksekveres av klienten. Når backup startes automatisk av klienten blir ikke input i konsollet sendt til SSH-prosessen og rdiff-backup feiler derfor. En vei omkring disse problemene er å ha en opplåst eller passordfri SSH-nøkkel til serveren.

[2] http://www.nongnu.org/rdiff-backup/FAQ.html#regress_failure

8.4 BackupServer

Serveren er utviklet for å kjøre på Linux, men vil sannsynligvis også fungere på andre Unixer med de nødvendige programmene og bibliotekene tilgjengelig.

For å kunne lese de forskjellige brukernes rdiff-backup-loggfiler må serveren kjøre som superbruker (root). Dette er i utgangspunktet noe man ønsker å unngå av sikkerhetsgrunner. Før serveren settes i drift som superbruker bør sikkerheten i den sjekkes grundigere enn hva tidsrammene for denne oppgaven har tillatt.

8.5 Test mot kravspesifikasjon

Her settes det ferdige systemet opp mot kravspesifikasjonen som ble utarbeidet før arbeidet startet. For hvert krav spesifiseres det om kravet er helt, delvis eller ikke oppfylt med en kort begrunnelse.

8.5.1 Omfang

Systemet skal:

- *inneholde serverprogramvare for en sentral backup-server.*
  Dette kravet er oppfylt. Pakken `backup.server` inneholder en serverløsning for bruk i dette systemet.
- *inneholde prototype for klientprogramvare for Windows, Linux og Mac OS X som lar brukeren koble til serveren for å sikkerhetskopiere og gjenopprette data.*
  Dette kravet er delvis oppfylt, delvis ikke oppfylt og delvis utestet, se 8.3.

8.5.2 Autentisering

Systemet skal:

- *autentisere tilkoblinger til backup-tjenesten med kryptert innlogging.*
  Dette kravet er ikke oppfylt. Alle kall autentiseres uten at brukerens passord på noe tidspunkt sendes over nettet, men kryptering av kommunikasjonen mellom klient og server er ikke implementert.
- *kun gi hver enkelt bruker oversikt over og tilgang til sine egne filer.*
  Dette kravet er i utgangspunktet oppfylt. Å få en liste over filene fordrer at brukeren er logget inn i systemet og å overføre data med rdiff-backup fordrer at denne får logget inn korrekt med SSH. At listen med data overføres ukryptert tilbake til klienten er dog en innsigelse mot dette.

8.5.3 Backup og restore

Systemet skal:

- *kryptere alle data som overføres mellom klient og server.*
  Dette kravet er delvis oppfylt. rdiff-backup sin transport av data krypteres med SSH, men kryptering av XML-RPC-kall er ikke implementert.
- *tilby brukeren backup og restore av enkeltfiler.*
  Dette kravet er oppfylt. BackupClient lar brukeren gjenopprette enkeltfiler.
- *tilby brukeren backup og restore av hele katalogstrukturer.*
  Dette kravet er oppfylt. BackupClient lar brukeren gjenopprette hele katalogstrukturer.
- *la brukeren starte backup manuelt.*
  Dette kravet er oppfylt. BackupClient lar brukeren starte backup manuelt.
- *kunne starte backup automatisk på brukerspesifiserte tidspunkt.*
  Dette kravet er delvis oppfylt. BackupClient kan starte backup automatisk på brukerspesifiserte tidspunkt, men dette fungerer kun dersom SSH-innlogging skjer passordløst med SSH-nøkkel.

Systemet bør:

- *forsøke å minimalisere mengden overførte data.*
  Dette kravet er oppfylt. rdiff-backup overfører kun endringer innad i hver enkelt fil.

8.5.4 Lagring

System bør:

- *kryptere alle sikkerhetskopierte data som lagres på serveren for å sikre disse ved en eventuell kompromittering av servermaskinen.*
  Dette kravet er ikke oppfylt.
- *lagre data på serveren på en måte som begrenser den nødvendige lagringsplassen.*
  Dette kravet er oppfylt. rdiff-backup lagrer siste versjon av filene og differansen til tidligere versjoner på serveren.

8.5.5 Dokumentasjon

Systemet skal:

- *leveres med brukerveiledning og kodedokumentasjon.*
  Dette kravet er oppfylt. Brukerveiledning finnes i vedlegg A og koden har fullstendig Javadoc.

8.6 Oppsummering

Disse feilene og mangler er identifisert i det nåværende backup-systemet, de alvorligste først:

- Metadataene til rdiff-backup kan tilsynelatende havne i en tilstand der det ikke holder å gå tilbake til forrige versjon. Dette medfører at all backup tilknyttet disse metadataene blir utilgjengelig. For Linux- og Mac-klienter vil dette omfatte alle data fra klienten, for Windows-klienter alle data fra en diskstasjon.
- rdiff-backup under Windows viser seg å være en for lite utprøvd løsning. Det finnes en ferdig pakke med rdiff-backup og Cygwin, men med denne feiler restore-operasjonen.
- XML-RPC-implementasjonen krever for mye av brukeren ved installasjon i form av bygging av biblioteker og krav til ekstern verktøy til dette. Dette medfører at det er vanskelig å lage en pakke for enkel installasjon av det komplette systemet.
- SSH-innlogging til rdiff-backup wrappes ikke av BackupClient. Dette medfører at automatisk backup ikke fungerer.
- BackupServer må kjøre med superbrukertilgang. Dette er en potensiell sikkerhetsrisiko.
- SSL-kryptering av kommunikasjon mellom BackupClient og BackupServer er ikke implementert. Dette medfører at uvedkommende kan overvåke nettverkstrafikken og få tak i listen av filer det er tatt backup av.

Del 9

Konklusjon

Denne rapporten har presentert en brukerundersøkelse om backup av bærbare PC-er, en oversikt over eksisterende backup-teknologi og design og implementasjon av et system for bruker-drevet backup av PC-er.

Backup-systemet som har blitt utviklet baserer seg på rdiff-backup, et backup-verktøy som ved hjelp av rsync-algoritmen kun overfører endringer internt i hver enkelt fil. Datatransporten går over SSH og annen kommunikasjon mellom klient og server benytter XML-RPC.

9.1 Backup-systemet

Prototypen av backup-systemet viser seg vanskelig å installere for vanlige brukere, da XML-RPC-implementasjonen som har blitt benyttet krever eksterne verktøy og flere steg for installasjon. Dette er også et problem ved installasjon av rdiff-backup i Windows. Det finnes en ferdig pakke med Cygwin og rdiff-backup, men med denne fungerer ikke restore-operasjoner. Automatisk backup fungerer ikke, grunnet prototypens problemer med å kommunisere med SSH-prosessen startet av rdiff-backup.

Å måtte kjøre BackupServer som superbruker er ikke ønskelig, men nødvendig for å få tilgang til loggene fra rdiff-backup. Manglende kryptering av kommunikasjonen mellom BackupClient og BackupServer er et svakt punkt i brukersikkerheten.

9.2 rdiff-backup

En viktig erfaring gjort i denne oppgaven er at det er for lett å korruptere metadataene til rdiff-backup ved avbrutt backup. Korrupte metadata medfører at

rdiff-backup verken kan gjennomføre backup eller restore mot datalageret. Dette er en dårlig egenskap for et backup-system. I tilfeller der metadataene har blitt korrupte forsøker BackupServer å falle tilbake på gamle metadata. Dette løser problemet i de aller fleste tilfellene, men ikke alle.

Det viser seg også at rdiff-backup på Windows er en ustabil og utilgjengelig løsning. Støtten for det under Cygwin er ikke fullstendig, noe som er meget uheldig for et backup-system.

Konklusjonen om rdiff-backup er at det er en god idé og et godt verktøy for brukere med nok kunnskap, men at det er sårbart for korrupsjon av metadata ved uforutsette hendelser i backup-prosessen og ikke godt nok støttet på Windows-plattformen.

9.3 Løsningen

Systemet som har blitt presentert i denne rapporten har i dag flere mangler som må rettes før det kan settes i drift som en fullverdig backup-løsning. De fleste av disse feilene kan korrigeres, men to punkter gjenstår som gjør fremtiden til en rdiff-backup-basert løsning usikker; dens sårbarhet ovenfor korrupte metadata og problemer med en stabil og enkel løsning i Windows.

Disse problemene er så viktige og store at videre satsing på et system basert på rdiff-backup ikke anbefales før konkrete og stabile løsninger på dem eventuelt finnes. En alternativ løsning som anbefales undersøkt nøyere er Box Backup[6]. Den ble valgt bort fordi den krypterer backup-dataene med en privat nøkkel. Tap av denne nøkkelen vil medføre at backupen ikke kan dekrypteres. Det bør vurderes om den tapte brukervennligheten ved sertifikatsignering og å måtte ha en ekstra kopi av nøkkelen sin kan være løsningen for å få et backup-system som på best mulig vis oppfyller kravene som stilles.

Del 10

Videre arbeid

I denne delen presenteres forslag til videre arbeid mot en ferdig løsning for backup av nomadiske PC-er ved IDI.

10.1 Backup-systemet

Som påpekt i konklusjonen har rdiff-backup to virkelig store problemer som ødelegger. Problemet relatert til korrupte metadata kan vies mer tid enn det har vært rom for i denne oppgaven. Hvis det finnes en løsning på dette er det et viktig steg nærmere å være et godt grunnlag for et backup-system. Det andre problemet handler om brukervennlighet ved installasjon og generell funksjonalitet og stabilitet på Windows-plattformen. Å teste forskjellige konfigurasjoner av Cygwin og de nødvendige komponentene og ut fra det finne en løsning som kan kombineres i en enkel installasjonspakke vil eventuelt løse det andre store problemet.

Andre punkt for å korrigere feil og videreutvikle systemet:

- Det må finnes en måte å wrappe SSH-prosessen som startes av rdiff-backup.
- Det bør undersøkes om det finnes alternative XML-RPC-implementasjoner for Java som lar seg installere med mindre jobb enn Apache XML-RPC.
- Kryptert versjoner av Connection og ConnectionListener bør implementeres for å sikre kommunikasjonen.
- Gammel backup bør slettes fra serveren. Den faktiske slettingen kan implementeres ved hjelp av rdiff-backup sitt parameter `--remove-older-than` og felter kan slettes fra databasen med utgangspunkt i feltet `date`. Dette vil i likhet med den implementerte løsningen for logganalyse kreve at serveren kjører med superbruker-tilgang.

10.2 Box Backup

Av backup-løsningene som ble vurdert innledningsvis i oppgaven var Box Backup den som kom nærmest av de som ikke ble valgt. Momentet den ble valgt bort på var løsningen med en privat nøkkel for kryptering av dataene. Det bør revurderes om litt brukervennlighet kan ofres i installasjonsprosedyren til fordel for å få et system som ser ut til å oppfylle de andre kravene som stilles.

Box Backup kan kjøres i Windows med Cygwin, men erfaringen fra rdiff-backup er at dette ikke alltid virker så godt som ønskelig. Derfor har Box Backup blitt portet til Windows. Denne versjonen er ennå merket som “Eksperimentell” og inneholder kjente bugs, men det er grunn til å tro at dette vil bedre seg i senere versjoner.

En veldig viktig del av jobben vil da være å få brukerne til å forstå viktigheten av å ha en eller flere separate kopier av den private nøkkelen sin. Alternative måter å gjøre dette på kan vurderes, som å skrive ut nøkkelen på papir eller å tilby brukere som ønsker det å lagre en digital kopi av nøkkelen hos IDI.

Referanser

[1] Apache XML-RPC 2.0. http://ws.apache.org/xmlrpc/xmlrpc2/.

[2] Java 2 Platform Standard Edition 5.0. http://java.sun.com/j2se/1.5.0/.

[3] The rsync algorithm. http://www.samba.org/rsync/tech_report/.

[4] AMANDA, the Advanced Maryland Automatic Network Disk Archiver. http://www.amanda.org.

[5] Apache Ant. http://ant.apache.org/.

[6] Box Backup. http://www.fluffy.co.uk/boxbackup/.

[7] rdiff backup. http://www.nongnu.org/rdiff-backup/.

[8] BackupPC. http://backuppc.sourceforge.net/info.html.

[9] Bacula. http://www.bacula.org/.

[10] Capivara. http://capivara.sourceforge.net/.

[11] Cygwin. http://www.cygwin.com/.

[12] dump. http://dump.sourceforge.net/.

[13] Duplicity. http://www.nongnu.org/duplicity/.

[14] James Gosling og Henry McGilton. The Java language environment: A white paper. Technical report, Sun Microsystems, oktober 1996.

[15] XML-RPC: Simple cross-platform distributed computing based on the standards of the Internet. http://www.xmlrpc.com.

[16] Javadoc. http://java.sun.com/j2se/javadoc/.

[17] Kerberos. http://web.mit.edu/kerberos/.

[18] MySQL. http://www.mysql.com/.

[19] NasBackup. http://www.nasbackup.com/.

[20] The Perl Directory: About Perl. http://www.perl.org/about.html.

[21] Gregory Pluta, Larry Brumbaugh, William Yurcik og Joseph Tucek. Who Moved My Data? A Backup Tracking System for Dynamic Workstation Environments. November 2004.

[22] What is Python? http://www.python.org/doc/Summary.html.

[23] rsync. http://rsync.samba.org/.

[24] rsync.py. http://www.vdesmedt.com/~vds2212/rsync.html.

[25] Samba. http://www.samba.org/.

[26] Subversion. http://subversion.tigris.org/.

[27] Concurrent Versions System. http://www.nongnu.org/cvs/.

[28] GNU tar. http://www.gnu.org/software/tar/.

[29] How Volume Shadow Copy Service Works. http://technet2.microsoft.com/WindowsServer/en/Library/2b0d2457-b7d8-42c3-b6c9-59c145b7765f1033.mspx.

[30] Xtar. http://www.helios.de/news/news03/N_06_03.phtml.

Vedlegg A

Brukerveiledning for backup-system

Dette er en kortfattet brukerveiledning for installering og bruk av BackupClient og BackupServer.

A.1 Installasjon

I denne delen forklares først steg som er felles for både klient og server, deretter spesifikke intruksjoner for serveren.

A.1.1 rdiff-backup

rdiff-backup må finnes på både server og klient. Det kan lastes ned fra `http://www.nongnu.org/rdiff-backup/`. På Linux vil rdiff-backup i de fleste tilfeller tilgjengelig i pakkearkivet. For Windows-klienter kan en ferdig pakke med Cygwin, Python og rdiff-backup lastes ned fra `http://solutionsfirst.com.au/~dave/backup/`. Windows-brukere som ønsker å installere de nødvendige delene selv kan finne instruksjoner på `http://katastrophos.net/andre/blog/?p=19`.

A.1.2 Java

Både server og klient krever Java Runtime Environment 5.0 (JRE). For å installere Apache XML-RPC (se A.1.3) trengs Java Development Kit (JDK), som også inkluderer JRE.

Operativsystemspesifikke installasjonsinstruksjoner og nedlasting av JDK/JRE finnes på Suns websider: `http://java.sun.com/j2se/1.5.0/`.

A.1.3 Apache XML-RPC

For kommunikasjonen seg i mellom krever server og klient Apache XML-RPC 2.0 for Java[1]. Denne kan lastes ned fra `http://www.apache.org/dist/ws/xmlrpc/binaries/`.

Før Apache XML-RPC kan brukes må pakken bygges. Dette krever JDK (se A.1.2) og Ant. Ant kan lastes ned fra `http://ant.apache.org/`.

Når Apache XML-RPC er ferdig bygget må `./xmlrpc-2.0.jar` og `./lib/commons-codec-1.3.jar` fra XML-RPC-katalogen legges til i CLASSPATH.

A.1.4 Server

Serveren trenger en MySQL-driver for JDBC for å kunne koble til databasen. Denne finnes i en del distribusjoner av Linux i pakkearkivet, eller kan lastes ned fra `http://dev.mysql.com/downloads/index.html`. Når driveren er installert må `mysql-connector-java-3.1.13-bin.jar` (eller tilsvarende, avhengig av versjon og distribusjon) legges til i CLASSPATH.

Alle brukere av backup-systemet må ha egen hjemmekatalog og SSH-innlogging på serveren. Når en ny bruker legges til må det opprettes en katalog `~brukernavn/backup/` der backup vil bli lagret.

A.2 Bruk

Denne delen forklarer bruk av server og klient.

A.2.1 Server

Serveren startes med kommandoen

```
java backup.server.BackupServer
```

[1] http://ws.apache.org/xmlrpc/

De tilgjengelige kommandolinjeparameterne er:

`-v`	Verbose. Skriver informasjon om hendelser som innlogging, sjekk av metadata o.l. til konsollet.
`-p <port>`	Portnummer. Spesifiserer portnummer serveren skal lytte etter XML-RPC-kall på.

Etter oppstart presenterer serveren en enkel kommandolinje:

```
>
```

Operasjon

Kommandoer på serveren:

`list`	Lister innloggede brukere.
`help`	Viser hjelpeteksten.
`exit`	Avslutter serveren.

A.2.2 Klient

Serveren startes med kommandoen

```
java backup.client.BackupClient
```

De tilgjengelige kommandolinjeparameterne er:

`-g`	GUI. Starter det grafiske brukergrensesnittet (Ikke implementert i prototypen.)

Etter oppstart presenterer klienten en enkel kommandolinje:

```
Not logged in >
```

Etter innlogging endres dette til:

```
bruker@maskin >
```

der *bruker* er brukernavnet på backup-serveren og *maskin* er navnet på den lokale maskinen.

Operasjon

Kommandoene på klienten:

`login`	Logger inn på serveren.
`add <mål>`	Legger til et nytt mål for backup.
`remove <mål>`	Fjerner et eksisterende mål for backup.
`ignore <mål>`	Legger til et nytt unntak for backup.
`unignore`	Fjerner et eksisterende unntak for backup.
`list [targets|ignores]`	Lister mål og/eller unntak for backup.
`set [valg [verdi]]`	Viser eller setter innstillinger i konfigurasjonen[1].
`save`	Lagrer konfigurasjonen til disk.
`backup`	Starter backup manuelt.
`restore [-f] [-t ts] <mål>`	Starter restore av et spesifisert mål. `-f` spesifiserer tvunget overskriving[2] og `ts` er timestamp som rapportert fra serveren.
`files [søkestreng]`	Lister tilgjengelig versjoner av filer og kataloger i backupen som svarer til søkestrengen. Dersom det ikke gis noen søkestreng listes alle filer og kataloger.
`exit`	Avslutter klienten.
	Viser hjelpeteksten.

[1] "`set`" viser alle valgene, "`set valg`" viser ett enkelt valg og "`set valg verdi`" setter et valg. Tilgjengelig valg finnes i tabellen under.

[2] Tvunget overskriving må brukes meget forsiktig! Dersom målet er en katalog vil rdiff-backup slette *alle* filer i denne katalogen før restore starter, også filer som ikke finnes i backupen.

`user`	Brukernavn på backup-serveren. Default: Lokalt brukernavn
`password`	Passord på backup-serveren. Default: Tomt
`host`	Adresse til backup-serveren. Default: "localhost"
`port`	Port backup-serveren lytter på. Default: "9798"
`hostname`	Navn på den lokale maskinen. Default: Lokalt maskinnavn
`autobackup`	true eller false angir om automatisk backup skal utføres[3]. Default: "false"
`autobackuptime`	Tidspunkt for automatisk backup. Default: "12:00"
`autologin`	true eller false angir om klienten skal logge inn på serveren automatisk ved oppstart. Default: "false"
`rdiffcommand`	Sti til rdiff-backup. Default i Linux og Mac OS X: "rdiff-backup". Default i Windows: "c:\rdiff\rdiff-backup.bat"

[3] I prototypesystemet fungerer automatisk backup kun ved passordløs innlogging med SSH-nøkkel.

Vedlegg B

Resultater fra brukerundersøkelse

B.1 Spørsmål

1. Hva er din tilknytning til IDI?
2. Hvilke(t) operativsystem bruker du på din bærbare PC?
3. Hvor ofte tar du backup av data på din bærbare PC?
4. Hvis du ikke svarte “Aldri” på forrige spørsmål, hvordan tas denne backupen?
5. Hvor mye data ville du tatt backup av fra din bærbare PC?
6. Hvor ofte ville du tatt backup?
7. Hva slags internetttilknytning ville du benyttet? (Du kan velge flere)
8. Hvor lang tid mener du det ville være akseptabelt å vente mens en backup kjører?
9. Burde backup utføres på faste tidspunkt, ved tilkobling til NTNUs nett e.l. uten at du selv måtte gjøre noe, eller ville ønsket å kontrollere dette selv?
10. Ville du benyttet tjenesten dersom dataene dine ble lagret ukryptert på backuptjener, og administrator dermed i teorien hadde tilgang til dem?
11. Ville det vært viktig for deg at det ble tatt backup av filrettigheter?
12. Eventuelle kommentarer:

B.2 Svar

Svar nummer: 1
1. Ansatt
2. Mac OS X
3. Flere ganger i uka
4. -Ved hjelp av scp til hjemmekatalog (Solaris).
5. 100MB - 500MB
6. Hver dag
7. NTNUs fastnett, med kabel, Privat bredbånd
8. Annet: -Bør helst kunne kjøre i bakgrunnen (og kan da godt kjøre lenge)
9. Det burde styres av backup-programvaren
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Konfigurerbarhet er bra. Jeg vil/trenger f.eks. ikke ta backup av *alt* på laptop-en – f.eks. bare enkelte kataloger (kanskje igjen med unntak av enkelte filer).Mengden data over er et estimat på mengden data (totalt for de gjeldende filene) som ville kunne måtte synkroniseres hver gang. Den totale lagrede datamengden er litt usikkert, men kanskje i samme størrelsesorden.

Svar nummer: 2
1. Ansatt
2. Mac OS X
3. Hver dag
4. -Jeg bruker Apple sitt backupprogram og sikkerhetskopierer 2006-filer mot .mac. Data fra tidligere å har jeg på CD/DVD.
5. 500MB - 1GB
6. Hver dag
7. NTNUs fastnett, med kabel
8. 5 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 3
1. Ansatt
2. Windows
3. Sjeldnere
4. -Ved ujevne mellomrom kopierer jeg filene fra katalogene med dokumenter over på stasjonær PC eller ekstern harddisk. Kopierer altså kun dokumenter.
5. 500MB - 1GB
6. Minst en gang i måneden
7. NTNUs trådløsnett
8. 45 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 4
1. Ansatt
2. Windows
3. Minst en gang i uka
4. -Jeg tar rutinemessig backup av det jeg vurderer som viktige filer etter hvert som de endres. Frekvensen blir dermed ganske uregelmessig.
5. 1GB - 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. Annet: -Uvesentlig dersom backup kunne kjøre om natta.
9. Jeg ville ha kontroll på dette selv
10. Nei
11. Nei, strukturen og innholdet i filene er det viktige
12. -Jeg ser ikke det store behovet for en slik tjeneste. Jeg greier å holde styr på nødvendig backup selv.

Svar nummer: 5
1. Ansatt
2. Windows
3. Aldri
4.
5. Mindre enn 100MB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. Annet: -Tidsbruk er uvesentlig hvis backup kjører om natten, evt. som bakgrunnsjobb.
9. Det burde styres av backup-programvaren
10. Nei
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 6
1. Ansatt
2. Windows
3. Sjeldnere
4. -Viktige filer kopieres til USB-stick, og av og til kopierer jeg større deler til hjemmeområde på jobb eller hjemme
5. Over 5GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 10 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Jeg vil gjerne selv kunne styre oppsettet av en automatikk (med mulighet for manuell overstyring). Jeg vil også gjerne kunne styre hva som det blir tatt backup (dvs kunne ekskludere enkelte områder).

Svar nummer: 7
1. Student
2. Mac OS X
3. Aldri
4. -har alt mitt arbeide på idi/stud sine tjenere, og mapper opp disse når jeg skal jobbe.
5. 1GB - 5GB
6. Hver dag
7. NTNUs fastnett, med kabel
8. 1 minutt
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 8
1. Ansatt
2. Windows
3. Minst en gang i måneden
4. -Til en 300Gb Firewire ekstern HD.
5. Over 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel
8. 20 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Veldig bra initiativ. Dette har mange savnet.Lag en engelsk versjon for alle de ved IDI som ikke er norske.

Svar nummer: 9
1. Ansatt
2. Windows
3. Sjeldnere
4. -Brenner utvalgte katalogtrær på CD. Har også brukt et program for inkrementell backup, men det fungerte for dårlig.
5. 1GB - 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel
8. Annet: -Over natta, ved å sette igjen laptop'en en natt i uka.
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Kommentar til spm. 9: Jeg ønsker ikke kontroll på hvordan backup'en tas (selv om det kan være greit å angi hvilke katalogtrær som er relevante), men ønsker å angi når det tas, f.eks. hver uke (altså så snart som mulig etter at en uke er gått) mellom 22 og 7. Dette vil i praksis gjøre at det skjer automatisk og ofte nok, når jeg setter igjen laptop'en.

Svar nummer: 10
1. Ansatt
2. Windows
3. Minst en gang i måneden
4. -Backup på nett og på hjemmemaskin
5. 1GB - 5GB
6.
7. NTNUs fastnett, med kabel
8.
9. Jeg ville ha kontroll på dette selv
10. Nei
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 11
1. Ansatt
2. Windows

3. Hver dag
4.
5. Mindre enn 100MB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 10 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Jeg har stort sett filene på den bærbare PCen også på området mitt på NTNU så det er ikke så viktig å ta backup for meg på den bærbare PCen.

Svar nummer: 12
1. Ansatt
2. Windows Mac OS X
3. Aldri
4. -tar ikke manuelle backup.MEN bruker aktivt filserverene ved IDI til å lagre det aller meste av tingene mine
5. Mindre enn 100MB
6. Flere ganger i uka
7. NTNUs trådløsnett, Privat bredbånd
8. 45 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 13
1. Ansatt
2. Mac OS X
3. Minst en gang i måneden
4. -Har nettopp kjøpt ekstern harddisk. Før det brente jeg en cd/dvd.
5. 1GB - 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel
8. 10 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 14
1. Ansatt
2. Windows Linux
3. Aldri
4.
5. 100MB - 500MB
6. Hver dag
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 1 minutt
9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 15
1. Ansatt
2. Windows
3. Sjeldnere
4.
5. Over 5GB
6. Hver dag
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 45 minutter
9. Jeg ville ha kontroll på dette selv
10. Nei
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 16
1. Ansatt
2. Mac OS X
3. Sjeldnere
4.
5. Over 5GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 20 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 17
1. Ansatt
2. Mac OS X
3. Minst en gang i måneden
4. -Kopiering av utvalgte filer til ekstern disk
5. 500MB - 1GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 5 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 18
1. Ansatt
2. Windows
3. Sjeldnere
4. -Kopi av sentrale områder til privat PC eller totalkopi til nettverksdrev som slettes etter en uke.
5. Over 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel
8. Annet: -Opptil 2 timer, men jeg måtte kunne arbeide samtidig.
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12. -En ideell løsning bør bygge en grunnbase av data og deretter kun kopiere endringer slik at man slapp å vente “evigheter” på en backup hver gang. En manuelt styrt backupløsning bør også ha en ’påminnelsesfunksjon’ som sjekker når siste backup er kjørt og sender en form for advarsel (pop-up) om at “nå bør du ta backup igjen”

Svar nummer: 19
1. Ansatt
2. Linux
3. Sjeldnere
4. -kopierer over til annen disk (hjemmeområde på IDI eller annen disk uten backup) når jeg føler behov for det
5. 1GB - 5GB
6. Minst en gang i måneden
7. NTNUs fastnett, med kabel, NTNUs trådløsnett
8. 10 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12. -vil ha automatisk backup (spm. 9), men diskret beskjed om at det er tatt backup/hvor lenge det er siden forrige backup o.l.vil ha fleksibel løsning for å bestemme hva det skal tas backup av (regexp e.l.)

Svar nummer: 20
1. Ansatt
2. Mac OS X
3. Sjeldnere
4. -Manuell kopi til ekstern harddisk
5. 500MB - 1GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 5 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 21
1. Ansatt
2. Windows
3. Aldri
4.
5. Over 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel
8. 5 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 22
1. Ansatt
2. Windows
3. Minst en gang i måneden
4. -CVS for et par programmeringsprosjekter hvor alt av egne skriftlige ting også blir committet.Bruker ellers backup på ekstern harddisk (gjøres sjeldent). Det meste av arbeidsrelaterte filer er på samba.idi.ntnu.no siden det i dag er mulig å jobbe mot

hjemmekatalogen fra de fleste plasser i verden.
5. Mindre enn 100MB
6. Minst en gang i måneden
7. NTNUs fastnett, med kabel, NTNUs trådløsnett
8. 10 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Med backup over nett ville det også vært mulig å kjøre backup hjemmefra. Da hadde det vært akseptabelt med lengre backup tid siden jeg bruker maskinen så og si hele tiden på arbeid, mens den ofte er påslått men ikke i bruk hjemme.

Svar nummer: 23
1. Ansatt
2. Windows Linux
3. Flere ganger i uka
4. -Alle data jeg ikke vil miste sjekker jeg inn CVS. Repositoriet ligger på hjemmeområdet mitt på selje.
5. 500MB - 1GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 5 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 24
1. Ansatt
2. Windows
3. Aldri
4.
5. 100MB - 500MB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 45 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Dette vil være en svært nyttig tjeneste for meg.

Svar nummer: 25
1. Ansatt
2. Mac OS X
3. Minst en gang i uka
4. -Ekstern harddisk med USB-grensesnitt
5. 1GB - 5GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel
8. 45 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Hadde diskkrasj på bærbar PC for to uker siden... Så bra å få backup-system på plass:)

Svar nummer: 26
1. Ansatt
2. Linux
3. Hver dag
4. -Har (nesten) alt i SVN-repositorium på IDIs sørver. Committer endringer minst en gang per dag.
5. 1GB - 5GB
6. Hver dag
7. NTNUs fastnett, med kabel
8. 5 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12. -Spørsmål 5- er forutsatt at jeg velger å bruke tjenesten. 10 og 11 er jeg ikke sikker på, men det hadde jo vært fint med den ekstra funksjonaliteten.

Svar nummer: 27
1. Ansatt
2. Windows
3. Minst en gang i måneden
4. -Delvis CVS, kopier til idi-konto
5. Mindre enn 100MB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 10 minutter
9. Jeg ville ha kontroll på dette selv
10. Nei
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 28
1. Ansatt
2. Windows
3. Sjeldnere
4. -tar zip av viktigste filer, og kopier dette så tile en av mine andre maskiner
5. 1GB - 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 45 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 29
1. Ansatt
2. Windows
3. Sjeldnere
4.
5. Over 5GB
6. Minst en gang i måneden
7. NTNUs fastnett, med kabel
8. 5 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 30
1. Ansatt
2. Windows Mac OS X Annet: -FreeBSD
3. Sjeldnere
4. -scp av utvalgte kataloger til andre maskiner
5. 1GB - 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 45 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 31
1. Ansatt
2. Linux
3. Hver dag
4. -cron-jobb-styrt rsync av utvalgte kataloger fra bærbar til en server på IDI. Dette innebærer altså en inkrementell backup, som har akkumulert til ca. 7.5 GB, mens overført mengde per dag er _mye_ mindre. Kanskje bare noen kB. (Diff.) backup utføres annenhver time i løpet av arbeidsdagen.
5. Over 5GB
6. Hver dag
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 20 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 32
1. Ansatt
2. Windows Linux
3. Minst en gang i uka
4. -cvs commit
5. Mindre enn 100MB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 5 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 33
1. Ansatt
2. Linux Mac OS X
3. Flere ganger i uka
4. -rsync av Documents og et par foldere til, til en katalog på login.idi
5. Mindre enn 100MB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 5 minutter

9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 34
1. Ansatt
2. Windows
3. Sjeldnere
4. -Bruker CD, tar ikke av alt, bare de “viktigste” katalogene / filene
5. 500MB - 1GB
6. Minst en gang i måneden
7. NTNUs fastnett, med kabel
8. 20 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 35
1. Ansatt
2. Mac OS X
3. Aldri
4.
5. 100MB - 500MB
6. Hver dag
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 1 minutt
9. Det burde styres av backup-programvaren
10. Nei
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12. -Jeg backer ikke opp laptop’en min fordi jeg har et godt verktøy til det. I stedet mapper jeg opp hjemmekatalogen på idi og lagrer alt av verdi der, i tillegg til at jeg bruker cvs e.l. på ting jeg arbeider aktivt med.

Svar nummer: 36
1. Ansatt
2. Windows
3. Minst en gang i uka
4.
5. 500MB - 1GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett
8. 20 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 37
1. Ansatt
2. Windows
3. Flere ganger i uka
4.
5. Mindre enn 100MB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel
8. 10 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 38
1. Ansatt
2. Windows
3. Sjeldnere
4. -Laster over manuelt til idi-kontoen
5. 500MB - 1GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 1 minutt
9. Jeg ville ha kontroll på dette selv
10. Nei
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 39
1. Ansatt
2. Windows
3. Minst en gang i måneden
4. -Manuell kopiering av filer til hjemmeområde
5. 100MB - 500MB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 10 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Ønsker et system! :-)

Svar nummer: 40
1. Ansatt
2. Mac OS X
3. Minst en gang i måneden
4. -Til ekstern firewire disk
5. Over 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 5 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12. -Ang. 8. Jeg regner med at en backup kan køre i baggrunden, så de 5 min er for “låste” filer.

Svar nummer: 41
1. Ansatt
2. Windows Linux
3. Aldri
4. -Jeg lagrer alt på hjemmeområdet til IDI via samba og subversion. Dokumenter på disk er dumt :)
5. 500MB - 1GB
6. Hver dag
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 1 minutt
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 42
1. Ansatt
2. Windows
3. Sjeldnere
4. -Kopierer til hjemmeområdet på NTNUs server
5. 1GB - 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 20 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 43
1. Ansatt
2. Linux
3. Minst en gang i måneden
4.
5. 1GB - 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel
8. 45 minutter
9. Det burde styres av backup-programvaren
10. Nei
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12. -For viktige dokumenter benytter jeg CVS som “backup” ved at CVS-repositoriet ligger på en IDI-server.

Svar nummer: 44
1. Ansatt
2. Windows
3. Aldri
4.
5. Mindre enn 100MB
6. Sjeldnere
7. NTNUs fastnett, med kabel
8. 45 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 45
1. Ansatt
2. Windows
3. Flere ganger i uka

4.
5. Mindre enn 100MB
6. Sjeldnere
7. Annet: -Backup på minnepinner
8.
9.
10.
11.
12. -Jeg ville ikke bruke en slik backup-tjeneste. Ser bare et poeng i å ta backup at det som jeg skriver selv, det er ikke store volumet. Foto vil jeg ha annen løsning på. Derfor ahr jeg ikke svart på de siste spm.

Svar nummer: 46
1. Ansatt
2. Mac OS X
3. Minst en gang i måneden
4.
5. Over 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 45 minutter
9. Jeg ville ha kontroll på dette selv
10. Nei
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 47
1. Ansatt
2. Linux
3. Minst en gang i uka
4. -rsync med Desktop ved behov (dvs. før og etter bruk av bærbar PC)... så det er ikke riktig backup i seg selv, men min Desktop kjører rsync med min IDI konto flere ganger daglig.
5. 1GB - 5GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. Annet: -etter behov som f.eks. med rsync
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 48
1. Ansatt
2. Windows
3. Sjeldnere
4. -Pr . I dag har jeg lagrer jeg (artikler f.eks.) primært på server - og lager kopier på server av ting jeg jobber med. Systematisk backup blir ikke gjort. Det er noe jeg savner, men det har ikke blitt slik at jeg har satt meg inn i hvordan det burde gjøres. Synes absolutt idi burde tilby assisistanse på det.
5. 1GB - 5GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel
8. 5 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12. -Hvor mye skal lagres - litt usikker her. Ser liten grunn til å ta backup av programvare. Kun mine filer - og evt outlook info (kalender, contacts, og mail som ikke ligger på mail-server)

Svar nummer: 49
1. Ansatt
2. Windows
3. Sjeldnere
4. -har ikke tatt backup av all data på bærbar PC i en runde, men har tatt backup av viktige mapper ved å bruke USB-brikke eller å flytte filer over til trådløst NAS.
5. 100MB - 500MB
6. Minst en gang i måneden
7. Privat bredbånd
8. Annet: -en natt
9. Det burde styres av backup-programvaren
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 50
1. Ansatt
2. Linux
3. Hver dag
4. -Subversion på annen maskin
5. 1GB - 5GB
6. Hver dag
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 10 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 51
1. Ansatt
2. Mac OS X
3. Sjeldnere
4. -Brenner DVD
5. Over 5GB
6. Hver dag
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 20 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 52
1. Ansatt
2. Mac OS X
3. Flere ganger i uka
4. -CVS
5. 100MB - 500MB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 5 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 53
1. Ansatt
2. Windows
3. Aldri
4. -Eg har ikkje data lagra lokalt på maskina. Eg lagrar alle data på server på idi.
5. Over 5GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. Annet: -Dette er vel ikkje noko eg kan velje sidan det er avhengig av data mengden, men 1 - 5 min lengre tid enn det tek å overføre data er uakseptabelt.
9. Det burde styres av backup-programvaren
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 54
1. Ansatt
2. Mac OS X
3. Flere ganger i uka
4. -ssh + tar.
5. 1GB - 5GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett
8. Annet: -forventer at backup foregår online
9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12. -det er og viktig at metadata, acl-er, attributter o.l blir tatt vare på.

Svar nummer: 55
1. Ansatt
2. Windows
3. Aldri
4.
5. 1GB - 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 5 minutter
9. Det burde styres av backup-programvaren
10. Nei
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 56
1. Ansatt
2. Windows
3. Flere ganger i uka
4. -Overfører data til IDI-brukeren min.
5. Mindre enn 100MB
6. Minst en gang i måneden

7. NTNUs fastnett, med kabel
8. 20 minutter
9. Jeg ville ha kontroll på dette selv
10. Nei
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 57
1. Ansatt
2. Mac OS X
3. Sjeldnere
4. -CD
5. 1GB - 5GB
6. Minst en gang i måneden
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 10 minutter
9. Det burde styres av backup-programvaren
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Svar nummer: 58
1. Ansatt
2. Mac OS X
3. Hver dag
4. -Bruker programmet Personal Backup mot stasjonærmaskin på jobb (programmet fikser alt fra remote login og synkronisering etc - starter automatisk hver dag)
5. Over 5GB
6. Hver dag
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. Annet: -Backup må kunne være inkrementell, med max tid 10 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Viktig også med tilbud om synkronisering mellom maskiner (mulighet for å jobbe på kopier av "samme" katalog på jobbmaskin og tilsvarende på bærbar).

Svar nummer: 59
1. Ansatt
2. Windows
3. Aldri
4.
5. Ville ikke tatt backup
6. Sjeldnere
7. Privat bredbånd
8. 10 minutter
9. Jeg ville ha kontroll på dette selv
10. Nei
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 60
1. Ansatt
2. Windows
3. Aldri
4. -Jeg lagrer viktige dokumenter på hjemmekatalog.
5. 500MB - 1GB
6. Flere ganger i uka
7. NTNUs fastnett, med kabel, NTNUs trådløsnett, Privat bredbånd
8. 5 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12.

Svar nummer: 61
1. Ansatt
2. Windows
3. Minst en gang i måneden
4. -Instituttets server (Les hjemmeområdet mitt).
5. Over 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel, Privat bredbånd
8. 5 minutter
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Nei, strukturen og innholdet i filene er det viktige
12. -Spm 9: Kan gjerne ha setting for at den programvaren skal gi en "reminder" dersom det f eks er gått mer enn en uke siden siste backup. Men alt dette MÅ det være mulig å sette selv.

Svar nummer: 62
1. Ansatt
2. Mac OS X
3. Minst en gang i måneden
4. -"Dump" til ekstern disk eller tjener.
5. Over 5GB
6. Minst en gang i uka
7. NTNUs fastnett, med kabel
8. Annet: -Det tar den tid det tar....
9. Jeg ville ha kontroll på dette selv
10. Ja
11. Ja, det er viktig for meg å kunne gjenopprette dataene nøyaktig som de var
12.

Vedlegg C

Kodelisting

C.1 Klient

C.1.1 BackupClient.java

```
1   package backup.client;

    import java.security.MessageDigest;
    import java.text.SimpleDateFormat;
5   import java.util.*;
    import java.io.*;

    /**
     * Main client engine.
10   *
     * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
     */
    public class BackupClient extends Thread {
        private UserInterface ui;
15
        private RdiffWrapper rdiff;

        // Encrypted or unencrypted connection
        // private SSLConnection conn;
20      private Connection conn;
        private ArrayList<String> targets;
        private ArrayList<String> ignores;
        private String host;
        private int port;
25      private String hostname;
        private String user;
        private String password;
        private Boolean autoBackup;
        private String autoBackupTime;
30      private Boolean autoLogin;
```

```
    private String rdiffCommand;

    private boolean running;
    private String osName;
    private boolean backupIsRunning = false;
    private Config config;
    private boolean loggedIn = false;
    private String nextChallenge;
    private final String configFilename = "config.dat";

    /**
     * Creates a new <code>BackupClient</code>.
     *
     * @param args command line arguments
     */
    public BackupClient(String[] args) {

        if (args.length > 0 && args[0].equals("-g")) {
            ui = new Gui(this);
        }
        else {
            ui = new Cli(this);
        }

        osName = System.getProperty("os.name");

        loadConfig();

        rdiff = new RdiffWrapper(ui);

        running = true;

        if (autoLogin) {
            login();
        }
    }

    /**
     * Main.
     *
     * @param args command line arguments
     */
    public static void main(String[] args) {
        BackupClient bc = new BackupClient(args);
        bc.run();
    }

    /**
     * Start main thread. This thread keeps track of tasks performed at regular
     * intervals.
     */
    public void run() {
        SimpleDateFormat formatter = new SimpleDateFormat("HH:mm");
        String prev = formatter.format(new Date());
```

```
        int mins = 0;

        try {
            while (running) {
                Thread.sleep(1000);
                String now = formatter.format(new Date());
                if (!now.equals(prev)) {
                    prev = now;
                    mins++;

                    // Send keep-alive signal to server every minute
                    if (isLoggedIn()) {
                        keepAlive();
                    }

                    // Start scheduled backup
                    if (autoBackup && autoBackupTime.equals(now)) {
                        startBackup();
                    }

                    // Reset counter every hour
                    if (mins == 60) {
                        mins = 0;
                    }
                }
            }
        }
        catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    /**
     * Tell the <code>BackupClient</code> to shut down.
     */
    public void exit() {
        if (ui != null) {
            ui.status("Client shutting down.");
        }
        if (isLoggedIn()) {
            logout();
        }
        running = false;
    }

    /**
     * Log in to the server. Requests a challenge from the server and
     * responds to it.
     */
    public void login() {
        if (loggedIn) {
            return;
        }
        ui.status("Logging in...");
```

```
140         // Encrypted or unencrypted connection
            // conn = new SSLConnection(this, host, port);
            conn = new Connection(this, host, port);

            Vector<String> query = new Vector<String>();
145         query.add(user);

            // Start login sequence and request a challenge from the server
            conn.send("startLogin", query);

150         query.clear();
            query.add(hash(hash(password) + nextChallenge));
            query.add(user);

            // Reply to the server with the response
155         String success = (String)conn.send("finishLogin", query).get("success");

            if (success.equals("true")) {
                loggedIn = true;
                ui.status("Logged in.");
160         }
            else {
                loggedIn = false;
                ui.warn("Login failed.");
            }
165     }

        /**
         * Log out from the server.
         */
170     public void logout() {
            if (!loggedIn) {
                return;
            }
            Vector<String> query = new Vector<String>();
175         query.add(hash(hash(password) + nextChallenge));
            query.add(user);
            conn.send("logout", query);
            loggedIn = false;
            ui.status("Logged out.");
180     }

        /**
         * Starts backup process.
         */
185     public void startBackup() {
            if (backupIsRunning) {
                ui.warn("Backup already running, aborting.");
                return;
            }
190
            if (targets.isEmpty()) {
                ui.warn("No targets for backup, aborting.");
```

```
            return;
        }

        if (!loggedIn) {
            ui.warn("Not logged in, aborting.");
            return;
        }

        ui.status("Starting backup...");
        backupIsRunning = true;

        Vector<String> query = new Vector<String>();
        query.add(hash(hash(password) + nextChallenge));
        query.add(user);

        // Make sure target directory on the server is clean
        conn.send("checkBackup", query);

        ArrayList cmds = new ArrayList();
        if (osName.startsWith("Linux") || osName.startsWith("Mac")) {
            cmds = buildBackupCmdUnix();
        }
        else if (osName.startsWith("Windows")) {
            cmds = buildBackupCmdWin();
        }

        Boolean success = true;

        for (int i = 0; i < cmds.size(); i++) {
            String[] cmd = (String[])cmds.get(i);

            // Tell Rdiffwrapper to start and break out of loop if it fails.
            if (!rdiff.start(cmd)) {
                success = false;
                break;
            }
        }

        // If everything is ok, tell server to update database
        if (success) {
            query.clear();
            query.add(hash(hash(password) + nextChallenge));
            query.add(user);

            // Update the server's file database
            conn.send("updateDB", query);
            ui.status("Backup finished.");
        }
        else {
            ui.status("Backup failed.");
        }

        backupIsRunning = false;
    }
```

```
        /**
         * Starts restore process.
250      *
         * @param timestamp timestamp of target to restore, according to
         *            rdiff-backup standards
         * @param target target to restore
         * @param force if true, rdiff-backup will delete existing target before
255      *            restoring
         */
        public void startRestore(String timestamp, String target, Boolean force) {
            if (backupIsRunning) {
                ui.warn("Backup already running, aborting.");
260             return;
            }

            if (!loggedIn) {
                ui.warn("Not logged in, aborting.");
265             return;
            }

            ui.status("Starting restore...");
            backupIsRunning = true;
270
            // In case of invalid target from the cli
            Hashtable result = listFiles(target);
            if (!result.isEmpty()) {
                String[] cmd = { "" };
275             if (osName.startsWith("Linux") || osName.startsWith("Mac")) {
                    cmd = buildRestoreCmdUnix(timestamp, target, force);
                }
                else if (osName.startsWith("Windows")) {
                    cmd = buildRestoreCmdWin(timestamp, target, force);
280             }

                if (rdiff.start(cmd)) {
                    ui.status("Restore finished.");
                }
285             else {
                    ui.status("Restore failed.");
                }
            }
            else {
290             ui.warn("Unknown target.");
            }

            backupIsRunning = false;
        }
295
        /**
         * Add new target for backup.
         *
         * @param target new target
300      */
```

```
    public void addTarget(String target) {
        if (targets.indexOf(target) == -1) {
            targets.add(target);
        }
305 }

    /**
     * Remove existing target for backup.
     *
310  * @param target existing target
     */
    public void removeTarget(String target) {
        int i = targets.indexOf(target);
        if (i != -1) {
315         targets.remove(i);
        }
    }

    /**
320  * Return current targets.
     *
     * @return current targets
     */
    public ArrayList listTargets() {
325     return targets;
    }

    /**
     * Add a new ignore from backup.
330  *
     * @param ignore new ignore
     */
    public void addIgnore(String ignore) {
        if (ignores.indexOf(ignore) == -1) {
335         ignores.add(ignore);
        }
    }

    /**
340  * Remove an existing ignore.
     *
     * @param ignore existing ignore
     */
    public void removeIgnore(String ignore) {
345     int i = ignores.indexOf(ignore);
        if (i != -1) {
            ignores.remove(i);
        }
    }
350
    /**
     * Return current ignores.
     *
     * @return current ignores
```

```
355         */
        public ArrayList listIgnores() {
            return ignores;
        }

360     /**
         * Request a list of files matching search term from BackupServer.
         *
         * @param search search term
         * @return list of files
365      */
        public Hashtable listFiles(String search) {
            if (!loggedIn) {
                ui.warn("Not logged in, aborting search.");
                return null;
370         }

            Vector<String> query = new Vector<String>();
            query.add(hash(hash(password) + nextChallenge));
            query.add(user);
375         query.add(search.replace("*", "%"));

            return conn.send("listFiles", query);
        }

380     /**
         * Are we logged in on the server?
         *
         * @return logged in
         */
385     public boolean isLoggedIn() {
            return loggedIn;
        }

        /**
390      * Get username.
         *
         * @return username
         */
        public String getUser() {
395         return user;
        }

        /**
         * Set username.
400      *
         * @param user username
         */
        public void setUser(String user) {
            this.user = user;
405     }

        /**
         * Set password.
```

```
     *
410  * @param password password
     */
    public void setPassword(String password) {
        this.password = password;
    }
415
    /**
     * Get local hostname.
     *
     * @return hostname
420  */
    public String getHostname() {
        return hostname;
    }

425 /**
     * Set local hostname.
     *
     * @param hostname hostname
     */
430 public void setHostname(String hostname) {
        this.hostname = hostname;
    }

    /**
435  * Get server address.
     *
     * @return server address
     */
    public String getHost() {
440     return host;
    }

    /**
     * Set server address.
445  *
     * @param host server address
     */
    public void setHost(String host) {
        this.host = host;
450 }

    /**
     * Get server port.
     *
455  * @return server port
     */
    public String getPort() {
        return String.valueOf(port);
    }
460
    /**
     * Set server port.
```

```
         *
         * @param port server port
465      */
        public void setPort(String port) {
            this.port = Integer.parseInt(port);
        }

470     /**
         * Get automatic backup on or off
         *
         * @return automatic backup
         */
475     public Boolean getAutoBackup() {
            return autoBackup;
        }

        /**
480      * Set automatic backup on or off
         *
         * @param autoBackup automatic backup
         */
        public void setAutoBackup(Boolean autoBackup) {
485         this.autoBackup = autoBackup;
        }

        /**
         * Get time of day for automatic backup in 24 hour format, e.g. "23:30".
490      *
         * @return time for automatic backup
         */
        public String getAutoBackupTime() {
            return autoBackupTime;
495     }

        /**
         * Set time of day for automatic backup in 24 hour format, e.g. "23:30".
         *
500      * @param autoBackupTime time for automatic backup
         */
        public void setAutoBackupTime(String autoBackupTime) {
            this.autoBackupTime = autoBackupTime;
        }
505
        /**
         * Get automatic login on startup on or off.
         *
         * @return automatic login
510      */
        public Boolean getAutoLogin() {
            return autoLogin;
        }

515     /**
         * Set automatic login on startup on or off.
```

```
         *
         * @param autoLogin automatic login
         */
520      public void setAutoLogin(Boolean autoLogin) {
             this.autoLogin = autoLogin;
         }

         /**
525       * Get path to rdiff-backup executable
          *
          * @return rdiff-backup command
          */
         public String getRdiffCommand() {
530          return rdiffCommand;
         }

         /**
          * Set path to rdiff-backup executable
535       *
          * @param rdiffCommand rdiff-backup executable
          */
         public void setRdiffCommand(String rdiffCommand) {
             this.rdiffCommand = rdiffCommand;
540      }

         /**
          * Pass a warning to the user interface.
          *
545       * @param s warning text
          */
         public void warn(String s) {
             ui.warn(s);
         }
550
         /**
          * Records a new challenge for authentification of next message.
          *
          * @param challenge the new challenge
555       */
         public void setChallenge(String challenge) {
             nextChallenge = challenge;
         }

560      /**
          * Saves the configuration to disk.
          */
         public void saveConfig() {
             try {
565              config.targets = targets;
                 config.ignores = ignores;
                 config.host = host;
                 config.port = port;
                 config.hostname = hostname;
570              config.user = user;
```

```
            config.password = password;
            config.autoBackup = autoBackup;
            config.autoBackupTime = autoBackupTime;
            config.autoLogin = autoLogin;
            config.rdiffCommand = rdiffCommand;
            FileOutputStream fout = new FileOutputStream(configFilename);
            ObjectOutputStream oos = new ObjectOutputStream(fout);
            oos.writeObject(config);
            oos.close();
            ui.status("Configuration saved.");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }

    /*
     * ----------------- Private methods -----------------
     */

    /*
     * Loads the configuration from the file config.dat or calls initConfig() if
     * the file does not exist.
     */
    private void loadConfig() {
        File configFile = new File(configFilename);
        // Load configuration from file if it exists...
        if (configFile.isFile()) {
            try {
                FileInputStream fin = new FileInputStream(configFilename);
                ObjectInputStream ois = new ObjectInputStream(fin);
                config = (Config)ois.readObject();
                ois.close();

                targets = config.targets;
                ignores = config.ignores;
                host = config.host;
                port = config.port;
                hostname = config.hostname;
                user = config.user;
                password = config.password;
                autoBackup = config.autoBackup;
                autoBackupTime = config.autoBackupTime;
                autoLogin = config.autoLogin;
                rdiffCommand = config.rdiffCommand;
            }
            catch (Exception e) {
                e.printStackTrace();
            }
        }
        // ...else initialize new configuration
        else {
            initConfig();
        }
```

```
625     }

        /*
         * Initialize a new configuration
         */
630     private void initConfig() {
            config = new Config();

            targets = new ArrayList<String>();
            ignores = new ArrayList<String>();
635         host = "";
            port = 9798;
            hostname = "";
            user = "";
            password = "";
640         autoBackup = false;
            autoBackupTime = "12:00";
            autoLogin = false;
            if (osName.startsWith("Windows")) {
                rdiffCommand = "c:\rdiff\rdiff-backup.bat";
645         }
            else {
                rdiffCommand = "rdiff-backup";
            }

650         // Default values
            targets.add(System.getProperty("user.home"));
            host = "localhost"; // TODO Should be set according to server address
            try {
                java.net.InetAddress localMachine = java.net.InetAddress
655                     .getLocalHost();
                hostname = localMachine.getHostName();
            }
            catch (java.net.UnknownHostException e) {
            }
660         user = System.getProperty("user.name");

            // Save the new configuration
            saveConfig();
        }
665
        /*
         * Generates rdiff-backup command line for backup of Unix systems
         */
        private ArrayList buildBackupCmdUnix() {
670         ArrayList<String[]> cmds = new ArrayList<String[]>();
            ArrayList<String> cmd = new ArrayList<String>();
            cmd.add(rdiffCommand);
            cmd.add("-b");
            cmd.add("-v5");
675
            for (int i = 0; i < targets.size(); i++) {
                cmd.add("--include");
                cmd.add(targets.get(i));
```

```
        }

        for (int i = 0; i < ignores.size(); i++) {
            cmd.add("--exclude");
            cmd.add(ignores.get(i));
        }

        cmd.add("--exclude");
        cmd.add("/");
        cmd.add("/");
        cmd.add(user + "@" + host + "::backup/" + hostname);

        String[] type = {};

        cmds.add(cmd.toArray(type));

        return cmds;
    }

    /*
     * Generates rdiff-backup command line for backup of Windows systems
     */
    private ArrayList buildBackupCmdWin() {
        ArrayList<String[]> cmds = new ArrayList<String[]>();
        ArrayList<String> cmd = new ArrayList<String>();
        Hashtable<String, ArrayList<String>> drives =
            new Hashtable<String, ArrayList<String>>();
        Hashtable<String, ArrayList<String>> ignoredrives =
            new Hashtable<String, ArrayList<String>>();

        // Build list of backup targets for each drive
        for (int i = 0; i < targets.size(); i++) {
            String target = targets.get(i);
            String drive = target.substring(0, 1);
            if (!drives.containsKey(drive)) {
                ArrayList<String> t = new ArrayList<String>();
                t.add(target);
                drives.put(drive, t);
            }
            else {
                ArrayList<String> t = drives.get(drive);
                t.add(target);
            }
        }

        // Build list of backup ignores for each drive
        for (int i = 0; i < ignores.size(); i++) {
            String ignore = ignores.get(i);
            String drive = ignore.substring(0, 1);
            if (drives.containsKey(drive)) {
                if (!ignoredrives.containsKey(drive)) {
                    ArrayList<String> t = new ArrayList<String>();
                    t.add(ignore);
                    ignoredrives.put(drive, t);
```

```
                }
                else {
                    ArrayList<String> t = ignoredrives.get(drive);
                    t.add(ignore);
                }
            }
        }

        // Run through the different drive letters
        for (Enumeration e = drives.keys(); e.hasMoreElements();) {
            String key = ((String)e.nextElement()).toLowerCase();

            cmd.add(rdiffCommand);
            cmd.add("-b");
            cmd.add("-v5");

            ArrayList d = drives.get(key);

            // Include all the targets for current drive letter
            for (int i = 0; i < d.size(); i++) {
                cmd.add("--include");
                cmd.add((String)d.get(i));
            }

            // Exclude all (if any) ignores for current drive letter
            if (ignoredrives.containsKey(key)) {
                ArrayList id = ignoredrives.get(key);

                for (int i = 0; i < id.size(); i++) {
                    cmd.add("--exclude");
                    cmd.add((String)d.get(i));
                }
            }

            cmd.add("--exclude");
            cmd.add(key + ":\\");
            cmd.add(key + ":\\");
            cmd.add(user + "@" + host + "::backup/" + hostname + "/" + key
                    + "_drive_rdiff-backup");

            String[] type = {};
            cmds.add(cmd.toArray(type));
        }

        return cmds;
    }

    /*
     * Generates rdiff-backup command line for restore of Unix systems
     */
    private String[] buildRestoreCmdUnix(String timestamp, String target,
            Boolean force) {
        ArrayList<String> cmd = new ArrayList<String>();
        cmd.add(rdiffCommand);
```

```
            cmd.add("-v5");
            if (force) {
                cmd.add("--force");
790         }
            cmd.add("-r");
            cmd.add(timestamp);
            cmd.add(user + "@" + host + "::backup/" + hostname + target);
            cmd.add(target);
795         String[] type = {};
            return cmd.toArray(type);
        }

        /*
800      * Generates rdiff-backup command line for restore of Windows systems
         */
        private String[] buildRestoreCmdWin(String timestamp, String target,
                Boolean force) {
            ArrayList<String> cmd = new ArrayList<String>();
805
            String drive = target.substring(0, 1);
            String from = target.substring(2).replace("\\", "/");

            cmd.add(rdiffCommand);
810         cmd.add("-v5");
            if (force) {
                cmd.add("--force");
            }
            cmd.add("-r");
815         cmd.add(timestamp);
            cmd.add(user + "@" + host + "::backup/" + hostname + "/" + drive
                    + "_drive_rdiff-backup" + from);
            cmd.add(target);

820         String[] type = {};
            return cmd.toArray(type);
        }

        /*
825      * Returns a SHA-1 hash of the given string.
         */
        private String hash(String s) {
            MessageDigest alg = null;
            byte[] buffer;
830         String result;

            try {
                alg = MessageDigest.getInstance("SHA-1");
            }
835         catch (java.security.NoSuchAlgorithmException e) {
                return null;
            }

            buffer = s.getBytes();
840         alg.reset();
```

```
            alg.update(buffer);
            byte[] digest = alg.digest();

            StringBuffer hexString = new StringBuffer();
            for (int i = 0; i < digest.length; i++) {
                hexString.append(Integer.toHexString(0xFF & digest[i]));
            }
            result = hexString.toString();
            // System.out.println(s + " = " + result);
            return result;
        }

        /*
         * Sends dummy XML-RPC call to server to keep connection alive.
         */
        private void keepAlive() {
            Vector<String> query = new Vector<String>();
            query.add(hash(hash(password) + nextChallenge));
            query.add(user);
            conn.send("keepAlive", query);
        }
    }
```

C.1.2 Cli.java

```
1   package backup.client;

    import java.io.*;
    import java.util.*;
5   import java.text.SimpleDateFormat;

    /**
     * Command line interface for <code>BackupClient</code>, implementing
     * <code>UserInterface</code>.
10   *
     * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
     */
    public class Cli extends Thread implements UserInterface {

15      BackupClient bc;
        boolean running = true;
        InputStream inputstream;
        InputStreamReader inputstreamreader;
        BufferedReader bufferedreader;
20
        /**
         * Constructs a new Cli.
         *
         * @param bc parent BackupClient
25       */
        public Cli(BackupClient bc) {
            this.bc = bc;
```

```
            inputstream = System.in;
30          inputstreamreader = new InputStreamReader(inputstream);
            bufferedreader = new BufferedReader(inputstreamreader);

            start();
        }
35
        /**
         * Start a separate thread for keeping track of keyboard input.
         */
        public void run() {
40          try {
                String string = null;

                System.out.print("Not logged in > ");
                while (running) {
45                  string = bufferedreader.readLine();
                    if (!string.equals("")) {
                        in(string);
                    }
                    String user = "Not logged in";
50                  String hostname = "";
                    if (bc.isLoggedIn()) {
                        user = bc.getUser();
                        hostname = "@" + bc.getHostname();
                    }
55                  if (running) {
                        System.out.print(user + hostname + " > ");
                    }
                }
            }
60          catch (Exception e) {
                e.printStackTrace();
            }
        }

65      /*
         * (non-Javadoc)
         *
         * @see backup.client.UserInterface#status(java.lang.String)
         */
70      public void status(String msg) {
            Date today = new Date();
            SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");
            String time = formatter.format(today);
            System.out.println(time + " " + msg);
75      }

        /*
         * (non-Javadoc)
         *
80       * @see backup.client.UserInterface#warn(java.lang.String)
         */
```

```
    public void warn(String msg) {
        Date today = new Date();
        SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");
        String time = formatter.format(today);
        System.out.println(time + " [!] " + msg);
    }

    /*
     * (non-Javadoc)
     *
     * @see backup.client.UserInterface#processingFile(java.lang.String)
     */
    public void processingFile(String filename) {
        Date today = new Date();
        SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");
        String time = formatter.format(today);
        System.out.println(time + " Processing: " + filename);
    }

    /*
     * Processes one line of user input read from the keyboard.
     */
    private void in(String msg) {
        String[] tmp = msg.split(" ", 2);
        String cmd = tmp[0];
        String allargs = "";
        String[] args = {};

        if (tmp.length > 1) {
            allargs = tmp[1];
            args = allargs.split(" ");
        }

        if (cmd.equals("exit") && args.length == 0) {
            bc.exit();
            running = false;
        }
        else if (cmd.equals("add") && args.length > 0) {
            bc.addTarget(allargs);
            listTargets();
        }
        else if (cmd.equals("remove") && args.length > 0) {
            bc.removeTarget(allargs);
            listTargets();
        }
        else if (cmd.equals("ignore") && args.length > 0) {
            bc.addIgnore(allargs);
            listIgnores();
        }
        else if (cmd.equals("unignore") && args.length > 0) {
            bc.removeIgnore(allargs);
            listIgnores();
        }
        else if (cmd.equals("list")) {
```

```
            if (args.length == 0) {
                listTargets();
                listIgnores();
            }
            else if (args[0].equals("targets")) {
                listTargets();
            }
            else if (args[0].equals("ignores")) {
                listIgnores();
            }
        }
        else if (cmd.equals("files")) {
            String search;
            if (args.length > 0) {
                search = allargs;
            }
            else {
                search = "";
            }
            listFiles(search);
        }
        else if (cmd.equals("set")) {
            if (args.length == 0) {
                status("user = " + bc.getUser());
                status("password = <hidden>");
                status("host = " + bc.getHost());
                status("port = " + bc.getPort());
                status("hostname = " + bc.getHostname());
                if (bc.getAutoBackup()) {
                    status("autobackup = true");
                }
                else {
                    status("autobackup = false");
                }
                status("autobackuptime = " + bc.getAutoBackupTime());
                if (bc.getAutoLogin()) {
                    status("autologin = true");
                }
                else {
                    status("autologin = false");
                }
                status("rdiffcommand = " + bc.getRdiffCommand());
            }
            else if (args.length == 1) {
                if (args[0].equals("user")) {
                    status("user = " + bc.getUser());
                }
                else if (args[0].equals("password")) {
                    status("password = <hidden>");
                }
                else if (args[0].equals("host")) {
                    status("host = " + bc.getHost());
                }
                else if (args[0].equals("port")) {
```

```
                status("port = " + bc.getPort());
            }
            else if (args[0].equals("hostname")) {
                status("hostname = " + bc.getHostname());
            }
            else if (args[0].equals("autobackup")) {
                if (bc.getAutoBackup()) {
                    status("autobackup = true");
                }
                else {
                    status("autobackup = false");
                }
            }
            else if (args[0].equals("autobackuptime")) {
                status("autobackuptime = " + bc.getAutoBackupTime());
            }
            else if (args[0].equals("autologin")) {
                if (bc.getAutoLogin()) {
                    status("autologin = true");
                }
                else {
                    status("autologin = false");
                }
            }
            else if (args[0].equals("rdiffcommand")) {
                status("rdiffcommand = " + bc.getRdiffCommand());
            }

        }
        else {
            if (args[0].equals("user")) {
                String user = allargs.split(" ", 2)[1];
                bc.setUser(user);
                status("user = " + user);
            }
            else if (args[0].equals("password")) {
                String password = allargs.split(" ", 2)[1];
                bc.setPassword(password);
                status("password = " + password);
            }
            else if (args[0].equals("host")) {
                String host = allargs.split(" ", 2)[1];
                bc.setHost(host);
                status("host = " + host);
            }
            else if (args[0].equals("port")) {
                String port = allargs.split(" ", 2)[1];
                bc.setPort(port);
                status("port = " + port);
            }
            else if (args[0].equals("hostname")) {
                String hostname = allargs.split(" ", 2)[1];
                bc.setHostname(hostname);
                status("hostname = " + hostname);
```

```
                }
245             else if (args[0].equals("autobackup")) {
                    String autobackup = allargs.split(" ", 2)[1];
                    if (autobackup.equalsIgnoreCase("true")) {
                        bc.setAutoBackup(true);
                        status("autobackup = true");
250                 }
                    else if (autobackup.equalsIgnoreCase("false")) {
                        bc.setAutoBackup(false);
                        status("autobackup = false");
                    }
255
                }
                else if (args[0].equals("autobackuptime")) {
                    // TODO Check that argument is a valid 24h timestamp
                    String autobackuptime = allargs.split(" ", 2)[1];
260                 bc.setAutoBackupTime(autobackuptime);
                    status("autobackuptime = " + autobackuptime);
                }
                else if (args[0].equals("autologin")) {
                    String autologin = allargs.split(" ", 2)[1];
265                 if (autologin.equalsIgnoreCase("true")) {
                        bc.setAutoLogin(true);
                        status("autologin = true");
                    }
                    else if (autologin.equalsIgnoreCase("false")) {
270                     bc.setAutoLogin(false);
                        status("autologin = false");
                    }

                }
275             else if (args[0].equals("rdiffcommand") && args.length > 0) {
                    String rdiffCommand = allargs.split(" ", 2)[1];
                    bc.setRdiffCommand(rdiffCommand);
                    status("rdiffcommand = " + rdiffCommand);
                }
280         }
        }
        else if (cmd.equals("save") && args.length == 0) {
            bc.saveConfig();
        }
285     else if (cmd.equals("backup")) {
            bc.startBackup();
        }
        else if (cmd.equals("restore") && args.length > 0) {
            String timestamp;
290         String target;
            Boolean force;
            if (args[0].equals("-f")) {
                status("force");
                force = true;
295             if (args.length > 3 && args[1].equals("-t")) {
                    timestamp = args[2];
                    target = allargs.split(" ", 4)[3];
```

```
                }
                else {
300                 timestamp = "now";
                    target = allargs.split(" ", 2)[1];
                }
            }
            else {
305             force = false;
                if (args.length > 2 && args[0].equals("-t")) {
                    timestamp = args[1];
                    target = allargs.split(" ", 3)[2];
                }
310             else {
                    timestamp = "now";
                    target = allargs;
                }
            }
315         if (target.contains("*")) {
                warn("Wildcards not allowed.");
            }
            else {
                bc.startRestore(timestamp, target, force);
320         }
        }
        else if (cmd.equals("login") && args.length == 0) {
            bc.login();
        }
325     else if (cmd.equals("logout") && args.length == 0) {
            bc.logout();
        }
        else if (cmd.equals("help")) {
            usage();
330     }

    }

    /*
335  * Prints help.
     */
    private void usage() {
        status("login                           login on backup server");
        status("add <target>                    add a new target for backup");
340     status("remove <target>                 remove a target for backup");
        status("ignore <target>                 add an exception from backup");
        status("unignore <target>               remove an exception from backup");
        status("list [targets|ignores]          lists targets and/or exceptions");
        status("set [option [value]]            views or sets configuration");
345     status("save                            saves current configuration");
        status("backup                          start manual backup");
        status("restore [-f] [-t ts] <target>   start restore");
        status("                                -f specifies force overwrite, USE");
        status("                                WITH CAUTION!!");
350     status("                                ts is timestamp from rdiff-backup");
        status("                                default ts is \"now\"");
```

```
            status("files [search term]             list files available for restore");
            status("                                search term may contain *");
            status("exit                            exit client");
            status("help                            this text");
        }

        /*
         * Prints a list of current targets.
         */
        private void listTargets() {
            ArrayList t = bc.listTargets();

            if (t != null) {
                status("Targets:");
                for (int i = 0; i < t.size(); i++) {
                    status((String)t.get(i));
                }
            }

        }

        /*
         * Prints a list of current ignores.
         */
        private void listIgnores() {
            ArrayList t = bc.listIgnores();

            if (t != null) {
                status("Ignores:");
                for (int i = 0; i < t.size(); i++) {
                    status((String)t.get(i));
                }
            }
        }

        /*
         * Prints a list of files matching search term and their available
         * timestamps
         */
        private void listFiles(String search) {
            status("Searching...");
            Hashtable result = bc.listFiles(search);
            if (result == null) {
                return;
            }
            else if (result.isEmpty()) {
                warn("No files found matching your search.");
            }
            else {
                ArrayList<String> tmplist = new ArrayList<String>();
                Hashtable<String, Vector<String>> filelist =
                    new Hashtable<String, Vector<String>>();

                // Run through hashtable returned from server
```

```
                for (Enumeration e = result.keys(); e.hasMoreElements();) {
                    String k = (String)e.nextElement();
                    String entry = (String)result.get(k);
                    String timestamp = entry.substring(0, 25);
                    String filename = entry.substring(26);
                    // If we've seen this filename before, just add another
                    // timestamp
                    if (tmplist.contains(filename)) {
                        ((Vector<String>)filelist.get(filename)).add(timestamp);
                    }
                    // Else add it to the list of filenames, and add a new timestamp
                    else {
                        Vector<String> v = new Vector<String>();
                        v.add(timestamp);
                        filelist.put(filename, v);
                        tmplist.add(filename);

                    }
                }

                // Create a alphabetized list of filenames
                String[] filenamelist;
                String[] a = {};
                filenamelist = (tmplist.toArray(a));
                Arrays.sort(filenamelist);

                status("Files matching your search:");

                // Run through the files alphabetically and list their names and
                // timestamps
                for (int i = 0; i < filenamelist.length; i++) {
                    String filename = filenamelist[i];
                    String item = filename + " - ";
                    Vector<String> v = filelist.get(filename);
                    item += (String)v.remove(0);
                    while (!v.isEmpty()) {
                        item += ", " + (String)v.remove(0);
                    }
                    status(item);
                }
            }
        }

    }
```

C.1.3 Config.java

```
1   package backup.client;

    import java.io.Serializable;
    import java.util.ArrayList;
5
```

```
/**
 * Configuration class implementing <code>Serializable</code> for writing to
 * file.
 *
 * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
 */
public class Config implements Serializable {
    private static final long serialVersionUID = -3594491352132439555L;
    public ArrayList<String> targets;
    public ArrayList<String> ignores;
    public String host;
    public int port;
    public String hostname;
    public String user;
    public String password;
    public Boolean autoBackup;
    public String autoBackupTime;
    public Boolean autoLogin;
    public String rdiffCommand;

    /**
     * Constructs a new config with default values.
     */
    public Config() {
        user = "";
        password = "";
        host = "";
        port = 9798;
        hostname = "";
        autoBackup = false;
        autoBackupTime = "12:00";
        autoLogin = false;
        rdiffCommand = "rdiff-backup";

        targets = new ArrayList<String>();
        ignores = new ArrayList<String>();
    }
}
```

C.1.4 Connection.java

```
1  package backup.client;

   import java.net.ConnectException;
   import java.net.MalformedURLException;
5  import java.util.*;
   import org.apache.xmlrpc.*;

   /**
    * Connection for XML-RPC calls to BackupServer.
10  *
    * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
```

```
 */
public class Connection {
    XmlRpcClient server;
    BackupClient bc;

    /**
     * Constructs a new Connection.
     *
     * @param bc parent BackupClient
     * @param host server address
     * @param port server port
     */
    public Connection(BackupClient bc, String host, int port) {
        this.bc = bc;

        String server_url = "http://" + host + ":" + Integer.toString(port);

        try {
            server = new XmlRpcClient(server_url);
        }
        catch (MalformedURLException e) {
            e.printStackTrace();
        }
    }

    /**
     * Executes a XML-RPC call on the server.
     *
     * @param action method to execute on the server
     * @param query parameters
     * @return result from server
     */
    public synchronized Hashtable send(String action, Vector query) {
        try {
            // Call the server, and get our result.
            Hashtable result = (Hashtable)server.execute("BackupServer."
                    + action, query);

            // Record next authentification challenge
            String challenge = (String)result.remove("challenge");
            bc.setChallenge(challenge);

            return result;
        }
        catch (ConnectException e) {
            bc.warn("Connection refused.");
        }
        catch (XmlRpcException e) {
            System.err.println("XML-RPC Fault #" + Integer.toString(e.code)
                    + ": " + e.toString());
        }
        catch (Exception e) {
            System.err.println("Exception: " + e.toString());
        }
```

```
            return null;
        }

    }
```

C.1.5 Gui.java

```
1   package backup.client;

    /**
     * Graphical user interface implementing <code>UserInterface</code>.
5    *
     * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
     */
    public class Gui implements UserInterface {
        private BackupClient bc;
10
        /**
         * Constructs a new Gui.
         *
         * @param bc
15       */
        public Gui(BackupClient bc) {
            this.bc = bc;
            System.out.println("GUI not implemented, exiting.");
            this.bc.exit();
20      }

        /*
         * (non-Javadoc)
         *
25       * @see backup.client.UserInterface#status(java.lang.String)
         */
        public void status(String msg) {
        }

30      /*
         * (non-Javadoc)
         *
         * @see backup.client.UserInterface#warn(java.lang.String)
         */
35      public void warn(String msg) {
        }

        /*
         * (non-Javadoc)
40       *
         * @see backup.client.UserInterface#processingFile(java.lang.String)
         */
        public void processingFile(String filename) {
        }
45  }
```

C.1.6 RdiffWrapper.java

```
1   package backup.client;

    import java.io.*;

5   /**
     * Reader for <code>InputStream</code> from rdiff-backup.
     *
     * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
     */
10  class StreamReader extends Thread {
        private InputStream is;
        private UserInterface ui;

        /**
15       * Constructs a new StreamReader.
         *
         * @param is InputStream to read from
         * @param ui client's UserInterface
         */
20      StreamReader(InputStream is, UserInterface ui) {
            this.is = is;
            this.ui = ui;
        }

25      /**
         * Start a separate thread for reading from the InputStream.
         */
        public void run() {
            try {
30              InputStreamReader isr = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(isr);
                String line = null;
                while ((line = br.readLine()) != null) {
                    if (line.startsWith("Processing changed file")) {
35                      String filename = line.substring(24);
                        if (!filename.equals(".")) {
                            ui.processingFile(filename);
                        }
                    }
40                  else if (line.startsWith("Starting restore of")) {
                        // TODO Parse line and tell UI about it
                        System.out.println(line);
                    }
                    else if (line.endsWith("specify --force to overwrite.")) {
45                      String filename = line.substring(28, line.length() - 46);
                        ui
                                .warn("\""
                                        + filename
                                        + "\" already exists, specify force option to overwrite.\n"
50                                      + "WARNING! If target is a directory its current contents "
                                        + "will be DELETED before restore starts!");
```

```
                }
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }

    }
}

/**
 * * Wrapper for executing rdiff-backup.
 *
 * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
 */
public class RdiffWrapper {
    private Runtime rt;
    private UserInterface ui;

    /**
     * Constructs a new Rdiffwrapper.
     *
     * @param ui UserInterface of parent BackupClient
     */
    public RdiffWrapper(UserInterface ui) {
        this.ui = ui;
    }

    /**
     * Executes rdiff-backup.
     *
     * @param cmd complete command to execute
     */
    public Boolean start(String[] cmd) {
        rt = Runtime.getRuntime();
        try {
            Process proc = rt.exec(cmd);
            StreamReader errors = new StreamReader(proc.getErrorStream(), ui);
            StreamReader output = new StreamReader(proc.getInputStream(), ui);
            errors.start();
            output.start();
            int exitVal = proc.waitFor();
            if (exitVal != 0) {
                return false;
            }
            else {
                return true;
            }
        }
        catch (Exception e) {
            e.printStackTrace();
            return false;
        }

```

```
        }
    }
```

C.1.7 UserInterface.java

```
1   package backup.client;

    /**
     * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
5    */
    public interface UserInterface {
        /**
         * Displays a status message.
         *
10       * @param msg message
         */
        public void status(String msg);

        /**
15       * Displays a warning message.
         *
         * @param msg message
         */
        public void warn(String msg);
20
        /**
         * Display info about file currently being processed by rdiff-backup.
         *
         * @param filename filename
25       */
        public void processingFile(String filename);
    }
```

C.2 Server

C.2.1 BackupServer.java

```
1   package backup.server;

    import java.io.*;
    import java.security.MessageDigest;
5   import java.text.SimpleDateFormat;
    import java.util.*;
    import java.util.Date;
    import java.util.zip.*;
    import java.sql.*;
10
    import org.apache.xmlrpc.*;

    /**
```

```
     * Main server engine.
15   *
     * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
     */
    public class BackupServer extends Thread {
        private WebServer server;
20      private Boolean running;
        private Boolean verbose = false;
        private Cli ui;
        private ArrayList<String> allUsers;
        private ArrayList<String> currentUsers;
25      private Hashtable<String, String> currentChallenges;
        private Hashtable<String, Calendar> lastSeen;
        private Random r;
        private Statement stmt;
        private Connection con;
30      private int port;
        private String dbUrl;

        private final String dbHost = "localhost";
        private final String dbPort = "3306";
35      private final String dbName = "BackupServer";
        private final String dbUser = "BackupServer";
        private final String dbPass = "testrun";

        /**
40       * Constructs a new BackupServer.
         *
         * @param args command line arguments
         */
        public BackupServer(String[] args) {
45          port = 9798;

            ui = new Cli(this);
            allUsers = new ArrayList<String>();
            currentUsers = new ArrayList<String>();
50          currentChallenges = new Hashtable<String, String>();
            lastSeen = new Hashtable<String, Calendar>();

            r = new Random();

55          // Parse command line parameters
            if (args.length > 0) {
                for (int i = 0; i < args.length; i++) {
                    String arg = args[i];
                    if (arg.equals("-v")) {
60                      verbose = true;
                    }
                    else if (arg.equals("-p")) {
                        if (args.length > i + 1) {
                            try {
65                              port = Integer.parseInt(args[i + 1]);
                            }
                            catch (NumberFormatException e) {
```

```
                            ui.status("Invalid port number.");
                            System.exit(1);
                        }
                        i++;
                    }
                    else {
                        ui.status("Please specify port number.");
                        System.exit(1);
                    }
                }
                else {
                    ui.status("Invalid option: " + arg);
                    System.exit(1);
                }
            }
        }

        loadUsers();

        // Start the XML-RPC server
        server = new WebServer(port);
        server.addHandler("BackupServer", new ConnectionListener(this));
        server.start();
        if (verbose) {
            ui.status("Listening for connections on port " + port);
        }

        ui.start();

        // Register the JDBC driver for MySQL
        try {
            Class.forName("com.mysql.jdbc.Driver");
            dbUrl = "jdbc:mysql://" + dbHost + ":" + dbPort + "/" + dbName;
        }
        catch (Exception e) {
            e.printStackTrace();
            System.exit(1);
        }

        running = true;
    }

    /**
     * Main.
     *
     * @param args command line arguments
     */
    public static void main(String[] args) {
        BackupServer bs = new BackupServer(args);
        bs.run();

    }

    /**
```

```
         * Start a separate thread for keeping track of tasks performed at regular
         * intervals.
         */
125      public void run() {
             SimpleDateFormat formatter = new SimpleDateFormat("HH:mm");
             String prev = "";
             int mins = 0;

130          try {
                 while (running) {
                     Thread.sleep(1000);
                     String now = formatter.format(new Date());
                     if (!now.equals(prev)) {
135                      prev = now;
                         mins++;

                         // Check for lost connections every minutes
                         checkConnections();
140
                         // Check for aborted backups
                         if (mins % 30 == 0) {
                             checkBackups();
                         }
145
                         // Reset counter every hour
                         if (mins == 60) {
                             mins = 0;
                         }
150                  }
                 }
             }
             catch (InterruptedException e) {
                 e.printStackTrace();
155          }
         }

         /**
          * Shut down server.
160       */
         public void exit() {
             ui.warn("Server shutting down.");
             server.shutdown();
             running = false;
165      }

         /**
          * Generates a new password challenge.
          *
170       * @param user user requesting challenge
          * @return challenge
          */
         public String getChallenge(String user) {
             // Generate random seed for password challenge
175          String challenge = Long.toString(Math.abs(r.nextLong()), 36);
```

```
            String tmp = getPWHash(user) + challenge;
            currentChallenges.put(user, hash(tmp));

            return challenge;
        }

        /**
         * Check a response against the issued challenge.
         *
         * @param response response from the client
         * @param user user sending the response
         * @return true or false for success
         */
        public Boolean checkResponse(String response, String user) {
            if (!currentChallenges.containsKey(user)) {
                return false;
            }

            if (response.equals(currentChallenges.get(user))) {
                // Any successful response means we can update the clients
                // last seen record.
                try {
                    lastSeen.put(user, Calendar.getInstance());
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
                return true;
            }
            else {
                return false;
            }
        }

        /**
         * Start login procedure for a user.
         *
         * @param user username trying to log in
         * @return challenge for client
         */
        public String startLogin(String user) {
            if (verbose) {
                ui.status("User '" + user + "' trying to log in.");
            }
            return getChallenge(user);
        }

        /**
         * Conclude login procedure for a user.
         *
         * @param user username trying to log in
         * @param response response to challenge
         * @return success
```

```
230         */
        public Boolean finishLogin(String user, String response) {
            // Is this a valid user?
            if (!allUsers.contains(user)) {
                if (verbose) {
235                 ui.status("User '" + user + "' does not exist. Login failed.");
                }
                return false;
            }

240         // Is the user already logged in?
            if (currentUsers.contains(user)) {
                if (verbose) {
                    ui.status("User '" + user
                            + "' is already logged in. Login failed.");
245             }
                return false;
            }

            // Is the response correct?
250         if (checkResponse(response, user)) {
                currentUsers.add(user);
                currentChallenges.remove(user);
                if (verbose) {
                    ui.status("User '" + user + "' logged in.");
255             }
                return true;
            }
            else {
                currentChallenges.remove(user);
260             if (verbose) {
                    ui.status("User '" + user + "' failed login.");
                }
                return false;
            }
265     }

        /**
         * Log out out a user from the server.
         *
270      * @param user username
         */
        public void logout(String user) {
            currentUsers.remove(user);
            if (verbose) {
275             ui.status("User '" + user + "' logged out.");
            }
        }

        /**
280      * Get users currently logged in
         *
         * @return users currently logged in
         */
```

```
    public ArrayList getUsers() {
285     return currentUsers;
    }

    /**
     * Checks for incomplete backups in the specified user's home directory on
290  * the server. Fixes problems according to
     * http://www.nongnu.org/rdiff-backup/FAQ.html#regress_failure
     *
     * @param user username
     */
295 public void checkBackup(String user) {
        ArrayList<String[]> list = findMetaDirs(user);

        // Filename filter for current_mirror*.data
        FilenameFilter currentmirrorFilter = new FilenameFilter() {
300         public boolean accept(File dir, String name) {
                return (name.startsWith("current_mirror") && name
                        .endsWith(".data"));
            }
        };
305
        // Filename filter for mirror_metadata*.snapshot.gz
        FilenameFilter mirrormetadataFilter = new FilenameFilter() {
            public boolean accept(File dir, String name) {
                return (name.startsWith("mirror_metadata") && name
310                     .endsWith(".snapshot.gz"));
            }
        };

        // Run through all the user's meta dirs
315     for (int i = 0; i < list.size(); i++) {
            File dir = new File(list.get(i)[1]);

            if (verbose) {
                ui.status("Checking " + list.get(i)[1]);
320         }

            // Find and delete duplicate current_mirror.*.data files...
            String[] currentmirrorFiles = dir.list(currentmirrorFilter);
            if (currentmirrorFiles == null) {
325             return;
            }
            else if (currentmirrorFiles.length > 1) {
                System.out.println("kom hit");
                Arrays.sort(currentmirrorFiles);
330             File deleteMe = new File(dir.toString() + "/"
                        + currentmirrorFiles[0]);
                if (verbose) {
                    ui.status("Deleted file " + deleteMe);
                }
335             deleteMe.delete();

                // ...and rename corrupted mirror_metadata.*.snapshot.gz
```

```
                String[] mirrormetadataFiles = dir.list(mirrormetadataFilter);

                Arrays.sort(mirrormetadataFiles);
                File renameMe = new File(dir.toString() + "/"
                        + mirrormetadataFiles[mirrormetadataFiles.length - 1]);
                File newName = new File(renameMe.toString().replaceAll(
                        "mirror_metadata", "aborted-metadata"));
                if (verbose) {
                    ui.status("Renamed file " + renameMe + " to " + newName);
                }
                renameMe.renameTo(newName);
            }
        }
    }

    /**
     * Reads file statistics from rdiff-backup logs and inserts records for new
     * and updated files into the database.
     *
     * @param user username
     */
    public void updateDB(String user) {
        int fileCount = 0;
        ArrayList<String[]> metaDirs = findMetaDirs(user);

        // Filename filter for file_statistics*.data.gz
        FilenameFilter filter = new FilenameFilter() {
            public boolean accept(File dir, String name) {
                return (name.startsWith("file_statistics") && name
                        .endsWith(".data.gz"));
            }
        };

        try {
            con = DriverManager.getConnection(dbUrl, dbUser, dbPass);

            for (int i = 0; i < metaDirs.size(); i++) {
                String root = metaDirs.get(i)[0];
                File dir = new File(metaDirs.get(i)[1]);
                String[] statFiles = dir.list(filter);
                Arrays.sort(statFiles);

                // Only process the newest statistics file
                String gzipFilename = statFiles[statFiles.length - 1];
                String time = gzipFilename.substring(16, 41);

                GZIPInputStream gzipInputStream;
                try {
                    gzipInputStream = new GZIPInputStream(new FileInputStream(
                            dir.toString() + "/" + gzipFilename));
                    BufferedReader in = new BufferedReader(
                            new InputStreamReader(gzipInputStream));
                    String line = null;
```

```
                    // Read each line from the log file
                    while ((line = in.readLine()) != null) {
                        int tmp;
                        tmp = line.lastIndexOf(" ") + 1;
                        String incrementSize = line.substring(tmp, line.length());
                        line = line.substring(0, tmp - 1);
                        tmp = line.lastIndexOf(" ") + 1;
                        String mirrorSize = line.substring(tmp, line.length());
                        line = line.substring(0, tmp - 1);
                        tmp = line.lastIndexOf(" ") + 1;
                        String sourceSize = line.substring(tmp, line.length());
                        line = line.substring(0, tmp - 1);
                        tmp = line.lastIndexOf(" ") + 1;
                        String changed = line.substring(tmp, line.length());
                        String filename = line.substring(0, tmp - 1);

                        // Changed or added. Not deleted.
                        if (changed.equals("1") && !sourceSize.equals("NA")
                                && !filename.equals(".")) {
                            insertDB(root + filename, user, time);
                            fileCount++;
                        }
                    }
                }
                catch (FileNotFoundException e) {
                    e.printStackTrace();
                }
                catch (IOException e) {
                    e.printStackTrace();
                }

            }
            con.close();
        }
        catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

        ui.status("Updated/added " + String.valueOf(fileCount)
                + " files/directories for user '" + user + "'.");
    }

    /**
     * Returns a list of a user's files matching the specified search term, or
     * all files if search term is blank.
     *
     * @param user username
     * @param search search term
     * @return list of files with timestamps
     */
    public Hashtable<String, String> getFiles(String user, String search) {
        Hashtable<String, String> result = new Hashtable<String, String>();
        try {
```

```
            con = DriverManager.getConnection(dbUrl, dbUser, dbPass);
            ResultSet rs;
            stmt = con.createStatement();
            String query = "SELECT * FROM `files` WHERE `user` LIKE '" + user
                    + "'";

            if (!search.equals("")) {
                query += " AND `filename` LIKE '" + search + "'";
            }
            query += ";";

            rs = stmt.executeQuery(query);

            while (rs.next()) {
                String id = String.valueOf(rs.getInt("id"));
                String timestamp = rs.getString("timestamp");
                String filename = rs.getString("filename");

                result.put(id, timestamp + "_" + filename);
            }
            con.close();
        }
        catch (SQLException e) {
            e.printStackTrace();
        }

        return result;
    }

    /*
     * ----------------- Private methods -----------------
     */

    /*
     * Returns an ArrayList of all the meta data directories and the root
     * directory they correspond to on the client.
     */
    private ArrayList<String[]> findMetaDirs(String user) {
        File root = new File("/home/" + user + "/backup");
        ArrayList<String[]> metadirs = new ArrayList<String[]>();

        if (!root.isDirectory()) {
            return metadirs;
        }

        String[] dirs = root.list();

        for (String dirname : dirs) {
            File tmp = new File(root.toString() + "/" + dirname
                    + "/rdiff-backup-data/");
            if (tmp.isDirectory()) {
                String[] mdir = { "/", tmp.toString() };
                metadirs.add(mdir);
            }
```

```
500
                // Need to check deeper in case of Windows backup
                File dir = new File(root.toString() + "/" + dirname);
                String[] subdirs = dir.list();
                for (String subdirname : subdirs) {
505                 if (subdirname.endsWith("_drive_rdiff-backup")) {
                        File subdir = new File(root.toString() + dirname + "/"
                                + subdirname + "/rdiff-backup-data/");
                        if (subdir.isDirectory()) {
                            String[] mdir = {
510                                 subdirname.substring(0, 2).toUpperCase()
                                            + ":\\", tmp.toString() };
                            metadirs.add(mdir);
                        }
                    }
515             }
            }

            return metadirs;
        }
520
        /*
         * Returns a SHA-1 hash of the given string.
         */
        private String hash(String s) {
525         MessageDigest alg = null;
            byte[] buffer;
            String result;

            try {
530             alg = MessageDigest.getInstance("SHA-1");
            }
            catch (java.security.NoSuchAlgorithmException e) {
                return null;
            }
535
            buffer = s.getBytes();
            alg.reset();
            alg.update(buffer);
            byte[] digest = alg.digest();
540
            StringBuffer hexString = new StringBuffer();
            for (int i = 0; i < digest.length; i++) {
                hexString.append(Integer.toHexString(0xFF & digest[i]));
            }
545         result = hexString.toString();
            // System.out.println(s + " = " + result);
            return result;
        }

550     /*
         * Checks whether all logged in clients have reported to the server during
         * the last three minutes.
         */
```

```
        private void checkConnections() {
555         for (int i = 0; i < currentUsers.size(); i++) {
                String user = (String) currentUsers.get(i);
                Long then = ((Calendar) lastSeen.get(user)).getTimeInMillis() / 1000;
                Long now = Calendar.getInstance().getTimeInMillis() / 1000;
                Long diff = now - then;
560
                if (diff > 180) {
                    if (verbose) {
                        ui.status("User '" + user + "' timed out. (Last seen "
                                + diff.toString() + " seconds ago)");
565                 }
                    logout(user);
                }
            }
        }
570
        /*
         * Run through all the users, calling checkBackup on each one.
         */
        private void checkBackups() {
575         for (int i = 0; i < allUsers.size(); i++) {
                checkBackup(allUsers.get(i));
            }
        }

580     /*
         * Inserts an entry into the file database.
         */
        private void insertDB(String filename, String user, String timestamp) {
            String date = timestamp.substring(0, 16).replace("T", " ");
585         try {
                stmt = con.createStatement();
                String query = "INSERT INTO `files` (`filename`, `user`, `date`, `timestamp`) "
                        + "VALUES ('"
                        + filename
590                     + "', '"
                        + user
                        + "', '"
                        + date
                        + "', '" + timestamp + "');";
595             stmt.executeUpdate(query);
            }
            catch (SQLException e) {
                e.printStackTrace();
            }
600     }

        /*
         * Returns a SHA-1 hash of the user's password. Should be customized
         * according to the user/password backend.
605      */
        private String getPWHash(String user) {
            if (!allUsers.contains(user)) {
```

```
            return hash("notfound");
        }
        return hash("test");
        // TODO Return real password hashes
    }

    /*
     * Load a list of all registered users. Should be customized according to
     * the user/password backend.
     */
    private void loadUsers() {
        allUsers.add("mauset");
        allUsers.add("test");
        // TODO Real loading
    }
}
```

C.2.2 Cli.java

```
1   package backup.server;

    import java.io.BufferedReader;
    import java.io.InputStream;
5   import java.io.InputStreamReader;
    import java.util.*;
    import java.text.SimpleDateFormat;

    /**
10   * Command line interface for <code>BackupServer</code>
     *
     * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
     */
    public class Cli extends Thread {
15
        private BackupServer bs;
        private boolean running = true;
        private String last = "";

20      /**
         * Constructs a new Cli.
         *
         * @param bs parent <code>BackupServer</code>
         */
25      public Cli(BackupServer bs) {
            this.bs = bs;
        }

        /**
30       * Start a separate thread for keeping track of keyboard input.
         */
        public void run() {
            try {
```

```
            InputStream inputstream = System.in;
            InputStreamReader inputstreamreader = new InputStreamReader(
                    inputstream);
            BufferedReader bufferedreader = new BufferedReader(
                    inputstreamreader);
            String string = null;

            System.out.print("> ");
            last = "prompt";
            while (running) {
                string = bufferedreader.readLine();
                last = "";
                if (!string.equals("")) {
                    in(string);
                }
                if (running) {
                    System.out.print("> ");
                    last = "prompt";
                }
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Prints a status message.
     *
     * @param msg message
     */
    public void status(String msg) {
        if (last.equals("prompt")) {
            System.out.println("");
            last = "msg";
        }
        Date today = new Date();
        SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");
        String time = formatter.format(today);
        System.out.println(time + " " + msg);
    }

    /**
     * Prints a warning message.
     *
     * @param msg message
     */
    public void warn(String msg) {
        if (last.equals("prompt")) {
            System.out.println("");
            last = "msg";
        }
        Date today = new Date();
        SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");
```

```
            String time = formatter.format(today);
            System.out.println(time + " [!] " + msg);
90      }

        /*
         * Processes one line of user input read from the keyboard.
         */
95      private void in(String msg) {
            String[] tmp = msg.split(" ", 2);
            String cmd = tmp[0];
            String allargs = "";
            String[] args = {};
100
            if (tmp.length > 1) {
                allargs = tmp[1];
                args = allargs.split(" ");
            }
105
            if (cmd.equals("exit") && args.length == 0) {
                bs.exit();
                running = false;
            }
110         else if (cmd.equals("list")) {
                listUsers();
            }
            else if (cmd.equals("help")) {
                usage();
115         }
        }

        /*
         * Prints help.
120      */
        private void usage() {
            status("list     list logged in users");
            status("exit     shut down server");
            status("help     this text");
125     }

        /*
         * Prints a list of users currently logged in.
         */
130     private void listUsers() {
            status("Current users:");

            ArrayList userlist = bs.getUsers();

135         for (int i = 0; i < userlist.size(); i++) {
                status((String)userlist.get(i));
            }
        }

140 }
```

C.2.3 ConnectionListener.java

```
1   package backup.server;

    import java.util.*;

5   /**
     * Handler for XML-RPC calls from <code>BackupClient</code>.
     *
     * @author <a href="mailto:mauset@idi.ntnu.no">Tore Mauset</a>
     */
10  public class ConnectionListener {
        BackupServer bs;

        /**
         * Constructs a new ConnectionListener.
15       *
         * @param bs parent <code>BackupServer</code>
         */
        public ConnectionListener(BackupServer bs) {
            this.bs = bs;
20      }

        /**
         * Start login procedure for specified user.
         *
25       * @param user username
         * @return hashtable containing new password challenge
         */
        public Hashtable startLogin(String user) {
            String challenge = bs.startLogin(user);
30          Hashtable<String, String> result = new Hashtable<String, String>();
            result.put("challenge", challenge);
            return result;
        }

35      /**
         * Finsh login procedure for specified user.
         *
         * @param response response to previous password challenge
         * @param user username
40       * @return hashtable containing success and new password challenge, or
         *         failure.
         */
        public Hashtable finishLogin(String response, String user) {
            Hashtable<String, String> result = new Hashtable<String, String>();
45          Boolean success = bs.finishLogin(user, response);
            if (success) {
                String challenge = bs.getChallenge(user);
                result.put("challenge", challenge);
            }
50          result.put("success", success.toString());
            return result;
```

```
    }

    /**
     * Log out specified user from server.
     *
     * @param response response to previous password challenge
     * @param user username
     * @return empty hashtable
     */
    public Hashtable logout(String response, String user) {
        if (bs.checkResponse(response, user)) {
            bs.logout(user);
        }
        return new Hashtable();
    }

    /**
     * Dummy call to update BackupServer's Last Seen record for the specified
     * user.
     *
     * @param response response to previous password challenge
     * @param user username
     * @return hashtable containing new password challenge
     */
    public Hashtable keepAlive(String response, String user) {
        // checkResponse will update the last seen record if the challenge
        // response is correct
        bs.checkResponse(response, user);
        String challenge = bs.getChallenge(user);
        Hashtable<String, String> result = new Hashtable<String, String>();
        result.put("challenge", challenge);
        return result;
    }

    /**
     * Tells BackupServer to check meta data of specified user.
     *
     * @param response response to previous password challenge
     * @param user username
     * @return hashtable containing new password challenge
     */
    public Hashtable checkBackup(String response, String user) {
        Hashtable<String, String> result = new Hashtable<String, String>();
        if (bs.checkResponse(response, user)) {
            bs.checkBackup(user);
            String challenge = bs.getChallenge(user);
            result.put("challenge", challenge);
        }
        return result;
    }

    /**
     * Tells BackupServer to update database for specified user.
     *
```

```
         * @param response response to previous password challenge
         * @param user username
         * @return hashtable containing new password challenge
         */
        public Hashtable updateDB(String response, String user) {
            Hashtable<String, String> result = new Hashtable<String, String>();
            if (bs.checkResponse(response, user)) {
                bs.updateDB(user);
                String challenge = bs.getChallenge(user);
                result.put("challenge", challenge);
            }
            return result;
        }

        /**
         * Requests a list of files matching the search term from BackupServer.
         *
         * @param response response to previous password challenge
         * @param user username
         * @param search search term
         * @return hashtable containing list of files and new password challenge
         */
        public Hashtable listFiles(String response, String user, String search) {
            Hashtable<String, String> result = new Hashtable<String, String>();
            if (bs.checkResponse(response, user)) {
                result = bs.getFiles(user, search);
                String challenge = bs.getChallenge(user);
                result.put("challenge", challenge);
            }
            return result;
        }

    }
```

C.2.4 BackupServer.sql

Dette er SQL-kode for å opprette en ny, tom database for serveren.

```
DROP TABLE IF EXISTS ‘files‘;
CREATE TABLE ‘files‘ (
  ‘id‘ bigint(20) NOT NULL auto_increment,
  ‘filename‘ text NOT NULL,
  ‘user‘ varchar(255) NOT NULL default ’’,
  ‘date‘ datetime NOT NULL default ’0000-00-00 00:00:00’,
  ‘timestamp‘ varchar(255) NOT NULL default ’’,
  PRIMARY KEY  (‘id‘)
) TYPE=MyISAM;
```