NTNU
Det skapende universitet

Mulig løsning for søk og indeksering i distribuerte anonymiserte systemer

Distribuerte tjenere

Morten Tvenning

Master i informatikk
Oppgaven levert: Juni 2006
Hovedveileder: Trond Aalberg, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Sammendrag

Denne avhandlingen gjør rede for hvordan søk og indeksering kan gjøres i anonymiserte distribuerte nettverk. Anonyme nettverk er interessante for den digitale tidsalder for å beskytte privatinformasjon og for å unngå sensurering. Anonymiseringen av brukere har gjort tidligere søk og indekseringsmetoder ubrukbare. I denne avhandlingen foreslås en løsning med distribuerte indekseringstjenere. Tjenerne vil fortone seg som ett lag over nettverket. Dette laget gir funksjonene indeksering og distribuerte søk, hvor tjenestene alltid er tilgjengelige for nettverket og indekser er persistent lagret.

Hvor høy kvaliteten på løsningen er, blir vist av design, simulering ved bruk av prototyp, tester og evaluering.

Stadig mer av informasjonen vi omgir oss med er i digital form. For å gjøre informasjon tilgjengelig er søkefunksjonen vital. Avhandlingen viser et nytt design for søkefunksjonalitet til anonymiserte nettverk.

Forord

Jeg vil gjerne si takk til min veileder Trond Aalberg for hyggelige veiledningstimer og svært presise og konstruktive tips og råd.

I tillegg vil jeg takke Tor Åge Ballo for hoderenskende gåturer, oppmuntring til skolearbeid og topp selskap over pulten med snedige kommentarer om førstepersons oppførsel. I tillegg vil jeg takke hele kontor 042 for konstruktive kaffeslaberas. Vil også ta anledningen til å takke Inger Tvenning for korrekturlesning. Christine Karlsen for uoppfordret optimisme, uventet forståelse og oppmunting under masteroppgaven.

Tusen takk!

Trondheim 01.06.2006

Morten Tvenning

Innhold

Figurer

Kapittel 1

Innledning

Giving me a new idea is like handing a cretin a loaded gun, but I do thank you anyhow, bang, bang.

Phillip K. Dick [32]

Distribuert fildeling har de siste 5 årene blitt en allment brukt tjeneste. Ikke lenger er det bare spesielt interesserte som deler filer igjennom "peer-to-peer"- systemer. Mer og mer av informasjonen vi omgir oss med er i digital form og fildelingsnettverk forenkler deling av informasjon mellom personer.

Eksisterende systemer tilfredsstiller i varierende grad brukernes krav til informasjonsdelingstjenester. Sensurering, personinformasjonsinnsamling og innsamling av informasjon for rettsforfølgelse er blitt synlig for brukeren. Anonyme informasjonsdelingstjenester er katalysert av behovet for å unngå overvåkning på Internett.

Flere anonymiserte system har allerede sett dagens lys, men hvor stor grad av funksjonalitet må til for at de skal bli allment brukt? Enkelhet i bruk, kvalitet på funksjonalitet og godt brukergrensesnitt er generelle kvalitetskrav til IT-systemer og må også følges av fremtidens fildelingsnettverk. Spesielt funksjonaliteten på informasjonslokalisering knyttet opp til brukerens krav er viktig i informasjonsdelingsnettverk. Fildeling har fokusert på lokalisering av spesielle filer, men deling av informasjon setter krav til henting av riktig dokument med størst mulig relevans for brukeren.

Ett av de store spørsmålene som følger større bruk av Internett er hvordan personer kan ta med seg sin privatsfære ut på Internett. Her kan ett av svarene være distribuerte anonyme informasjonsdelingstjenester.

1.1 Introduksjon til oppgaven

Hovedtema for denne avhandlingen er hvordan det er mulig å få til effektive søk i distribuerte anonyme informasjonsdelingstjenester. Måten informasjonen distribueres på, hvordan systemet anonymiseres og strukturen i det underliggende systemet har stor påvirkning på hvordan søke- og indekseringstjenester bør implementeres. Avhandlingen presentere distribuerte systemer, problemer knyttet til søk i disse, hvorfor det er grunn til å gå over til anonyme systemer og gi et forslag for å gi god søke og indekseringsfunksjonalitet i persistente anonymiserte distribuerte systemer.

1.1.1 Problemformulering

Hvordan lage en effektiv indekserings- og søketjeneste for persistente distribuerte anonyme informasjonsdelingssystemer?

1.1.2 Mål for oppgaven

Vise hvordan indeksering og søkefunkefunksjoner kan implementeres for distribuerte anonyme informasjonsdelingstjenester. Lage en løsning for indeksering og søking i anonymiserte distribuerte nettverk. Løsningen skal ivareta anonymiteten i nettverket, effektivisere søkefunksjonaliteten og forbedre søkekvaliteten.

Delmål for oppgaven

Utenom hovedtema er delmålene for avhandlingen:

- Peke på problem ved eksisterende informasjonsgjenfinningsmetoder innenfor anonyme "peer-to-peer"-nettverk.
- Peke på problemene ved eksisterende "peer-to-peer"-systemer.
- Diskutere de fundamentale forskjellene mellom sentraliserte søketjenester og distribuerte søketjenester.
- Utvikle enkle pseudokoder og algoritmer med forslag for å løse problemene. Disse skal vise hovedfunksjonalitet til løsningsforslaget.
- Implementere en prototyp for å simulere nettverket og vise funksjonalitet.
- Utføre tester i simuleringen.
- Evaluere testresultatene.

1.1.3 Avgrensning for oppgaven

Avhandlingen begrenser seg til søk på innhold i tekstdokumenter. Det finnes allerede løsninger for søk direkte på filnavn og dette vil ikke bli behandlet i oppgaven.

Routing begrenses til routing internt i nettverket. Overføring mellom noder ved bruk av allerede eksisterende internettfunksjonalitet blir ikke behandlet i avhandlingen.

Simulerer løsningen i stede for å implementere en fungerende løsning.

1.2 Fremgangsmåte

Fremgangsmåten kan karakteriseres som eksperimentell utvikling. I avhandlingen er det først behandlet tidligere distribuerte systemer. Så fulgte spesifisering og design av løsning og programmering av prototyp. Deretter er prototypen bruk til tester og resultatene evaluert.

Eksisterende løsninger: Avhandlingen bygger på grundig forståelse av allerede eksisterende systemer, utvelgelse av de mest avanserte og passende systemene.

Spesifisere løsning: Design av løsning er basert i kunnskap om eksisterende systemer og tankesett. Spesifiseringsprosessen bestod av kravspesifisering og design av en løsning som oppfyller kravene.

Simulering ved hjelp av prototyp: En prototyp er programmert i Java for å simulering og testing av løsningen.

Test og evaluering: Testene på prototypen produserte store mengder data om løsningen. Disse dataene er så evaluert for å trekke ut informasjon om hvor høy kvalitet løsningen har. I tillegg har dataene gitt informasjon om spesifikt hvor godt løsningen skalerer, effektivitet og kvalitet på de enkelte funksjonene.

Fremtidig arbeid: I løpet av testing og evaluering har flere aspekter dukket opp kravspesifikasjonene eller prototypen tar hensyn til. Disse er spesifisert og fremtidig arbeid er foreslått.

Utviklingen i avhandlingen har vært iterativ. Både prototyp og kravspesifikasjon har blitt utviklet i bolker for å tilpasse løsningen. Prototypen er modulbasert og hver modul har blitt lagt til etter verifisering av kvaliteten til nylagede moduler. Testene ble omgjort for å gi bedre data og mer presise observasjoner.

1.3 Relevant forskningslitteratur

De mest relevante kildene for denne oppgaven er to artikler publisert av Ian Clarke et. al. [29, 28] og beskriver arkitektur og implementasjon av et anonymisert distribuert informasjonsdelingsnettverk. Nyskapningen i artiklene reflekteres i at [29] var den mest siterte artikkelen på Citeseer.ist.edu [5] i 2002.

Kapittel 2

Bakgrunn og Teori

> *That truth-is-stranger-than-fiction factor keeps getting jacked up on us on a fairly regular, maybe even exponential, basis. I think that's something peculiar to our time. I don't think our grandparents had to live with that.*
>
> William Gibson[1]

Datanettverk gir muligheten for brukere å dele informasjon seg imellom. Nettverkene kan abstraheres til grafer av noder med bindinger i mellom. Distribuerte nettverk har blitt en del av hverdagen for folk flest på grunn av fildelingstjenester, søkemotorer på Internett og "chatte"-programmer.

2.1 Distribuerte systemer - P2P

Definisjonen for distribuerte systemer er et system hvor det finnes to eller flere applikasjonsmoduler lokalisert på forskjellige datamaskiner. Applikasjonene jobber sammen, koblet sammen over et kommunikasjonsnettverk [52]. Kommunikasjonsnettverket til P2P-systemer bygger oftest på TCP-IP [40]. Transport and Communication Protocoll (TCP) tar seg av overføring mellom datamaskinene. TCP garanterer for at filene kommer frem hele og benytter seg igjen av Internet Protocoll (IP). IP tar seg av oppkobling og routing av datapakker mellom datamaskiner som er koblet på Internett. Det et P2P system skal gjøre er å stå for sammenkoblingen av datamaskiner (omtales ofte som klienter eller servants i ett P2P nettverk [47]). I denne avhandlingen brukes noderfor å omtale datamaskiner i et P2P nettverk. Node referer mer spesifikt til datamaskinens deltagelse i nettverket. Sammenkoblingen av noder kan gjøres med flere nettverkstopologier i bunn.

[1] Hentet fra dokumentaren 'No Maps for These Territories' `http://www.nomaps.com/`

Matthias Bender et. al. [23] starter sin artikkel med å si at Peer-To-Peer (P2P) systemer gir enormt potensial for kraftige nettverk i form av skaleringsgrad, feiltoleranse, effektivitet og dynamikk. Den beste indikasjonen på hvor bra en tjeneste er kan ofte være hvor mange brukere tjenesten har [36]. Napster [14] er ett kroneksempel på hvor intuitivt god en P2P tjeneste kan være. P2P teknologi er ikke definert likt over alt, forskjellene i definisjonen er gjerne hvor "bred" definisjonen er. Det finnes ingen enighet om akkurat hva *er*og *ikke er* P2P [21]. Det som kan sies med sikkerhet om P2P er at det er distribuerte nettverk hvor klienter lager en direkte eller virtuelt direkte kobling mellom hverandre for å overføre dokumenter[2]. For noder i ett P2P finnes det ikke noe skille mellom klient og tjener. Alle nodene utfører samtidig funksjonene som tjener og klient.

Klienter i ett distribuert system innehar ofte mange roller. Klientene er samtidig publiserer, holder og konsumenter. I nettverk hvor pakker sendes gjennom nettverket og ikke rett fra node til node kan klienter ha rollen som overfører. Operasjonene nodene utfører er å sende ut søk, ta i mot søk og prosessere disse, dele filer og sende videre beskjeder/meldinger [47].

Bo Leuf [40] definerer tre forskjellige tankesett som ligger i bunn for P2P systemer: Atomisk P2P, Brukersentrert P2P og Datasentrert P2P.

2.1.1 Atomistic P2P - Fullt desentraliserte systemer

Atomisk P2P representeres av et sett av noder som både er klient og tjener. Hver node er typisk i kontakt med flere andre, men direkte eller virtuell kontakt kan opprettes mellom enhver node og en annen. Hver node har full kontroll over sine egne ressurser og tilkoblinger. Den atomiske P2P modellen har et fundamentalt problem; hvordan finne andre noder og kobler seg til? Dette problemet kan løses ved å ha en eller flere kjente tilkoblingsnoder eller at applikasjonen sender *her er jeg* meldinger til nettet og venter. Etter at en node er koblet til en annen node kan den lytte til aktivitet på nettet eller aktivt søke etter flere noder for å lage en større nodeliste (routingtabell [3]).

Senere versjoner av atomistiske P2P nettverk har beveget seg fra helt flat nodestruktur til å gi ett litt større ansvar til ressursnoder. Noder med høy båndbredde og kapasitet blir gitt oppgave som "supernode". Pålitelige og betrodde supernoder kan utføre tjenester som oppbevaringssted, primære

[2]I denne oppgaven vil dokumenter være en samlingsbetegnelse for alle typer filer publisert i P2P nettverk. Tekst, bilde, lyd og film filer blir omtalt som dokumenter i resten av oppgaven.

[3]Routingtabell er nodens liste over kontakter i nettverket. Routingtabellen kan inneholde flere typer informasjon. Disse er: lenker til noder(IP-adresser), informasjon om nodene(for eksempel hvilke nøkler de lagrer) og informasjon om hvor pakker skal sendes.

tilkoblingsnoder og søkesentraler [40]. Supernodene vil fungere som ett slags ryggrad og effektivisering for nettverket slik som backbonetjenere [4] gjør det på Internett. Kazaa er ett eksempel på en P2P applikasjon med en slik tankegang i bunn [12].

Atomistiske nettverk kan karakteriseres som *søk og oppdag*. Søkemekanismen bygger i hovedsak på strukturen og protokollen nettverket benytter seg av. Siden det ikke finnes noen sentral autoritær tjener må søk sendes direkte til tjenerdelen i den enkelte node. Hver node som mottar ett søk, søker igjennom sin egen indeks. Konvensjoner om hvordan søket skal gjennomføres ligger i programvaren eller selve søket. Denne søkemetoden kalles Random Walk. Om Random Walk algoritmen finner det den leter etter kan gi noden beskjed om å sende filen til den som søker, eller opprette en direkte tilkobling til den som søker. Siden det ikke finnes en sentral tjener vil ett hvert søk være ufullstendig. I tillegg er søk lagt under problemet med at filer er navngitt forskjellig og skrivefeil oppstår. Ett ekstra problem er at søket er ukoordinert og ufiltrert noe som fører til duplikater i søkeresultatet.

Klientene kan ha den samme eller forskjellige metoder for å prosessere spørringer. Det eneste alle klientene må ha er den samme forståelsen av protokollen for spørringer. Derfor er det ikke sikkert at den samme spørringen blir behandlet på samme måte i hver node [40].

Det finnes flere måter å effektivisere et atomisk nettverk. Dokumenter som blir spurt etter kan repliseres i de nodene som overfører dokumentet. Denne fremgangsmåten vil gjøre populære filer mer tilgjengelige i nettverket. Dokumenter replikeres[5] med hensikt av noden som publiserer de. Dokumenter kan også distribueres i fragmenter i mange noder, men denne fremgangsmetoden antar at det finnes en mulighet for å lokalisere fragmentene enkelt. Distribuerte hashtabeller[6] kan gjøre søket etter filene enklere, antatt at filene er lokalisert der hvor de er indeksert.

2.1.2 Brukersentrert P2P - Semisentraliserte systemer

Brukersentrert P2P er utvidelse av den atomiske P2P modellen. Nettverket blir semisentralisert av en sentral katalog tjener. Katalogtjeneren registrer hvilke brukere som logger seg på og gjør denne informasjonen kjent for

[4]Backbone tjenere er en del av backbone nettverket på Internett. Dette nettverket er på toppen av det hierarkiske datamaskinnettverket og kobler sammen de lavere delene av hierarkiet.

[5]Replikeres brukes i denne avhandlingen som en oversettelse av det engelske ord Replication. Det vil si at noe kopieres kopieres utover i nettverket.

[6]Hashtabell en er en datastruktur for å assosiere nøkler med vedier. I denne avhandlingen er vediene dokumenter.

alle nodene i nettverket. Statusen til nodene oppdateres periodisk av katalogtjeneren. Nodene bruker katalogtjeneren for å lete etter andre noder og innholdet på disse nodene. Når en node med den aktuelle informasjonen eller filen er lokalisert hjelper katalogtjeneren til å koble de to nodene direkte sammen. Brukersentrert P2P har vist seg å være den mest populære arkitekturen. Napster [14] er ett kjent eksempel på et brukersentrert P2P system. MSN, Miriblis ICQ og AOL Instant Messenging er andre populære systemer bygget på brukersentrert arkitektur [40].

Den sentrale katalogtjeneren er også objekt for noen problematiske forhold. For anonymiserte systemer er sentrale katalogtjenere problematisk. En sentral tjener er enkelt å identifisere og kan være *single point of failure*. Dette siden hele nettverket er avhengig av tjeneren. Atomiske nettverk har ikke en enkel plass i systemet alle nodene er avhengig av.

I tillegg kan brukeroppførsel kan overvåkes og generere personlige profiler. Salg av slike profiler kan være veldig interessant for annonsører og reklamebransjen [35]. Andre organisasjoner kan også overvåke nettverket til for eksempel personinformasjonsdatabaser eller for juridiske søksmål.

2.1.3 Datasentrert P2P - Sentraliserte systemer

Datasentrert P2P er svært lik brukersentrert P2P. Forskjellen er at datasentrert P2P bruker en sentral tjener som opprettholder en indeks over alle ressurser og ikke individuelle brukere. Klienter registrerer seg hos tjeneren og laster opp eller gjør tilgjengelig en liste over ressursene tilgjengelig i klienten. I motsetning til atomistiske og brukersentrerte P2P systemer er bruken av slike systemer rettet mot bedrifter. Siden det finnes en sentral tjener er det mulig å innføre regler og restriksjoner på bruk og aksessering. Sikkerhetsproblemer må på en helt annen måte reflekteres og implementeres i systemet da privat eller sensitiv informasjon befinner seg i klientene [40].

Et eksempel på et åpen kildekode system utviklet etter datasentrert P2P tankegang er DC++ [9]. Klienter i dette nettverket kobler seg til en tjener. Tjeneren støtter bare et visst sett med versjoner av applikasjonen. De siste versjonene har svært utstrakt funksjonalitet. Klienter lager en hash[7] av alle filer og laster opp hash og indeks av delte filer til en sentral tjener. Dette gir svært gode søkeresultat og muligheten for automatisk nedlasting, automatisk nedlasting fra flere klienter samtidig og fortsettelse av avbrutte nedlastninger.

[7]Hash er en nøkkelverdi generert av en matematisk funksjon. Funksjonen bruker dokumentet som "input".

2.1.4 Begrensninger ved P2P systemer

D. Verma [52] tar for seg begrensningene i P2P systemer. De karakteristikkene til P2P systemer som gjør fildelingstjenester populære er også de samme karakteristikkene som begrenser bruken av systemene.

- **Versjonering og Siste informasjon**. P2P systemer fungerer svært godt til ferdig og statisk informasjon, men gjør en dårlig jobb når det gjelder å forvalte forskjellige versjoner av et dokument og bytte ut et gammelt dokument med en nyere versjon. Gamle dokumenter blir liggende i systemet selv om mer oppdaterte dokumenter er tilgjengelig.

- **Ukorrekte dokumenter**. I P2P systemer kan ondsinnede eller spøkende brukere legge inn dårlig eller ukorrekt informasjon. Brukere med samme hensikt kan også annotere eller notere i filer med samme resultat. En vei rundt dette kan være for designere av systemet å kreve at publiserer signerer med en digital signatur på filer de deler ut.

- **Lovlighet**. Selv om P2P systemer er laget for å dele legitimt innhold er det ofte enkelt å dele kopibeskyttet innhold også i disse nettverkene. Ett eksempel på dette var da Metallica saksøkte Napster noe som resulterte i at Napster måtte fjerne innhold Metallica har opphavsrettighetene til. Brukere reagerte med å forandre filnavnene og forandre på filen slik at den binær signaturen ble litt annerledes [55]. Retningslinjer for lovlig delbart materiale på et nettverk kan være mulig å gjennomføre for små private nettverk, men umulig å gjennomføre i åpne nettverk.

- **Sikkerhet**. P2P applikasjoner gjør det mulig for alle å dele hva de vil. Hvis de nedlastede filene inneholder robotprogrammer, Trojanske hester eller virus kan datamaskinen bli infisert. En kjørbar fil på 200 megabyte lastet ned fra P2P nettverk kan inneholde et svært stort virus. Spredningen av viruset vil også fortsette fra alle brukere uten gode nok sikkerhetstiltak på maskinen.

2.1.5 Likheter og forskjeller i tradisjonelle distribuerte systemer

Tradisjonelle informasjonsgjenfinningssystemer bygger ofte på en sentralisert arkitektur. Forskjellene fra tradisjonelle systemer og sentraliserte P2P systemer er ikke stor. Begge systemene lagrer informasjonen sentralt og gjør denne søkbar gjennom ett grensesnitt. Hovedforskjellen fra P2P-systemer og tradisjonelle systemer er strukturen på informasjonen og de delte dokumentene.

P2P-systmer har i stor grad fokusert på å koble sammen enkeltstående datamaskiner for å dele filer imellom disse. Ofte er det film og musikkfiler som deles. Biblioteksystemer fokuserer ofte på å søke i strukturert informasjon laget for å indekseres. Metadata for film og musikkfiler er ofte mangelfull og ofte er søkene basert i rene filnavn.

SETI@home prosjektet [17] er et avansert distribuert system og har til hensikt å distribuere regneoperasjoner til hjemmedatamaskiner. Regneoperasjonene skal lete etter tegn på intelligent liv i universet. Den sentrale tjeneren gir regneoppgaver til klientene og samordner prosjektet.

Søkemotorer som Goolge og MSN på Internett kan også kategoriseres som distribuerte nettverk. Her brukes også en sentral tjener eller tjenerpark til å gå igjennom, indeksere og gi et søkegrensesnitt for brukere. Søkealgoritmene og indekseringsalgoritmene for slike systemer baserer seg i global informasjon om nettverket (hjemmesidene, lenket sammen) og store tjenere for å prosessere søkene. Global informasjon og kraftige tjenere gir muligheten for rangering og vekting.

En av hovedforskjellene mellom P2P systemer og tradisjonelle distribuerte systemer er måten dokumentene søkes på. P2P fokuserer ofte på søk i små metadatamengder og treff på rene filnavn. I motsetning gir Internett-søkemotorer søk i innholdet i dokumentene og gode algoritmer for å rangere disse.

Semisentraliserte og atomiske P2P nettverk har ikke muligheten til å la dedikerte store ressursmaskiner ta tyngden for systemet. All funksjonalitet må distribueres i nettverket.

2.2 Strukturerte og ustrukturerte systemer

En naturlig oppdeling av nettverkstopologier er om de er strukturerte eller ikke. Zhang et. al. [56] tar opp dette skillet. De fleste nye og innovative distribuerte algoritmene baserer seg i hashtabeller. Dokumenter representeres av et <key, data> (nøkkel, data) par hvor noder i nettverket kan få tilgang til data fra andre noder ved å bruke nøkkelverdien. Data kan i dette tilfelle være enhver form for dokumenter. Skille mellom strukturerte og ustrukturerte systemer er hvor <key, data> parene lagres i nettverket. I strukturerte systemer er lagringsplassen for <key, data> settene definert av nøkkelverdien. Visse deler av nettverket har ansvaret for å lagre data med ett spekter av nøkkel verdier.

Søking og henting ved bruk av dokument identifikatorer refereres ofte som *structured overlays* eller distribuerte hashtabeller. CAN [46], OceanStore [39] og Chord [49] bruker en spesifikk struktur og et hashingsystem for å gjøre henting av distribuerte dokumenter effektivt. Hensikten med systemene er å optimalisere objekthenting ved å gjøre antall meldinger og hopp i nettverket minimale. Ulempen med disse systemene er at de ikke gir mulighet til å søke etter dokumenter basert på innholdet i dokumentene.

Ustrukturerte nettverk har ingen kontroll med hvor de forskjellige nøkkel og dataparene blir lagret. Freenet genererer et <nøkkel, dokument> par hvor nøkkelen er generert ved å kjøre en hashfunksjon på en beskrivende streng om dokumentet. Som en sidenotis kan det nevnes av Freenet er ett ustrukturert nettverk, men nøkkelverdiene vil konvergere i deler av nettverket over tid. Så man kan påstå at til lenger nettverket til mer strukturert blir det [28].

2.3 Persistente og Anonymiserte systemer

> *Arguing with anonymous strangers on the Internet is a sucker's game because they almost always turn out to be – or to be indistinguishable from – self-righteous sixteen-year-olds possessing infinite amounts of free time.*
>
> Neil Stephenson [48]

Internett har utviklet seg fra å ha vært et ukontrollert anonymt rom til et svært overvåket, kontrollert og kommersialisert rom. Persistente og anonymiserte systemer er et naturlig resultat av utviklingen på Internett. Den etiske siden av anonyme systemer har blitt stadig mer betent ettersom hele verden vil bruke akkurat Internett til å avsløre og finne terrorister og organisert kriminelle. I motsetning til det som kanskje er allment antatt så var denne utviklingen allerede godt i gang i hele verden før terrorangrepet på USA 11. september 2001. Spesielt Europa og USA har lenge giret opp juridistikken som sikrer fritt innsyn i all dataoverføring og telefoni [20].

Informasjon og kommunikasjonsteknologi har effektivisert mange forskjellige bransjer. Informasjon om bruk av tele- og dataoverføringer har muligheten til å effektivisere etterforskningen av kriminalsaker og forbygging av terror. Det som viser seg er at lover og regler laget for kriminell aktivitet ofte også kan brukes til overvåkning av privatpersoner. Under og rett etter den kalde krigen ble det rullet opp noen få saker om overvåkning i Norge. Informasjonen statlige organisasjoner nå kan samle om vanlige privatpersoner, primært på Internett, kan være langt mer omfattende.

I sin bok Peer to "Peer Collaboration and Sharing over the Internett" lister Bo Leuf opp grunnene for å kryptere innhold på Internett:

> *reasons to encrypt content stored on the Internet can be many, but a dominant theme is the persistence of published information, and the ability to successfully resist all attempts to either remove or change data*
>
> [40, s. 245]

Det er ofte feilaktig å si at ett nettverk er anonymisert. For at et system skal være helt anonymt må det gi anonymitet for alle typene bruk. Informasjons publiserer, holder, overfører og konsument må gjøres anonyme i nettverket.

Persistent betyr i denne sammenhengen at dokumenter aldri eller i en svært liten grad blir fjernet fra nettverket over tid. Ethvert dokument skal

kunne distribueres og repliseres på en slik måte at selv om noder faller ifra vil dokumentene finnes og være tilgjengelige i nettverket.

Moderne anonymitetssystemer strammer fra ideene Chaum [26] introduserte, hvor han foreslo å skjule korrespondansen mellom sender og mottaker ved å kryptere meldingen i flere lag med Public Key. Meldingen skulle sendes igjennom flere *mix'es* som dekrypterer, venter, lager ny rekkefølge på pakkene og krypterer dem på nytt før de sendes videre. Nettverk implementert etter denne tankegangen kan deles i to kategorier. Systemer som Babel, Mixmaster og Mixminion forsvarer seg mot kraftige angrep, men dette skaper høye hentetider og belastninger for nettverket [34]. Grunnen til høyere hentetider og belastninger er algoritmene for overføring av data i nettverket. Systmer som Onion Routing [50], arvtakeren TOR [33] og Freedom [24] leverer mulighet for kjappe overføringer for teknologier som for eksempel web-browsing. En av måtene slike nettverk bevarer anonymiteten er diversitet i nodeplassering og veiseleksjon.

Det bør også nevnes at den mest effektive måten å bruke Internett anonymt på er å logge seg på ett åpent Wi-Fi-nett [8] noen andre kontrollerer. MAC-adressen [9] til datamaskinen blir bare lagret i trådløsruteren og en eventuell IPadresse er en NAT-adresse [10]. Trådløsruteren vil gjøre all trafikk anonym og om man tømmer loggen i ruteren vil det ikke finnes noen mulighet for å koble datamaskinen opp i mot eventuell aktivitet utført på nettverket.

2.4 Anonymiseringsprosessen

Sitater rundt Internetts antatte anonymisering av brukeren har blitt en klisje [53]. John Gillmore skrev: *"The Net treats censorship as damage and routes around it"*.

Denne oppfatningen har vært rådende, men eroderende, siden Internet ble populært rundt 1995. I større og større grad lærer brukere at hvis de ikke tar spesielle forhåndsregler kan Internettaktiviteten bli sensurert og lenket opp mot identiteten til brukeren [53].

Anonymisering på Internett eller andre datanettverk er et vagt uttrykk. Å forholde seg anonymt på Internett betyr at ingen informasjon for identifisering av person, personinformasjon, eller IP-adresser gjøres synlig for nettverket

[8] Wi-Fi nettverk er trådløse nettverk for privatpersoner. En router koblet til Internett gir oppkobling for datamaskiner med trødløst nettverkskort

[9] MAC-adressen er den fysiske og gloable identifiserende adressen til nettverkskortet

[10] NAT-adresen er en lokal IP adresse gitt av trådløstilkoblingspunktet. Trådløstilkoblingspunktet for personlig bruk bruker denne adressen for å sende datapakker til datamaskinen. NAT-adressen er ikke synlig for Internett

eller en overvåkende entitet. Videre defineres anonymiteten å gjelde klient applikasjonen og datamaskinen som overfører dokumenter i et nettverk. I denne oppgaven er fokuset for anonymiseringen ultimalt sett brukeren av applikasjonen som benytter seg av anonymiseringstjenesten. Informasjon om brukeren selv er likevel ikke fokuset for anonymisering på Internett. Anonymiseringen gjøres igjennom beskyttelse av informasjon om datamaskinen som igjen kan kobles mot brukeren. Anonymisering gjelder for flere typer handlinger og anonymiteten må behandles for hver og en av klassene under:

publiserer: Publiserer gjør dokumenter tilgjengelige for andre.

holder: Holder har dokumentene lagret på sin datamaskin og gjør de tilgjengelig for resten av nettverket.

overfører: Overføreren har i oppgave å sende meldinger (kan inneholde datapakker, meldinger eller spørringer etter dokumenter) videre i nettverket.

konsument: Konsumenten henter dokumenter fra andre i nettverket.

Publiserer må kunne gi dokumenter til nettverket uten at lokaliseringsinformasjon kan knyttes til publiserernoden.

Holder må ha muligheten til å beskytte informasjon om at dokumentet er lokalisert på datamaskinen og utsending av dokumenter anonymiseres. Algoritmen for lokalisering av dokumenter må ikke basere seg i overvåkbare pekere av typen IP-adresser. Når dokumenter sendes fra holdernoden må trafikken ut fra noden ikke kunne overvåkes av andre maskiner i nettverket eller fra en utenforstående kilde. I tillegg må dokumentene lagret på holdernoden være utilgjengelige for triviell lesbarhet. Om noden er lokalisert og angrepet må dokumentene være uttilgjengelig for angriperen.

Overfører må kunne sende datapakker uten at overfører eller andre noder vet at trafikken går igjennom noden. I tillegg må overfører ikke ha muligheten til å se hva som overføres. Dette kriteriet er for sikkerheten til publiserer og konsument. Pakker sendt gjennom noden må ikke kunne settes sammen og dekryperes.

Konsumenten må beskyttes fra å bekreftelse av mottatt dokument. Når ett dokument er lokalisert i nettverket og skal sendes til konsumentnoden må konsumenten kunne beskyttes fra bekreftelse av at det er denne noden som tar i mot dokumentet. Overvåkning av holder og konsumentnode med datastrømmen mellom de, må ikke kunne stadfeste at det er konsumentnoden dokumentet blir lagret hos.

Dokumenter med privatinformasjon av typen navn, personnummer og annen sensitiv informasjon er umulig å fjerne på en effektiv måte hvis brukeren selv er uoppmerksom eller uforsiktig. Dette er aktuelle problemstillinger da bilder, tekstdokumenter og videofilmer gjerne har mye metadata skjult i dokumentet.

2.4.1 Anonymiseringsteknikker

Metoder for å anonymisere nettverk er mange. Grove kategorier av forsvarsmekanismer mot sniklytting og overvåkning kan deles i fire [34] [33]:

Pakkeopphopning: Nettverket og noder i nettverket samler opp og sender pakker videre i en annen rekkefølge for å hindre motstanderen å overvåke hvilken melding i forsendelsen kom fra en gitt avsender.

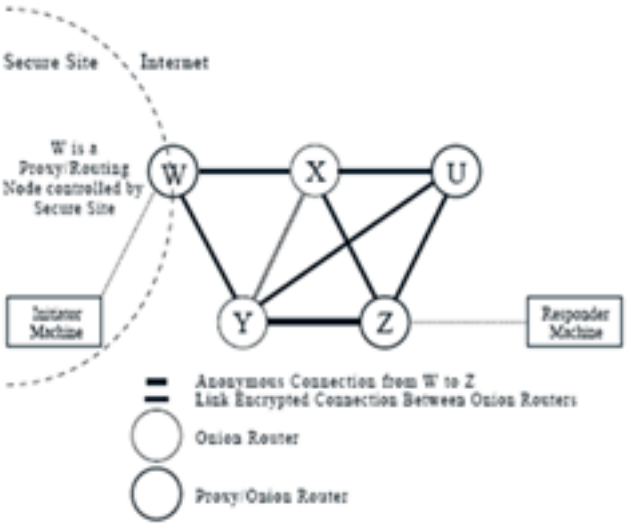
Feilsending: Nettverket sender med hensikt pakker feil og for langt. Lokketrafikk så vel som vanlig trafikk går samtidig over nettet.

Spredning: Fjerne muligheten for overvåker av nettverket til å se både sender og mottaker.

Kryptografi: Meldinger krypteres (gjerne i flere lag) for å vanskeliggjøre overvåkning av nettverket og for overføringsnode å snappe opp informasjon.

2.5 Small world nettverk modell

Small world (*verden er liten* på norsk) bygger på empiriske observasjoner i sosiale nettverk. For alle to individ finnes det en kort sekvens med mellomstående bekjentskaper [38, 54]. Prinsippet kan kobles mot grafteori i nettverk. Antagelsen og prinsippet sier at mellom alle sett av to noder i et nettverk finnes det en svært kort vei. Den korte veien mellom de to nodene måles i antall noder mellom i nettverket. Det har blitt vist at konstruksjon av *small-world* nettverk har grafer med liten diameter noe som fører til mindre routing distanse mellom to noder [54] [56]. Majoriteten av noder i et slikt nettverk har få lokale tilkoblinger til andre noder, men noen få noder har svært mange tilkoblinger. De svært tilkoblede nodene (har mange tilkoblinger til andre noder) muliggjør snarveier gjennom nettverket og snarveiene fører til effektive korte veier mellom alle noder i nettverket [28].



Figur 2.1: Onion Routing

2.6 State of the Art

Det finnes mange systemer og applikasjoner som gir bruker mulighet til å anonymisere brukerens aktivitet på Internett. Under presenteres de to viktigste og mest brukte systemene: **Freenet** og **TOR**. Systemene er både forskningsprosjekter og implementerte løsninger. Freenet er hovedfokuset siden dette systemet er en av de mest avanserte anonymiserte distribuerte systemene. Avhandlingen baserer seg i å løse innholdssøk for tekstdokumenter i systmer likt Freenet.

2.6.1 TOR: Onion Routing

TOR systemet bygger på tankegangen fra Onion Routing. Heller for å sette opp en tilkobling til en annen maskin lager applikasjonen den samme tilkoblingen gjennom en sekvens av maskiner kalt *onion routers* [50]. Denne type tilkobling gjør at tilkobler og den tilkoblede maskinen blir anonyme for hverandre. For at anonymitet skal opprettholdes må alle identifiserende informasjon fjernes fra datastrømmen før det blir sendt over den anonyme tilkoblingen. *Onion routing* er et distribuert nettverkdesign for å anonymisere TCP-baserte applikasjoner som web-browsing, secure shell, instant messaging (IM) og e-post. Tidligere foreslåtte systemer som Babel [37] og Mixmaster [44] skapte for store tidsbelastninger til å bruke interaktive oppgaver.

TOR er en protokoll for asynkron, løst sammensatte *onion routers* som utvider Onion Routing med [33]:

- **Perfekt fremover sikkerhet:** Heller enn å dekryptere meldinger som lagene i en løk, blir meldinger kryptert og dekryptert i veien fra dokument holder til dokument konsument.
- **Delt anonymisering og protokoll rengjøring:** En standard proxy tjener fjerner identifiserende informasojn for de fleste applikasjonene som benytter seg av TCP.
- **TCP routing:** For hver tilkobling kan alle applikasjoner bruke den samme krypterte veien til den andre datamaskinen uten å opprette flere Public Key nøkler.
- **Congestion control:** Tidligere anonymiseringsdesigner takler ikke flaskehalsproblematikken. Belastningsbalansering og flytkontroll krever mellomnodekommunikasjon og globalt bilde av trafikken. TOR bruker desentralisert belastningsbalansering og flytkontroll ved at løvnoder identifiserer og nøytraliserer overbelastning ved nedsatt aktivitet til overbelastede punkter.
- **Katalogtjenere:** TOR utvider *onion routing* fra å overbelaste nettverket med tilstandsinformasjon til å ha flere katalogtjenere. Kjente rutere og deres tilstand lastes periodisk ned fra tjeneren.
- **Flere typer utgang:** Pakker på vei ut fra TOR nettverket er kontrollert av applikasjonen. Ikke alle noder vil la all type kommunikasjon gå ut fra sin datamaskin og det finnes muligheter for å automatisk sette parameter for slik kommunikasjon.
- **Datasikkerhet:** TOR leverer en *checksum* [11]tjeneste som garanterer integriteten til data overført gjennom nettverket.

TOR er en kompromissløsning. Systemet takler overføring for alle applikasjoner som bruker TCP og overføring unngår et kjent problem med TCP over TCP overføring [51].

Hver router i nettverket har en åpen tilkobling til alle andre routere i nettverket. Hver bruker kjører en lokal applikasjon kaldt en *onion proxy* (OP). OP henter informasjon fra katalogtjenere, lager tilkoblinger over nettverket og håndterer tilkoblinger fra brukerapplikasjoner. Flere typer nøkler er holdt

[11]Checksum viser til en metode for å verifisere integriteten til datapakke. Bergninger på byte nivå regner ut om datapakken er riktig etter overføring gjennom nettverket.

for hver maskin i nettverket. Noen er permanente og noen er temporære og brukes til forskjellige kryptografiske operasjoner.

TOR kan tvinges til å gi fra seg identifiserende informasjon. Etter angrep på danske nettsider satte et dansk konsulentselskap (FortConsult) igang med å knekke TOR nettverket. Både informasjon om alle TOR brukere og hvem som prøver å aksessere en datamaskin kunne identifiseres [27]. For å identifisere nettbrukere lurer systemet TOR klienter til å gi fra seg informasjon gjennom Flash, Javascript, ActiveX eller Java programmene på TOR noden.

TOR er ikke bare en forskningsprototyp, men et system i bruk med flere hundre tusen brukere[18].

2.6.2 Freenet

> *Our Project Freenet, is a distributed information system designed to address information privacy and survivability concerns*
>
> Ian Clark [28, s. 40]

Freenet er et selvorganiserende strukturert brukersentrert P2P nettverk [29, 28]. Ressursene distribueres utover i nettverket hvor ubrukt harddiskplass hos noder blir brukt til å distribuere filene. Kommunikasjonsprotokollen i bunnen har svært lik funksjonalitet som Internet Protocoll, hvor alle nodene i Freenet utfører den funksjonaliteten en router gjør i Internett. Overføringen mellom noder er kryptert med Public Key. Likt som i IP vet noder bare om sine nabonoder og disse er av begrenset antall. Grunnen er at IP-adresser ikke lenger skal være bunnstrukturen i direkte sammenkobling av noder. En av de viktigste anonymiseringsfunksjonene i Freenet er at innsyn i IP-adresser fjernes. IP-adresser er mulig å spore tilbake til en identitet og kan brukes til å identifisere brukere. Pakker sendes i nettverket uten at nodene vet gjerne ikke hvor i nettverket mottakernoden befinner seg og sender i vilkårlig retning. Sending av pakker er ikke totalt vilkårlig. Hver node har en routingtabell som indikerer i hvilken retning pakken skal sendes. Om for eksempel en node har sendt en spørring etter et dokument, vil dokumentet følge samme rute tilbake til spørrenoden på grunn av routingtabellene og at nodene på nettverksruten midlertidig lagrer hvor pakker sendes fra og til.

Aktualitetn til Freenet kan sees i at "Freenet: A Distributed Anonymous Information Storage and Retrieval System" [29] var i følge CiteSeer [4] den mest siterte IT-artikkelen i 2000.

Designfokuset til Freenet [28]:

- mulighet for å beskytte privatinformasjon for informasjonspubliserer, informasjonskonsument og informasjonsholder.

- motvirke informasjonssensur
- høy tilgjengelighet og pålitelighet gjennom desentralisering og
- effektiv, skalerbar, adaptiv lagring og routing

De første anonymitetssystemene har vært en tjener som forhandler overførselen fra en datamaskin til en annen. Tjeneren er en tredje part begge de kommuniserende datamaskinene stoler på. I Freenet er anonymiteten en av hovedfokusene. Anonymiteten skal ikke gjelde bare for en type brukere, den skal gjelde for all type bruk av nettverket.

Informasjonspubliserer er den brukeren som legger ut dokumenter til deling, informasjonskonsumenten laster ned dokumenter fra nettverket og informasjonsholderen har dokumentene lagret på sin datamaskin. Tjenestene nettverket må levere for å anonymisere de forskjellige bruksområdene er både ulike og overlappende. For alle bruksområdene stiller nettverket med kryptografi. Alle pakker sendt i nettverket er kryptert i flere lag. Den mest brukte krypteringsmetoden er en versjon av Public Key kryptering. Denne krypteringsmetoden er svært sikker og baserer seg på 2 veldig store primtall. Tallteorien for krypteringsmetoden gjør det svært vanskelig å dekryptere pakkene uten rett krypteringsnøkkel. Public Key refererer til metoden for utveksling av krypteringsnøkler. Basiskonseptet er at man velger to veldig store primtall for å så gange disse sammen. To tall (nøkler) genereres av disse. Det ene tallet brukes til kryptering og det andre brukes til dekryptering. Krypteringsnøkkelen kan sendes til allmennheten mens dekrypteringsnøkkelen holdes hemmelig. Alle andre kan kryptere en pakke, men bare den som har dekrypteringsnøkkelen kan dekryptere pakken. Alle bruksområder tar også nytte av pakkeroutingen i systemet. Om en klient sender en pakke sendes denne en modifisert tilfeldig vei gjennom nettverket. Sendernoden vet bare om den lokale noden den først sendte pakken til.

Pakkerouting

En anonymiseringsfunksjonalitet er routing av pakkene i nettverket. Pakkene sendes videre og alle noder i veien gjennom nodegrafen lagrer pakken, men om mottakernoden vil det, kan den også sende pakken videre når pakken kommer frem. Pakken sendes så med en tid-å-leve verdi ut i nettverket. Mache et. al. [42] har foreslått forbedringer av pakkeroutingalgoritmen. Forslaget er å lagre mer informasjon om hentingsspørringene for å forbedre fremtidige spørringer.

Routing mekanismene i Freenet er designet for at nettverket skal utvikle seg over tid. Utviklingen fører til en tilstand hvor nettverket gir lave gjennomsnittlige veier til noder og større sannsynlighet for funn av dokumentet. Hentingen av dokumenter utvikler seg med den hensikt å gjøre den gjennomsnittlige veilengden til noder mindre. Routingtabellene lagrer midlertidig hvor de beste svarene kommer fra. Når dette gjøres i hele nettverket vil dette automatisk føre til kortere vei fra informasjonskonsument til informasjonsholder.

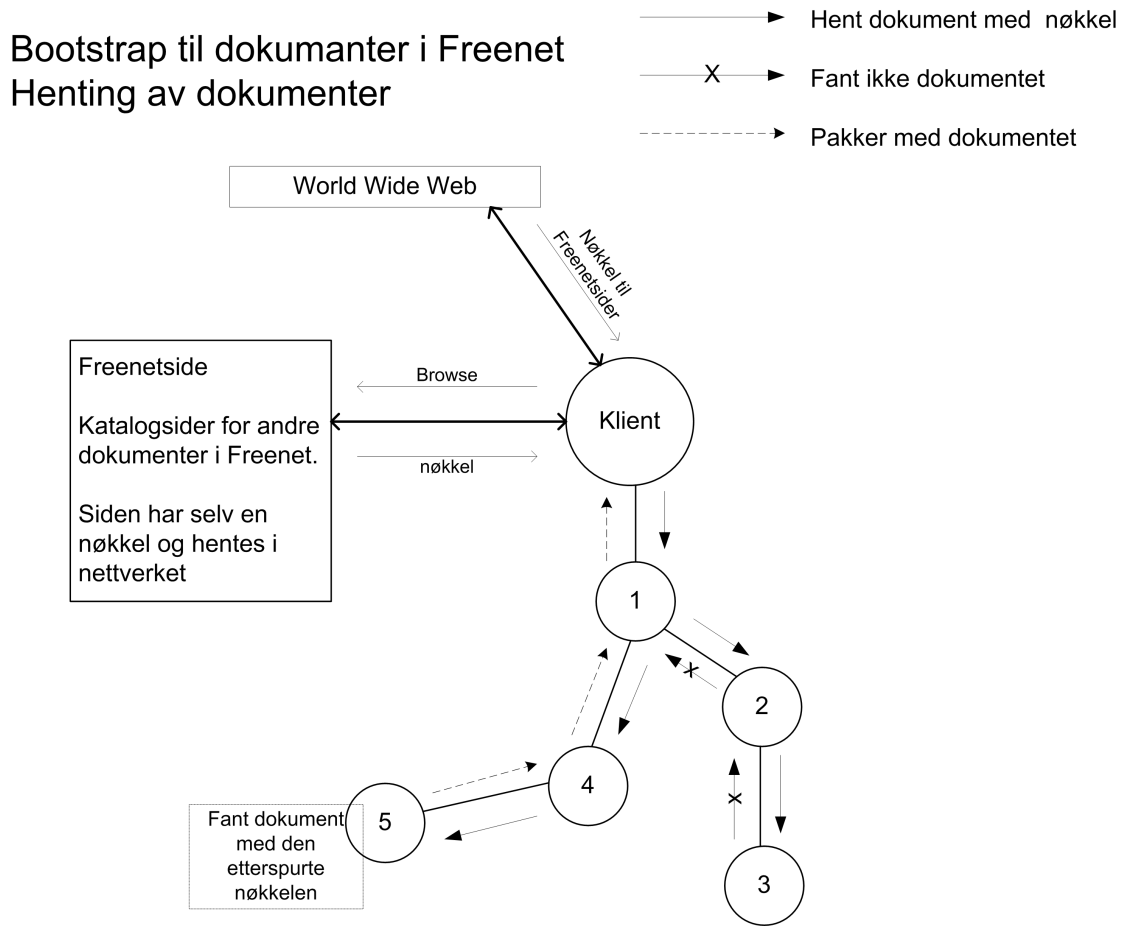
Skaleringen og feiltoleransekarakteristikkene til Freenet kan forklares etter *small world* nettverkmodellen [28].

Publisering og nøkler

Når ett dokument publiseres i nettverket blir dette lagret på flere andre noder enn den som publiserer. Publiserernoden lager en hashnøkkel også omtalt som *location-independent- globally unique identifier*(GUID nøkkel) av dokumentet og dette gjøres tilgjengelig for nettverket. GUID genereres ved bruk av SHA-1 sikker hashing algoritme. En hashalgoritme er en funskjon som leverer en verdi tilbake basert på informasjon lagt inn. Hashfunksjonen kan for eksempel ta inn en streng med tekst og gi tilbake ett langt tall.

GUID består av mer informasjon. En hashnøkkel lages av dokumentet og kalles *content-hash key* (CHK). CHK lagres sammen med dokumentet i informasjonsholder noden. Dokumentene hos informasjonsholderen krypteres og kobles til hashnøkkelen publisereren lagde. Hashnøkkelen gir ingen indikasjon av hva dokumentet inneholder. På denne måten kan dokumenter hentes uten at holdernoden vet hva den deler ut. I tillegg til CHK består GUID av en *signed-subsspace key* som beskriver dokumentet. Når filen publiseres lages en menneskelesbar tekststreng på formen POLITIKK/NORGE/NASJONALSIKKKERHET. Denne verdien brukes til søk etter dokumenter og for *checksum* verifisering av dataobjektet [29].

Når GUID og dokumentet blir publisert til nettverket blir dokumentet lagret og lenket til gjennom nettverket. Henting av dokumenter støttes av GUID nøkkel plassering i nettverket og publiseringen skjer mot den delen av



Steg 1: hente nøkkel til en Freenetside fra www. Klienten spør nettverket etter nøkkelen. Node 1 har ikke selv nøkkelen og sender spørringen videre til 2 og 3 som heller ikke har dokumentet (anta at *htl* er 3 og brukt opp når spørringen når node 3). Node 1 blir spurt på nytt om å finn nøkkelen og sender spørringen til 4 fordi 2 returnerte negativt sist. 4 har ikke nøkkelen selv og routingtabellinformasjonen sier at nøkkel av denne typen ligger mot node 5. 5 har nøkkelen, og returnerer så dokumentet i retningen spørringen kom fra. 4 gjør det samme og dokumentet blir levert gjennom 1 til Klienten. Klienten viser så dokumentet på skjerm (Dokumentet lagres også i 4 og 1 etter Path Replication prinsippet).
Steg 2: hent flere nøkler fra Freenetsider og spør nettverket etter nøkkelen. Etter flere spørringer vil routingtabellen bli bedre og bedre.

Figur 2.2: Bootstrapping og henting av dokumenter i Freenet

nettverket koblet til en viss type GUID nøkkel. Hver node mottar dokumentet og kan sette seg selv som publisereren av dokumentet [28].

Legge til noder

Noder akkurat pålogget nettverket må finne lenker til en eller flere noder fra utsiden av Freenet [29]. Når noden har kommet inn i nettverket kan den lytte for å finne flere noder, men andre noder må ha muligheten til å oppdage den nye noden. Den nypåloggede noden velger en tilfeldig *seed* og sender denne til en av de nodene den er koblet til. Mottakernoden velger en tilfeldig node i sin routingtabell og sender meldingen videre. På denne måten blir nye noder koblet inn i nettverket ved hjelp av de nodene den har funnet utenfor Freenet. *Seed* meldingen gir informasjon til mottakernodene om hvilke sett av GUID nøkler noden lagrer. Meldingen sprer seg i nettverket og mottakernodene lagrer den nye noden og kobler GUID spektret til denne noden.

Henting av dokumenter

Overføringen av filer mellom noder i nettverket gjøres ved at en node sender en spørring etter en dokumentnøkkel. Nettverket er strukturert for å mer effektivt hente frem dokumenter på denne måten. Om informasjonskonsumenten (noden som spør etter dokumentet) ikke har spurt etter dokumenter før sendes spørringen til en tilfeldig node i routingtabellen. For hver gang konsumentnoden gjør en spørring vil noden lagre informasjonen om hvordan søket gikk. Om spørringen gikk bra, vil noden kunne forbedre søket sitt i fremtiden og sende søket til den noden i routingtabellen som mest sannsynligvis vil returnere best svar. På denne måten vil hastigheten og recall[12] forbedres over tid. En annen forbedringsfunksjonalitet er Path Replication. Hver gang et dokument blir funnet i en holdernode vil dokumentet lagres i alle noder i veien tilbake til spørringsnoden. Selv om det ikke går an å observere ett fungerende Freenet utenfra etter det er satt i gang er det laget matematiske modeller for systemet. En effekt av Path Replication er i følge [41], [28] at dokumenter med like nøkler (nøklene er en hash av filen og sier ingenting om innholdet i filen) vil hope seg opp i visse deler av nettverket. Path Replication gjør også at populære filer blir lettere tilgjengelig og at populære filer forblir i nettverket over tid.

[12]Recall R = (antall relevante dokumenter hentet / antall relevante dokumenter). I denne avhandlingen brukes recall på to forskjellige måter. Recall i teori er gitt av definisjonen ovenfor. Recall for løsningen er (antall dokumenter hentet / antall totalt med søkeord). Recall bygger på at det er definert relevante dokumenter i dokumentmengden for ett gitt søkeord. Det finnes ikke slik informasjon om dokumentsett bruk i denne oppgaven.

```
request(Key, Hops_to_Live, Passer)
    if (a document in my store matches Key)
        pass Document back to Passer
        return SUCCESS
    endif
    if (Hops_To_Live equals 0)
        return EXPIRED
    endif
    decrement Hops_to_Live
    while(reference left in my routing tabel)
        in routing table find node with closest
         key that has not previously been tried
        request(Key, &Hops_To_Live, me)
        if(return EXPIRED)
           return EXPIRED
        endif
        if(return SUCCESS)
            add Document to my store //caching
            add reference pointing to Fulfiller
            pass Document back to Passer
            return SUCCESS
        endif
    endwhile
    return BACKTRACK
endrequest
```

Figur 2.3: Pseudokode for henting av dokumenter i Freenet[42]

Freenet er et persistent system i den grad at nettverket søker å lagre all informasjonen i uendelig lang tid. Sett ovenfor blir populære dokumenter spred utover nettverket mens upopulære dokumenter ikke vil sprees. I det tilfellet at en node ikke har mer lagringsplass må noen av dokumentene fjernes for gjøre plass til nye. Dette vil føre til en automatisk "popularitetssensurering" i Freenet. Kaldt popularitetssensurering fordi det er den eneste måten et dokument kan fjernes fra nettverket og Freenet har til hensikt å umuliggjøre sensurering ved bruk av dokumentspredning i nettverket. Populære dokument vil være spredt utover hele nettverket og det er ikke sannsynlig at filene synkront blir slettet i alle noder. Et veldig upopulært dokument vil sakte forsvinne ut av nettet når andre dokumenter kommer til. For publiserer er ikke dette ett problem. Om en informasjonspubliserer ser at dokumentet han publiserte er borte fra nettverket kan det samme dokumentet raskt publiseres igjen.

Pseudokode for dokumenthentingsalgoritmen i Freenet er beskrevet i figur 2.3.

Lagring av dokumenter

Freenet er hemmet av begrenset lagringsplass [29]. Alle noder holder et visst sett med dokumenter. Dokumentene er organisert etter *sist brukt* verdier. Sist brukt kan være publiseringstidspunkt eller sist etterspurt tidspunkt. Om lagringsplassen er full og nye dokumenter kommer til vil dokumentet med gamlest *sist brukt* verdi fjernes. Siden lenker til GUID-nøkkel for fjernet dokument fortsatt peker til noden som har slettet dokumentet er det mulig at det samme dokumentet hentes og lagres på nytt etter spørring.

Fordelen med en slik fremgangsmåte er at upopulære dokumenter kan forsvinne sakte ut fra nettet.

Søkefunksjonalitet

Fritekstsøkefunksjonaliteten i Freenet er foreløpig begrenset. Brukere må enten benytte seg av katalogoppslag på spesielle Freenet-sider eller søke i spesielle tekststrenger (*signed-subspace key*).

Katalogene er HTML sider hentet fra Freenet-nettverket. Disse sidene er lenket sammen slik at brukeren kan navigere seg videre til andre sider. Katalogene er løst organisert og inneholder sammendrag skrevet av andre brukere. For brukere med liten kunnskap om et tema kan katalognavigering være fornuftig. For å finne spesifikk og relevant informasjon er katalognavigering en dårlig løsning. Katalognavigering er bare like god som strukturingen av katalogen og brukeren må selv finne frem til relevante dokumenter.

Signed subspace key er en liten fritekst som beskriver dokumentet. Brukere har muligheten til å søke med Random Walk algoritmen gjennom disse. Det er begrenset hvor mye informasjon en streng på formen *Politikk/etikk/freenet* kan gi. I tillegg er disse tekststrengene skrevet av informasjonspubliserer og kan stemme dårlig overens med konsumentens ordbruk i søket.

Når en bruker publiserer ett dokument til Freenet gjøres dette ved å legge til en lenke til lokaliseringsnøkkelen til dokumentet i en Freenet-side.

Måten Freenet gjør dokumenter tilgjengelig er å bruke en form for hashtabeller. Når en node publiserer ett dokument genereres en nøkkel av en beskrivende tekststreng koblet til dokumentet. Dokumentet blir lagt i en hashtabell på formen <nøkkel, dokument>. Dokumentet gjøres så tilgjengelig ved at informasjonspubliserer sender nøkkel og dokument paret ut tilnettverket. Informasjonskonsumenten henter nøkkelverdien fra en Freenet-side og sender et søk etter denne nøkkelverdien til nettverket. På grunn av *Path Replication* vil nøkkel og dokument parene konvergere i visse deler av nettverket. Routingtabellene hos noder modifiserer henting av dokumenter ved å midlertidig (*cashing*) lagre hvilken nabonode er nærmest nøkkelverdiene [28].

Sikkerhet

Det finnes måter å bryte anonymiteten til Freenet. Det jobbes med å tette disse hullene, men Ian Clarke [29] peker på noen fareområder. Om inntrengeren får kontroll over nettverket nodene er på kan data om søk og henting av data gjøres synlig. Et annet problem kan være Public Key algoritmen. Om man setter en regnemaskin til å faktorisere produktet av de to store primtallene vil denne til slutt finne tallene. For 128 bits tall tar det rundt regnet 8 måneder å dekryptere en slik nøkkel ved bruk av Brute Force (algoritme hvor alle tall testes ut) og bruk av 1600 datamaskiner med distribuert prosessorkraft. Dekrypteringen vil bruke $((1.2)10^{23})$ prosessorsykler [25].

Dokumenter lagret på en node i Freenet er kryptert. Siden alle noder i nettverket kan hente en fil og må få denne dekryptert er krypteringsalgoritmen viden kjent for nettverket. Hensikten med å ha filene kryptert på holdernoden er for å gi denne noden argumentet at den ikke vet hva som er lagret på maskinen. For alle dokumenter lagret på denne måten er det enkelt mulig å dekryptere å se filene.

Ett problem med kryptering for å holde informasjonen hemmelig er at det i flere land er ulovlig med avansert kryptografi for privatpersoner. Når man installerer Freenet får man beskjed om at programvaren kan være ulovlig i noen land og at brukeren selv må sjekke dette.

System	Attacker	Sender anonymity	Key anonymity
Basic Freenet	local eavesdropper	exposed	exposed
	collaborating nodes	beyond suspicion	exposed
Freenet + pre-routing	local eavesdropper	exposed	beyond suspicion
	collaborating nodes	beyond suspicion	exposed

Figur 2.4: Sikkerhet i Freenet [29]

Pre-routing i figur 2.4 referer til en annen form for routingalgoritme enn den vanlige Freenet routing algoritmen. Pakker sendes gjennom en spesiell kryptert vei fra dokument holder til konsument.

Anonymitet

Informasjonskonsumenten beskytter alle metodene over. Alle funksjonene i Freenet er implementert for å beskytte alle typer bruk av nettverket fra identifisering av datamaskin og identiteten til brukeren.

Den viktigste anonymitetsfunksjonaliteten gies av routingalgoritmen og kryptografien. Routingalgoritmen umuliggjør forskjellige former for angrep på nettverket, men ikke alle som vist i figur 2.4. For alle angrep hvor "motstanderen" ikke har total kontroll over nettverket eller kontroll over mottaker og sender vil Freenet beskytte identiteten og aktiviteten til brukere. Kryptografien beskytter både korrespondansens innhold mellom nodene og muligheten for å finne ut hvilke pakker skal settes sammen og hvilken node har nøkkelen til å dekryptere de.

2.7 Teori tilknyttet Freenet

Anonymiseringstjenester er koblet sammen av flere forskjellige teknikker for å gi brukeren anonymisering på Internett. Noen av teknikkene anonymiseringsprosessen gjør seg nytte av kan knyttes opp mot lovgivning. Nedenfor presenteres kryptografilovgivning for OECD landene og teori rundt den private sfære.

2.7.1 Kryptografi

OECD landene har samarbeid om å utvikle lovverk for kypteringspolitikk [16]. OECD satt i 1996 i gang et prosjekt for krypteringspolitikk ved å sette sammen en gruppe eksperter innenfor feltet. Dette skjedde i regi av Committee for Information, Computer and Communications Policy (ICCP-komiteen).

> *Dersom offentlige myndigheter vurderer politikk for kryptometoder som sørger for rettshjemlet adgang, bør de nøye veie fordelene, herunder fordelene for offentlig sikkerhet, rettshåndheving og rikets sikkerhet, så vel som risikoen for misbruk, tilleggskostnader for eventuell underliggende infrastruktur, muligheter for tekniske uhell og andre kostnader.*
>
> Oversatt OECD rapport [16]

Oppsummert i en setning sier rapporten at lovverket for kryptografi er foreldet og trenger fornyelse. Lovverket rettes inn for å ta hensyn til privatinformasjon (denne paragrafen er tiltenkt e-handel og andre kommersielle verktøy) og riktets sikkerhet. Freenet gir svært sikker kryptografi til enkeltpersoner og kan falle i begge kategoriene ettersom hvordan hver nasjon tolker retningslinjene fra OECD.

> *På G7-toppmøtet om tiltak mot terrorisme i juli 1996 kunngjorde G7-regjeringene at det skulle holdes hyppigere samråd "i egnede bilaterale og multilaterale fora om bruk av kryptering som ved behov gir offentlige myndigheter rettshjemlet adgang til data og kommunikasjon for blant annet å forebygge eller etterforske terrorisme, samtidig som legitim kommunikasjon sikres fortrolighet".*
>
> Oversatt OECD rapport [16]

Public Key kryptografi på linjene Freenet har lagt seg på gir ikke innsyn for noen parter, og eneste måte å dekryptere meldinger er å bakover-programmere en beslaglagt datamaskin eller løse nøkkelen for hver og en melding. Siden meldingene repliseres utover i nettet og med hensik sendes feil kan ikke intrengere (her organisasjoner knyttet til nasjonal sikkerhet) vite hvor dekrypteringsnøkkelen ligger. I tillegg er det umulig å vite om dekrypteringsnøkkelen blir lagret eller kastet eller om den dekrypteringsnøkkelen på den beslaglagte maskinen er en ny eller gammel nøkkel. Freenet kan ikke regnes som lovlig programvare etter de overforstående G7 retningslinjene. Siden det ikke finnes gode internasjonale standarder for krytografilovgivning er det vanskelig å stadfeste hvorvidt Freenet er lovlig eller ulovlig programvare.

2.7.2 Privatsfæren

Agre og Rotenberg [19] peker på den konseptuelle forskjellen mellom den private sfære og den offentlige sfære. Hvorfor er det greit å lese over skulderen til en venn, men ikke til den som sitter fremfor deg på bussen? Hvorfor er det greit å se inn i butikkvinduer, men ikke greit å stå og titte inn hos naboen? Den offentlig og den private sfære har forskjellige implisitte regler rundt akseptabel oppførsel. IKT-applikasjoner har laget en gråsone hvor disse reglene ikke lenger er lært i voksen alder. I tillegg har trenden rundt Internet vært å fjerne muligheten til å ta med seg sin private sfære og medfølgende regler med seg ut på Internett.

Lov om behandling av personopplysninger (personopplysningsloven) [13] omhandler behandling og registrering av personopplysninger. I Kapittel 1 §2 defineres personopplysning som: *opplysninger og vurderinger som kan knyttes til en enkeltperson*. Det viktige for P2P systemer og overvåkning på Internett er *kan knyttes*. Kapittel 2 §8 defineres reglene for hvilke grunner behandling og registrering av personopplysninger kan ha. Overvåkning av P2P nettverk kan bare gjøres om det finnes en lov som tilsier registrering og behandling av slik informasjon. IP-adresser kan knyttes opp mot en enkeltperson og derfor ligger overvåkning av P2P tjenester under personopplysningsloven. Innsamling av informasjon om bruk av P2P systemer har blitt et kraftig verktøy for film og musikkbransjen representert av Motion Picture Association (MPA) og Den Internasjonale platebransjens forening (IFPI). Ikke uten grunn har det norske Datatilsynet gått ut å krevd innsyn i overvåkning av nordmenns aktivitet på P2P systemer [6]. Fildelingsnettverk deler gjerne ulovlig informasjon og denne aktiviteten bør kontrolleres, spesielt for de som tjener penger på slik aktivitet. Ulovlige handlinger bør ikke brukes for å straffe annen ulovlig aktivitet.

"Database nation" av S. Garfinkel omhandler akkurat denne problemstill-

ingen. Ut i fra boken kan man trekke den røde tråden at informasjons og kommunikasjons teknologi krenker retten personer har til egen informasjon [35].

Kapittel 3

IR i distribuerte systemer

Sam Lowry: Do you want to see my ID?

Porter - IR: No need, sir.

Sam Lowry: But I could be anybody.

Porter - IR: No you couldn't sir. This is Information Retrieval.

Jack Lint [1]

Informasjonsgjenfinning (Information Retrieval IR) omhandler representasjon, lagring, organisasjon av og tilgang til informasjonsdokumenter [22]. Informasjonsgjenfinning skiller seg fra datagjenfinning ved hvilke og hvor mage dokumenter som returneres. I datagjenfinning skal alle dokumenter som tilfredsstiller visse klart definerte relasjonsbaserte algebrautrykk returneres i søket. En liten feil i et datagjenfinningssystem kan bety total feil mens små feil og mangler i svarsettet til IR systemer ikke er farlig. IR-systemer skal gi tilbake relevante dokumenter gitt et søkeord. Noden tradisjonelle IR teknikker innenfor indeksering, søkeoperasjoner og søkealgoritmer kan overføres til distribuerte systemer. Teknikkene må omgjøres for å ta hensyn til ressursmangel (prosesseringskraft, minne, båndbredde og lagringsplass), mangel på sentral tjener, dokumentlokalisering og eventuell anonymisering. Nodene i nettverket skal utføre alle operasjonene og algoritmene må tilpasses ressursene nettverket har. For eksempel kan avanserte og gode algoritmer basert i inverterte filer kan være for ressurskrevende for datamaskinene i nettverket.

[1] Hentent fra filmen Brazil, `http://imdb.com/title/tt0088846/`

3.1 IR i distribuerte systemer(P2P)

I nettverk hvor det ikke finnes noen sentral tjener for å koordinere søk finnes det flere søketeknikker. Brian F. Cooper [30] tar for seg søk i atomiske nettverk. For sine forsøk har han brukt fire forskjellige søkemetoder og disse er representable for eksisterende P2P-systemer:

1. **Flooding** er den originale Gnutella søkeprotokollen. Når en node tar i mot et søk, søker den igjennom sine egne filer og sender søket ut til alle nabonoder. Søket gies en "*time-to-live*"(ttl), altså hvor langt ut i nettet det skal penetrere.

2. **Iterative deepening eller Expanding Ring**[41] er flooding hvor *ttl* verdien økes i flere iterasjoner hvis ikke nok søkesvar returneres.

3. **Random walks** sender ikke meldinger til alle nodene i routingtabellen. Hver gang en node tar i mot ett søk, prosesseres dette søket lokalt og sendes videre til en tilfeldig valgt nabo. Søket kan avsluttes ved at *ttl* verdien er utgått eller at nok dokumenter er funnet.

4. **Biased Random Walk** blir meldingen sendt videre til den noden med flest noder i routingtabellen. I tillegg legges det ved informasjon om besøkte noder i søket.

Flooding har flere begrensninger. Hvis *ttl* verdien er for høy så er det en stor byrde for nettverket og om den er for lav får ikke brukeren gode nok svar [41]. Ett annet problem det også pekes på er duplikater av søkemeldinger. Den samme noden kan få og sende den samme meldingen flere ganger om nettverket inneholder sykler. Ett forslag er at random walk melder tilbake til søkenoden med jevne mellomrom for å verifisere at den skal søke videre. I slike søk vil ttl være svært høye og når random walk utfører "*checking*" til søkenoden bruker denne operasjonen opp ttl. Qin Lv et. al. introduserer også en *K-Walker Random Walk* som skalerer bedre som søkemetode enn *flooding* og forbedrer vanlig *Random Walk* [41].

3.1.1 Eksempler på IR i P2P systmer

For P2P-systemene nedenfor gjelder søk etter filnavn. Ingen funksjonalitet for søk i innholdet av filene er implementert.

Gnutella [11] var den første P2P-tjenesten som brukte *flooding*. Flooding er veldig enkel å implementere og fungerte godt for små Gnutellanettverk. Problemer oppstår når nettverket blir stort. Effektiviteten går ned og klienter bruker mye av ressursene på å overføre og prosessere søkemeldingene [31].

Kazaa [12] bruker en modifisert form for *flooding*. Alle søkemeldingene sendes til spesielle supernoder. Disse nodene tar seg av det meste av belastningen med søk. Flooding med supernoder er også inneffektiv. Supernoder får for mye av belastningen og søkefunksjonaliteten reduseres [31].

Napster [14] brukte en eller flere sentraliserte tjenere. Klientapplikasjonen ga ett enkelt brukergrensesnitt for søk. Søket ble sendt til tjeneren hvor indekser over andre klienters ressurser ble søkt igjennom. Søket ble så levert tilbake med pekere til noden i nettverket med dokumentet. Napster i dag er en musikk-kjøpstjeneste hvor både indeksen og dokumentene er lagret sentralisert.

3.1.2 IR i Freenet

Det finnes tre forskjellige muligheter for å finne informasjon i Freenet.

- Katalognavigering
- Søk i dokumenteres signed-subspace
- Søk etter dokumenter med lik hashnøkkel.

Brukeren kan selv navigere i spesielle katalogsider i Freenet. Publiserte dokumenter kan publiseres på slike sider og lenke til en nøkkelverdi for dokumentet. Hentingen av Freenetsider gjøres fra nettverket og er ganske tregt. Sidene er lenket sammen og brukeren kan navigere til sider med interessant innhold. Sidene er representert på samme måte som sidene på Internett og har klikkbare lenker. Katalognavigering er bra for å finne dokumenter om brukeren ikke vet mye om temaet. God navigering forutsetter organisering og struktur noe som bare i en viss grad er implementert for Frenet. De fleste sidene er diversesider og er derfor vanskelig å navigere. Ved publiseringen av en lenke til et dokument ligger det et sammendrag eller beskrivende tekststreng. Hvordan informasjonspubliserer beskriver dokumentet (hvilke ord som brukes og språket det er skrevet på) begrenser hvor enkelt det er for informasjonskonsumenten finne relevante dokumenter.

Søk i *signed subspace* tekststrenger er også begrenset av at det bare er en liten beskrivende tekststreng. For eksempel kan informasjonskonsumenten søke etter "etikk AND personopplysninger" og det mest relevante dokumentet har en *signed subspace /POLITIKK/privatinformasjon/privatsfære*. For dette eksemplet vil ikke det mest relevante dokumentet leveres tilbake på grunn av dårlig beskrivende streng. Bruk av *signed subspace* er også begrenset av hvor mye informasjonskonsumenten orker å skrive ned. Generelt kan det sies at

informasjonsdelingstjenester må være enkle å bruke. Dette gjelder for både informasjonskonsument og informasjonspubliserer. Å skrive inn en beskrivende tekststreng er ikke trivielt både i ordvalg og tidsbruk.

Søk kan også gjøres etter dokumenter hvor brukeren ikke har en nøkkelverdi. Freenetklienten kan generere en hashverdi av søkeordene og søke etter denne. Nøkkelverdiene til dokumenter er basert på en beskrivende streng av ord om dokumentet. Den genererte hashverdien av søkeordene kan brukes til å levere dokumenter med liknende hashnøkler. Å bruke hashnøkler for å søke etter dokumenter kan fungere på dokumenter som film, musikk og bilder hvor metadatainformasjon kan genereres automatisk og ikke er så omfattende. For tekstdokumenter vil en hashverdi av sammendraget vanskelig stemme overens med to-tre søkeord om dokumentet.

3.2 Kritiske mangler

P2P har til dags dato ganske liten støtte for avansert IR. Anonymiserte systemer er enda mindre lagt opp for implementering av rangering, søk i hele dokumentmengden, beregning av recall og presisjon. Ingen av systemene omtalt i denne avhandlingen har søkefunksjonalitet for tekstinnhold i dokumentene. P2P-systemer i utbredt bruk, brukes i all hovedsak til deling av musikk, programmer og film.

Oppsummering av mangler ved dagens P2P systemer angående IR:

- ingen søk i dokumentinnhold.
- for mye distribuerte søk fører til ineffektive algoritmer.
- for mye distribuerte søk gir ikke mulighet for søk i hele dokumentsettet.
- klassiske IR-systemer er ikke beregnet for distribuert bruk.
- *flooding* fører til for mye meldinger.
- *Random Walk* fører til for få og for snevre søk.
- søk i anonymiserte P2P-systemer har enda mindre søkefunksjonalitet enn vanlige P2P-systemer.
- det finnes muligheter for å forbedre belastningen søk påfører nettverket.
- eksisterende system er ikke satt opp for å støtte avansert IR funksjonalitet.

Kapittel 4

Løsningsforslaget

> *Whenever a theory appears to you as the only possible one, take this as a sign that you have neither understood the theory nor the problem which it was intended to solve*
>
> Karl Popper

For å muliggjøre indeksering og søk i anonymiserte distribuerte systemer presenteres distribuerte anonymiserte indekseringstjenere. Indekseringstjenere skal effektivisere søkeprosessen og øke kvaliteten på søkeresultatet.

Tjenerne skal kunne passe inn i P2P-systemer lik Freenet. Ideen er å lage en applikasjonsmodul for klientene, slik at de frivillig kan sette opp en indekseringstjener. Indekseringstjenerene skal ta seg av mye av belastingen distribuerte søk påfører nettverket.

Løsningsforslaget består i en tjenermodul hos hver node med aktivert tjener, en algoritme for å koble nettverket til tjenerne (gjøre nettverket oppmerksom på at tjeneren finnes), en algoritme for søk over distribuerte tjenere, routing fra node til tjener og en publiseringsalgoritme for dokumenter.

I dette kapitlet presenteres design og kravspesifikasjoner. For å kunne gi interessante teoretiske observasjoner må designet av systemet følge et visst sett med spesifikk krav. Noen av kravene reflekterer kravene i et virkelig system. Andre krav reflekterer behovet for å forbedre kritiske mangler ved tidligere systemer. Krav er også satt for å gi god effektivitet, skalerbarhet og kvalitet for søke og indekseringsløsningen.

Kort oppsummert er designet for systemet å legge et lag av tjener på toppen av nettverket. Klienters kontakt med tjenere er abstrahert bort med en *vei til tjener* til en annen node. Tjenere utfører mottak av indekser, søk i egne indekser, distribuering av søk til andre tjenere og samling av svarsett for levering til klienter.

4.1 Kravspesifikasjoner

Et anonymisert distribuert nettverk har svært mange flere begrensninger enn andre peer-to-peer systemer. Enhver løsning for søk og indeksering i slike systemer må ikke kompromittere anonymiseringen. All informasjon brukt for identifisering av klienter eller indekseringstjenere må fjernes fra løsningen. Likevel må indekseringen og søk kunne være tilgjengelig for hele systemet. Anonymiseringen begrenser svært bruken av IP-adresser. Dette fører til at tjenere må kobles til nettverket uten å gi fra seg lokaliseringsinformasjon. Løsningen må implementeres for å ta hensyn til alle begrensningene og levere bedre resultater i form av skalerbarhet, kvalitet og effektivitet.

Nye algoritmer og forslag til applikasjonsmoduler må holde seg innenfor rammene kravspesifikasjonen gir. Det stilles også krav til måten løsningen testes ut. Kravene kobles opp til Freenet siden dette systemet er en av de mest avanserte distribuerte anonymiserte systemene. Freenet stiller til rådighet mye underliggende funksjonalitet et løsningsforslag kan benytte seg av.

Nedenfor følger kravene løsningen skal følge:

Nettverk: Lage ett randomisert sammensatt nettverk som simulerer P2P nettverk slik som Freenet. Nettverket av klientnoder simulerer Freenet ved å sette noder sammen randomisert. Freenet krypterer pakker i nettverket, men denne tjenesten er tilgjengelig for alle noder i nettverket og er ikke tatt med i simuleringen siden en indekseringstjener uansett kan benytte seg av de eksisterende funksjonene.

Anonymisering: Simulere begrensningene anonymiseringen pålegger nettverket. Noder kan bare vite om nærmeste nabo og lager tilkoblinger til andre noder med en vei gjennom nettverket. For klientene og tjenerne finnes det ingen mulighet til å tilegne seg global kunnskap om nettverket. Anonymisering er strippet ned til routingdelen av anonymiseringsteknikkene. Kryptering er en tjeneste allerede tilgjengelig for klient. Publisering av nøkler og håndtering av disse er også en tjeneste antatt tilgjengelig i systemene.

Oppretting av tjenere: Tjenere må opprettes i noder og gi seg til kjenne for nettverket. Tjenere skal være en del av noden. Hver node har en klient del. Noder kan også sette opp en indekseringstjener. Tjeneren er en modul lenket til klienten og all kommunikasjon med nettverket skal skje gjennom klienten.

Struktur for overføring: Nettverket må kunne sende meldinger fra node til node. Denne tjenesten er strippet ned til sending av meldinger til og fra tjenere i nettverket.

Indeksering hos klient: Indeksering av lokale dokumenter skal skje i klienten. Klienten må utføre en indekseringstjeneste. Siden belastningen for en tjener allerede blir stor kan selve indekseringen av dokumentene legges hos klienten. Indeksen må være på et lesbart format og kunne sendes for tillegging til indekseringstjeneren.

Indeksering hos tjeneren: Tjeneren må ta imot og legge til indekser i sin indeksbase. Fra klientene vil tjeneren motta tillegg til den eksisterende indeksen. Disse indeksene må legges til og distribueres videre til andre tjenere.

Distribuert søkefunksjonalitet: Søkefunksjonalitet skal være tilgjengelig for alle noder. Alle klienter må ha muligheten til å sende søk til en tjener. Tjeneren skal gi tilbake et søkeresultat basert på informasjon ikke bare fra seg selv, men fra flere tjenere. Tjeneren som mottar søket fra en klient har til ansvar å sette sammen søkeresultatene fra de andre tjenerne og levere søket tilbake.

Routing til tjener: Alle klienter må kunne sende meldinger til ukjente tjenere. Klienter i nettverket skal kunne sende melding til tjenere. For klientene skal indekseringstjenesten være abstrahert som ett lag av tjenere. Klienten skal kunne sende *tiltjener:melding* ut i nettverket. Meldingen skal routes mest effektivt til tjeneren ved bruk av *vei til tjener*.

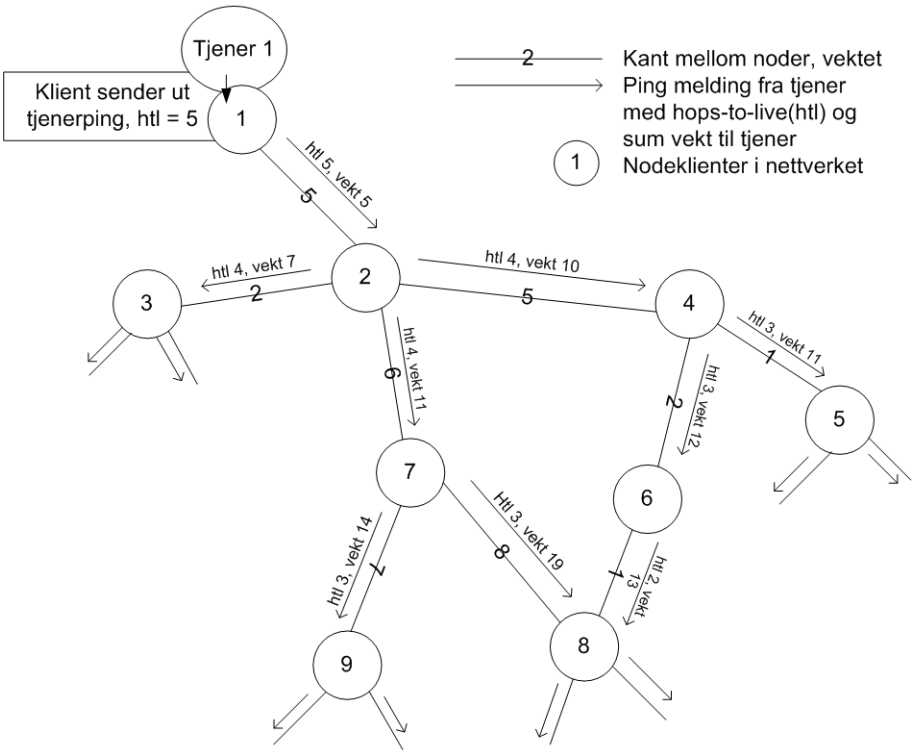
Testfunksjonalitet: Automatiserte tester. Disse skal produsere viktig informasjon om oppførselen av systemet gitt flere forskjellige størrelser av nettverk og antall tjenere. Testresultater som skal produseres er:

- Gjennomsnitt hopp på vei gjennom nettverket fra klient til tjener
- Worst-case hopp på vei gjennom nettverket fra klient til tjener
- Best og Worst-case antall returnerte dokumenter.

Best-case hopp til tjener er alltid 1 siden nabonoden til tjeneren vil returnere denne verdien og er uinteressant. Denne informasjonen skal kunne settes opp i mot de tre viktigste variablene for nettverkene. Variablene er nettverksstørrelsen, antall tjenere i nettverket og antall noder i routingtabellen for klientene.

4.2 Design

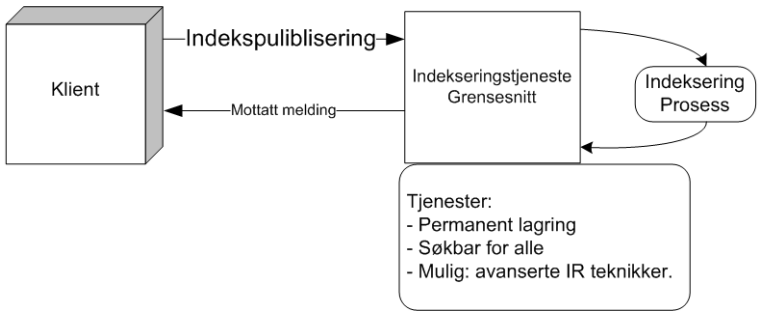
For å gjøre alle klienter i nettverket oppmerksomme på at tjeneren eksisterer skal tjeneren sende ut en ping"melding til nettverket. I figur 4.1 ser vi at tjener 1 aktiveres og klientdelen av noden starter pingalgoritmen. Pingmeldingen har *htl* verdi og latency verdier. Meldingene sprer seg utover i nettverket i dybde *htl*. Latency eller vekten til tjener (den sammenlagte belastningen ved å nå tjener) legges til ved hver node. I figur 4.1 ser vi at den korteste veien ikke alltid er i antall hopp, men kan være den minste latencyen fra noden til tjener. Node 8 har fått to pingmeldinger fra samme tjener og velger den med minst latency, meldingen som kommer fra node 6. Pingmeldingen node 8 mottar fra node 7 er større enn melding den mottar fra node 6, begge pingmeldingene sendes videre. Dette er fordi node 8 ikke har muligheten til å stadfeste at begge pingmeldingene kommer fra den samme tjeneren.



Figur 4.1: Ping algoritme

Når en node vil publisere et dokument blir en indeks generert av filen hos publiserernoden. Denne indeksen vil sendes ut i nettverket til en tjener. Publiserernoden for indeksen vet ikke hvor tjeneren ligger og sender til den noden i routingtabellen registrert som nærmest tjeneren basert på tidligere

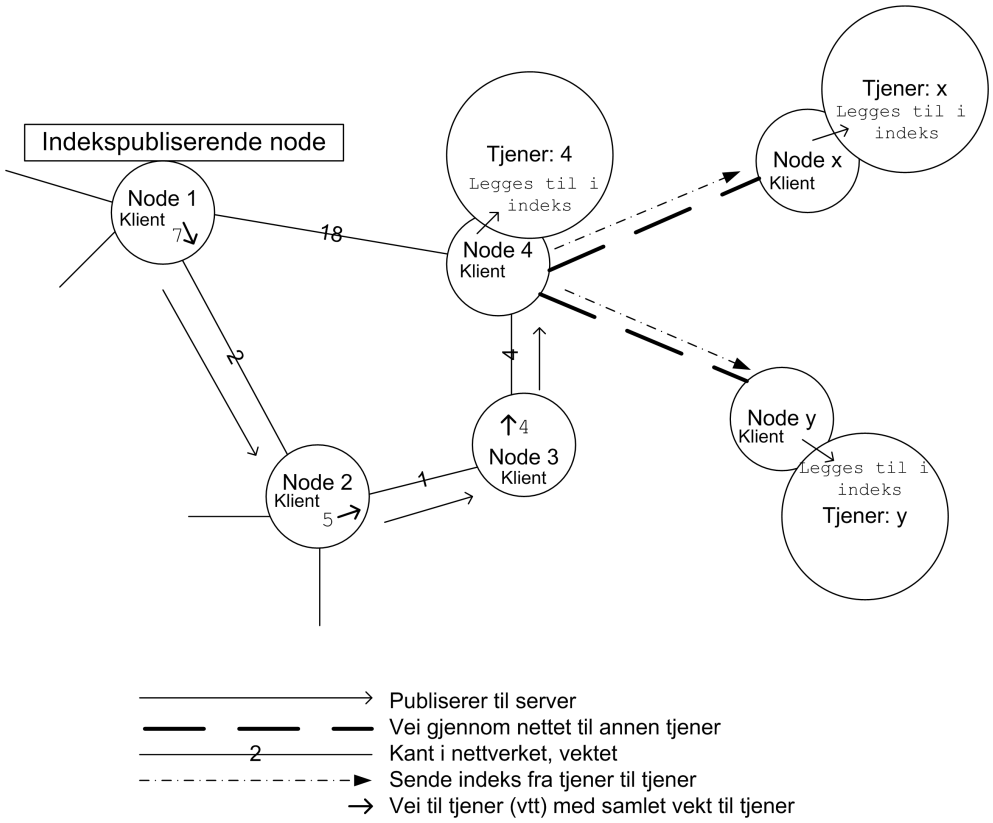
pingmeldinger fra tjeneren. I figur 4.2 vises hvordan indekspublisering abstraheres for klienten fra en bestemt node til nettverket generelt (ved sending til noden registret som *vtt* [1]).



Figur 4.2: Indekseringstjeneste abstrahert for klienten

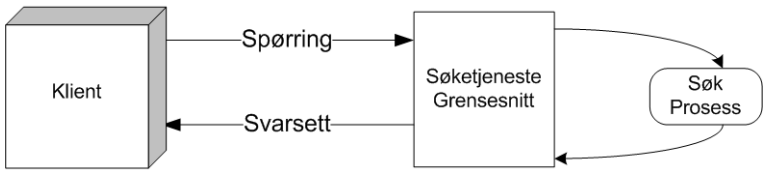
Figur 4.3 gir et eksempel på indekseringstjenere. Node 1 velger den noden i routingtabellen hvor veien til tjener er registrert som minst. Node 1 har node 2 som *vvt*. Indeksen sendes til node 2 og fra node 2 til node 3. Node 3 har node 4 som *vtt*, men vet ikke at tjeneren finnes i nabonoden. Node 4 tar så imot indeksen og legger denne til i sin indeks. Den samme indeksen distribueres så videre til ett gitt antall andre tjenere tjener 4 vet om.

[1] *vtt* er den noden registret som nærmest tjener. Noden er veien-til-tjener (vtt)



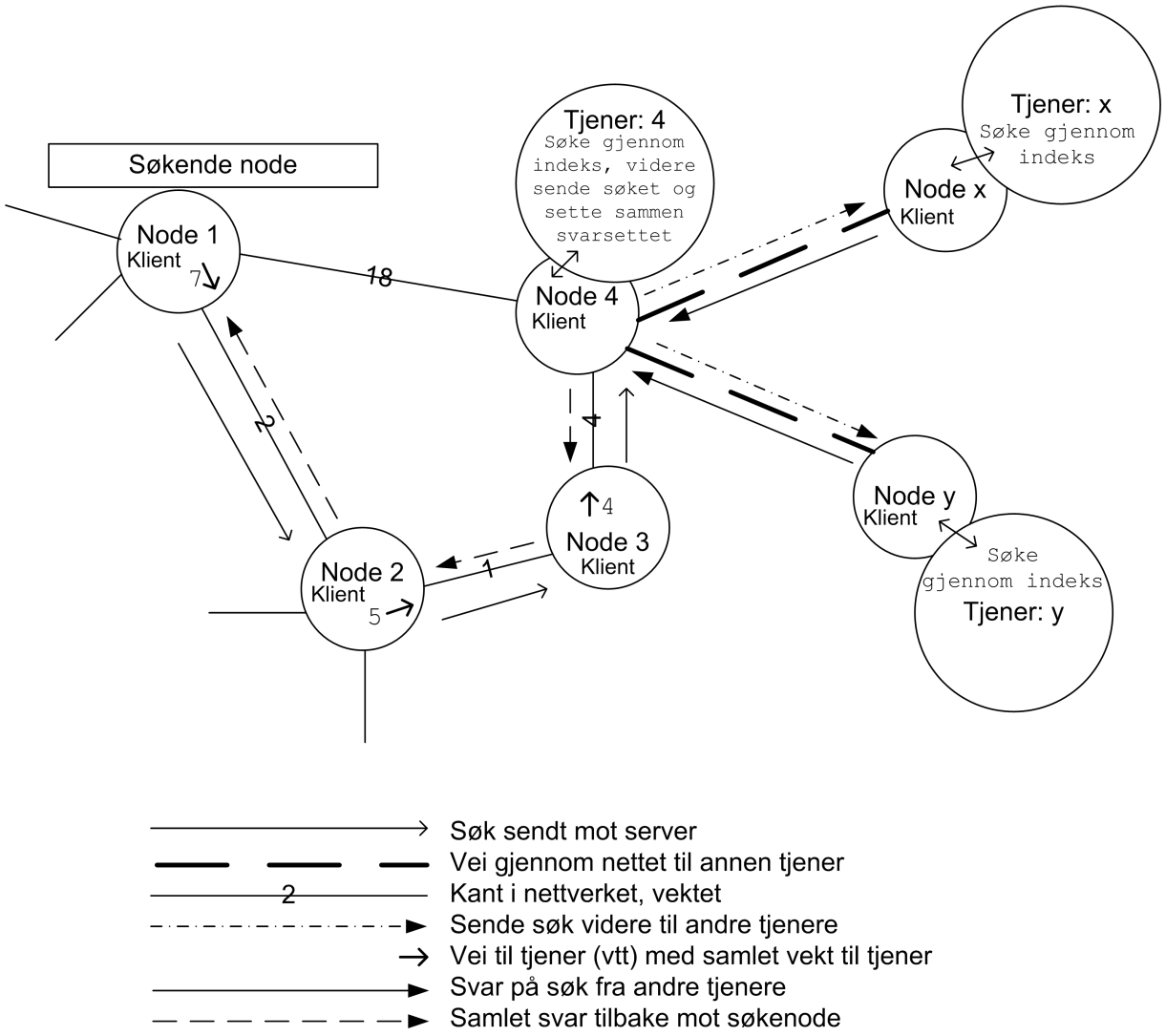
Figur 4.3: Indeks publisering og distribuering

I figur 4.4 vises det hvordan søketjenesten abstraheres for klienten. For å søke sender klienten ut en spørring til *vtt* noden og forventer ett svarsett tilbake fra den samme noden.



Figur 4.4: Søketjenste, abstrahert for klient

Figur 4.5 viser hvordan det distribuerte søket skjer i nettverket. Node 1 sender søket til den noden med minst *vtt* (i dette tilfelle node 2 med 7 i latency til tjener). Søket sendes på denne måten fren til node 4. Klientdelen i node 4 konstaterer at den har en aktivert tjener og søket sendes til tjenerdelen av noden. Tjeneren søker igjennom sin egen indeks og sender melding til klienten om å sende tjener til tjener søk til tjener x og tjener y. Svarene fra tjener 4, x og y samles og settes sammen i tjener 4. Tjener 4 leverer søkeresultatet til sin klientdel og routingen av svarsettet tilbake til node 1 går gjennom node 3 og 2. Node 2 og 3 sender automatisk tilbake den veien meldingen kom fra. Routing tilbake i nettverket er en tjeneste gitt av den underliggende algoritmen i det distribuerte systemet.



Figur 4.5: Distribuert Søk

Kapittel 5

Eksperiment

So then one disconnected thing led to another disconnected thing, and that's how all kinds of stuff happened.

Haruki Murakami [45]

Den falt seg naturlig å simulere nettverket, indekseringsfunksjonalitet og routing-funksjonalitet for å avdekke positive og negative sider ved løsningsforslaget. Clarke et. al [29, 28] simulerte Freenet for å teste ut den løsningen de hadde utformet. Anonymiserte systemer som Freenet er laget for å vanskeliggjøre observasjon så uttesting av ny funksjonalitet er vanskelig i et virkelig system. Andre forslag til forbedringer av Freenet baserer også konklusjonene sine teoretiske analyser av resultat fra simuleringer [42] [56]. Derfor er simulering av et distribuert nettverk likt Freenets design valgt for å teste ut distribuerte indekseringstjenere.

En teorietisk analyse av et simulert system er tenkt å gi god informasjon om hvordan distribuerte tjenere vil oppføre seg og gi pekepinner om forbedringspotensial. En simulering vil redusere kompleksiteten. Fordelen med å redusere kompleksiteten er større oversikt, men ulempen med redusert kompleksitet er muligheten for å teste med feil antagelser.

Løsningen har til hensikt å vise hvordan avansert indeksering og søkefunksjonalitet kan implementeres for distribuerte anonymiserte systemer. For å oppnå dette må en ny bunnstruktruren konstrueres hvor avansert IR enkelt kan legges over. Fokuset for simuleringen er å vise hvor effektiv distribuerte tjenere er og vise til hvor mange dokumenter distribuerte søk returnerer satt opp i mot hvor mange noder det søkes gjennom. For å kunne gi interessante teoretiske observasjoner må simuleringen på best mulig måte simulere ett virkelig anonymisert distribuert system.

5.1 Valg av parametere

Hensikten med den programmerte simuleringen er å avdekke feil, mangler og uventede resultat ved uttesting av løsningsforslaget. Alle parametere er valgt ut for at simuleringen skal ha samme funksjonalitet og begrensninger som et anonymisert distribuert nettverk. I tillegg er parametrene de variablene simuleringen skal bruke for uttesting.

- Latency (treghet eller belastning) hos noder
- Antall noder i systemet (skalering)
- Antall noder i routingtabellen hos nodene
- Antall tjenere i systemet
- Antall dokumenter som skal indekseres og spres
- Søkeord brukt i uttesting

Latency. For å kunne lage en optimalisert rute fra en hver node til en tjener trenges informasjon om hvor mange hopp det er fra klient til tjener og så i tillegg hvor lang tid det tar. I simuleringen finnes en algoritme som tar hensyn til både lengden i antall hopp og i hvor langt i latency det er fra klient til tjener. For simuleringen måtte ett sett med tall velges ut som skulle simulere hvor lang tid det tar for å sende en pakke fra en node til en annen. Alle noder i nettverket har fått en latency verdi og verdien er valgt tilfeldig ut fra 1 - 100 millisekunder. Disse verdiene reflekterer naturlig variasjon hvis nodene i nettverket er spredt rundt i hele verden.

Noder i systemet For å teste om løsningsforslag skalerer er nettverksstørrelsesverdier valgt til 10000, 20000, 40000, 60000, 100000 og 150000. Systemet er også testet med 200000 noder for noen få verdier. Nettverksstørrelsene er valgt for å være fornuftige størrelser for en slik type nettverk og for å gi en pekepinn på hvor godt løsningsforslag skalerer.

Noder i routingtabellen I simuleringsnettverket er antall noder i routingtabellen satt til 20. Antageligvis vil noder i et dynamisk nettverk ha forskjellige antall noder i routingtabellen. Grunnen til varierende routingtabell størrelse vil være for eksempel at noder som var i tabellen logger seg av eller at tabellen ikke er fylt opp enda. 20 noder i routingtabellen er lite, Zhang [56] operer med 240 noder i routingtabellen ved uttesting av forbedringsalgoritmer for Freenet. For å teste ut vei til tjener er en liten routingtabell ikke svært viktig. Om løsningsideen virker for 20 noder i routingtabellen vil den

virke veldig mye bedre for nettverk hvor nodene har 240 noder i routingtabellen (innlysende at hvis hver node kjenner til flere noder vil den potensielle veien til tjener være kortere).

Tjenere i systemet Løsning for indeksering og søk i anonymiserte nettverk gir alle noder muligheten til å fungere som tjener. Det er umulig å finne ut hvor mange tjenere det vil finnes i ett virkelig nettverk (siden det er brukere med mye prosesseringskraft og lagringsplass selv må ta valget om å *sette opp en tjener*). I simuleringer er forskjellige tjenerantall testet ut. Antallet tjenere er antatt å påvirke hvor godt og kjapt et nettverk greier å levere en indekserings- og søketjenesten til brukeren. Ved simuleringen er det brukt brukt 8 forskjellige antall tjenere. Disse er 4, 8, 40, 100, 200, 300, 500 og 1000.

Indekserte dokumenter Det viktige for uttesting av prtotyp er ikke å test ut reell IR-funksjonalitet. 3327 dokumenter på 20 linjer hver brukt som dokumentsett. Dokumentene er generert av en public-domain bok lastet ned fra Gutenbergprosjektet [1]. Siden indekseringsdelen bare skulle reflektere hvor god recall løsningen gir var det naturlig å ikke bruke svært lang tid for å få tilgang til et testsett av dokumenter. Løsningens recall kan lett sammenliknes ved å bruke søk i Windows eller Google-Desktop siden alle dokumentene er på vanlig tekstform (.txt).

Søkeord bruk i uttesting Søkeordene er tilfeldig valgt ut og har både mer og mindre søkesvar enn det satte taketfor svarsettet. En tekstfil innholder 33 søkeord.

5.2 Implementer Prototyp

Simuleringsprototypen er programmert i Java(SE 1.5) [43]. Selve prototypen består av 11 klasser hvor den ene leverer et svært enkelt tekstbasert brukergrensesnitt. De viktigste delene er NodeKlient, NodeTjener og testalgorimene. Disse skal til sammen simulere et anonymisert nettverk og teste kvaliteten på løsningen med distribuerte indekseringstjenere.

De syv viktige algoritmer i systemet:

[1] Gutenbergprosjektet en samling bøker og dikt hvor opphavsretten har gått ut. `www.gutenberg-project.com`

5.2.1 Nodegenerering

Denne algoritmen lager alle nodeobjektene som skal simulere en enkelt klient i ett distribuert system. Nodene instansieres med navn og en *default* nabo blir lagt til. Hver node av typen NodeKlient instansieres med en NodeServer tilkoblet. Pseudokoden i 5.1 viser konseptene i nodegenereringsalgoritmen.

```
lagNoder(antallNoder){
    opprett nodeliste;
    teller=0;

    while(teller<antallNoder){
        if(nodoeliste.size() ==0){
            lag node nr 1;
            teller++;
        }
        else{
            lag ny node med navn ”nodenr”+teller;
            legg til i listen;
            koble ny node til forrige node;
        } // end else
    } // end while
renturn nodeliste;
} // end makeNodes
```

Figur 5.1: lagNoder algoritme

5.2.2 Legge til tilfeldige noder til i routingtabellen

Denne metoden tar inn en liste av NodeKlient-objekter(listen over alle nodene i nettverket) og en routingtabell størrelse. Routingtabellen fylles opp med tilfeldige NodeKlienter (unike) valgt ut fra NodeKlientlisten. Algoritmen i figur 5.2 er basisen for hele nettverket og simulerer den kaotiske måten et distribuert nettverk er koblet sammen på.

Routingtabellen gir noden mulighet til å kalle metoder i andre noder. Kallet av metoder i de andre nodene simulerer sending, mottak og svar på meldinger i Freenet. Routingtabellen har mindre funksjonalitet enn den virkelige routingalgoritmen i Freenet, men den funksjonaliteten simuleringen trenger for å teste ut distribuerte tjenere finnes.

```
leggNodertilTilfeldigiRT(nodeliste, antnoderiRT){
// RT = routingtabeller
teller = 0; i =0;
    while(i<antnoderiRT){
        while(teller<nodeliste.size()){
            tempnode = hent node fra liste(teller);
            generer random int;
            ikkeSant = true;
                while(ikkeSant){
                    hent random node;
                    if(randomnode allerede i RT){
                        generer ny random int;
                        hent ny random node;
                        }
                    else{
                        legg til node i routingtabell;
                        ikkeSant = false;
                        } // end else
                } // end whil ikkeSant
        teller++;
        } // end while teller
    i++;
    } // end while i
} // end leggNodertilTilfeldigiRoutingtabeller
```

Figur 5.2: generer tilfeldig routingtabeller

5.2.3 Ping algoritme for tjenere

Ping algoritmen kobler noder og tjenere sammen. Før noen tjenere er lagt til nettverket har ingen noder en *vei-til-tjener (vtt)*. NodeKlientene har ingen vtt hvor den kan sende søk eller indekser. Når en tjener aktiveres, i en node, kalles denne metoden for å gjøre nettverket oppmerksom på at tjeneren er til stede. Ping funksjonen sender en "*tjener denne veien!*" melding til tjenerens nærmeste naboer. Meldingen bruker en "hops-to-live" (htl) for å kontrollere hvor langt ut i nettet meldingen penetrerer. Meldingen sender også med hvor stor belastning (latency) denne *vtt* har. Når en node tar i mot en tjenerpingmelding vil den sjekke om latency er mindre til denne tjeneren enn den som allerede eksisterer. Om latency i tjenerpingmeldingen er mindre enn den eksisterende eller om nodeklienten ikke har en *vtt* så legges den noden som har sendt tjenerpingmeldingen til som *vtt*. Hvorvidt den er mindre eller større sender noden meldingen videre. For å holde på anonymiteten oppfører tjenere seg på samme måte som noder. Noder vet aldri hvor tjeneren ligger, bare i hvilken retning (hvilken node i routingtabellen, *vtt*) meldinger skal sendes for å nå den nærmeste tjeneren.

Algoritmen vil varsle $(routingtabellstorrelse^{htl} - antalldublisertemedlinger)$ antall nodeklienter i nettverket. Antall dubliserte medlinger vil variere med antallet små sykler i nettverket. Bedre tilkobling i nettverket vil gi muligheten for bedre veier til tjener. Dette fører med seg at pingalgoritmen vil levere kortere vei til tjener om nodene har bedre sammenkobling (har større routingtabell).

```
goTjenerPing(hopstolive, innNode, latency)
    if(hopstolive>0){
        for(0 → routingtabellstørrelse){
            if(ikke innNode){
                node.getServerPing(hopstolive, denneNode, latency)
                }
            }
        } // end goServerPing

getTjenerPing(hopstolive, innNode, latency )
    if(eksisterende latency til tjener > latency){
        innNode = vei til tjener;
        denneNode.goTjenerPing(hopstolive-1, innNode, (latency+denneNodeLatency));
        }
    else{
        denneNode.goTjenerPing(hopstolive-1, innNode, (latency+denneNodeLatency));
}
```

Figur 5.3: Ping algoritmene

Figur 5.3 algoritmen består av to metoder. Disse metodene kaller hverandre semi-rekursivt. Den ene metoden kaller den andre metoden i alle noder i routingtabellen. Kalles semirekursivt fordi begge metodene har et *base-case* og metoden kaller ikke seg selv, men den samme metoden i et annet objekt. GoTjenerPing kaller getTjenerPing og omvendt i mange iterasjoner over alle node-objektene i nettverket.

5.2.4 Distribuert søk

Simuleringen bruker en semi-rekursiv metode for å sende søket fra søkenoden til tjeneren (semi-rekursivt fordi en node kaller den samme metoden i en annen node hvor base-case er at noden har en aktivert tjener). Når en node skal søke etter et ord sender den søkEtter til noen i retning av tjeneren. Figur 5.4 viser hvordan søket sendes videre fra hver node som selv ikke er tjener og gjøres helt til noden som har aktivert tjener mottar søket og sender det videre til sin tjener.

I figur 5.5 vises det hvordan tjenermodulen til mottakernoden søker igjennom sin egen indeks. For alle søk er det satt et tak på søkesvar. Dokument returnerings-*taket* er en verdi på hvor mange svardokumenter tjeneren trenger før den returnerer svarsettet. Tjeneren som initielt tar imot søket søker igjennom sin egen indeks. Om dokumentmassen ikke er høyere enn *taket* sendes søket videre til en annen tjener. Tjeneren samler så inn søkeresultat fra maks

```
søkEtter(søkestreng)
        if(aktiv tjener){
                send søk til tjenermodul;
                returner svar fra tjenermodul;
                // returnerer svaret mot søkeklienten
        }
        else{
                send søk videre mot tjener
        }
```

Figur 5.4: Søkealgoritme fra tjener

halvparten av de andre tjenerne og returnerer svaret til søkenoden. *Taket* i min prototyp er satt til 150. Brukere av web-søk som Google, Yahoo og MSN lager ett nytt søk om svarene på den første siden ikke er relevante nok. I prototypen finnes ikke noe rangeringssystem for filene og satt derfor taket til 150. Ved rangering kan potensielt alle dokumenter settes sammen hos tjener og bare de beste sendes som svar til søkeklienten. For uttesting av systemet er det brukt både søkeord med mindre og mer enn 150 relevante svar. Tjeneren sender søket videre til maksimalt halvparten av indekseringstjenerne i sin liste over andre tjenere. Detter gjøres ved whilesløyfen i figur 5.5. Indeksene distribueres og det ikke er naturlig å tenke seg at alle tjenere i et virkelig nettverk vet om alle andre tjenere. Svarsettet returneres til søkenoden selv om *taket* ikke er nådd om halvparten av de andre tjenerne også er søkt igjennom. I systemet er det to typer søkealgoritmer. Den ene kan klienter kalle hos tjener og den andre som fungerer svært likt kan tjenere kalle hos andre tjenere. Søkealgoritmen for tjener til tjener er lik den i figur 5.5, men sender ikke søket videre.

```
søkEtterOrd(søkestreng)
        for(0 til størrelsen på indeksen ){
            if(søkeord == ord i indeksen){
                legg til dokumenternavn i svarlisten
                }
        }


        while(ikkeferdig){
            if(ikke nok dokumenter å returnere){
                send søk til annen tjener;
                legg til svar fra tjener i svarlisten;
            if(nok svar i svarlisten){
                ikkeferdig = false;
                }
            if(distribuert til halvparten av de andre tjenere){
                ikkeferdig = false;
                }
            } //end if

        }//end while
        returner svarliste til klienten;
} // end søkEtterOrd
```

Figur 5.5: Søk i tjener og distribuering av søk

5.2.5 Indeksering og tillegging av indeks

```
publisereIndeks(innindeks){
    if(denne noden har aktiv tjener){
        send indeks til tjener
        }
    else{
        node på vei mot tjener.publisereIndeks(innindeks)
        }
}
```

Figur 5.6: Publisering av indeks

Tjenere i den denne typer system tar seg av mye av belastningen for hele systemet. En måte å distribuere tyngden av brukt CPU tid er å distribuere deler av indekseringen til nodene. Nodene tar seg av å lage en liste av ord som hver dokument inneholder. Denne listen sendes så til tjeneren sammen med ett navn på dokumentet. I det virkelige nettverket må nodene sende med en system-unik hash av filen. Lokaliseringsprosessen av dokumenter i Freenet gjøres etter en eller flere hash-nøkler. Siden dette er en funksjon nettverket kan benytte seg av brukes filnavnet for hver fil. Filnavnet er generert etter en enkelt hashfunksjon og representerer den type navn indekseringstjenesten skal knytte opp i mot søkeordene. Filnavnene er på formen *C+hash+.txt* .

Sendingen av indeks til tjener er helt lik funksjonaliteten i det distribuerte søket. Indeksen sendes semi-rekursivt nærmere tjener ved å sende videre mot den noden registrert som *vei til tjener*. Noden med aktivert tjener sender indeksen til sin tjenerdel for videre behandling.

Når tjenerdelen til en nodeklient tar i mot en indeks legges denne til den eksisterende indekslisten. Indeksliten består av en todimensjonal liste. Listen er dynamisk i begge retninger og blir automatisk alfabetisert ved innlegging. Enkelt forklart så er ordene første element horisontalt og alle elementene vertikalt. Alfabetiseringen skjer utelukkende vertikalt i listen. Etter første element horisontalt følger alle dokument identifikatorene (i simuleringen er disse tekstfil navn).

5.2.6 Søkealgoritme

Søkealgoritmen i tjenerdelen av systemet er en vanlig sekvensiell søkemetode. Siden fokuset for denne oppgaven ligger i å finne den beste topologien for nettverk som best støtter søk i anonymiserte distribuerte systemer er det valgt å ikke bruke for mye tid på å lage en veldig effektiv søkealgoritme.

Lagringen av indeksen er alfabetisert. Dette gjør det sekvensielle søket relativt enkelt. Siden en indeks aksesseres langt flere ganger enn det legges til struktureres denne i utgangpunktet. Ved søk sjekkes første element i listen sekvensielt til elementet er funnet. Det sekvensielle søket stoppes om algoritmen ikke har funnet et svar og verdien for ordet i listen er større en for søkeordet (sjekkes ved bruk av Sting.compareTo(String)). Når ordet er funnet i listen returneres hele listen som er koblet til søkeordet. Klausulen for at søkeresultatet returneres til søkenoden er at resultatet overgår et gitt antall svardokumenter. Om dette tallet ikke er nådd sendes søket videre til andre tjenere helt alle aktuelle tjenere er prøvd eller svardokumentene er mange nok.

Det finnes to forkjellige søkealgoritmer hos tjeneren. Den ene kan kalles fra klienter i systemet og den andre kan kalles av tjenere i systemet. Dette for å skille mellom søk til tjener og distribuert søk fra en tjener. Algoritmene leverer det samme, men tjener til tjener algoritmen distribuerer ikke søket videre.

5.2.7 Testalgoritme

Basert på verdier fra den virkelige verden skal testalgoritmen generere menneskelesbar informasjon om systemet. Testen leser inn to verdier. Verdiene er hvor mange tjenere det skal være med i testen og hvor mange noder hver node skal ha i sin routingtabell. Testalgoritmen generer så iterativt 7 forskjellige nettverk og genererer statistikk om oppførselen til nettverket gitt disse.

```
genererTestResultat(søkeord){
    for(0→ størrelse på nettverket){
        svarliste = node.søkEtter(søkeord)

        if(maxsvar<svarliste.size()) maxsvar = svarliste.size();
        if(minsvar>svarliste.size()) minsvar = svarliste.size();
        if(minstehopp> anthopp fra svarliste) minstehopp = anthopp fra svarliste;
        if(maxhopp < anthopp fra svarliste) maxhopp = anthopp fra svarliste;


        } // end for

Generere gjennomsnitthopp.
Skrive ut resultatene til fil med navn  ”søkeord+antallnoderinettverket”

} // end genererTestResultat
```

Figur 5.7: Algoritme genererTestResultat

De syv forskjellige nettverkene har forskjellige størrelser. 10000, 20000, 40000, 60000, 100000 og 150000. Tjenere blir aktivert i tilfeldige noder og disse gir seg til kjenne for nettverket ved bruk av ping algoritmen. Når det er aktivert nok antall tjenere blir dokumentene i dokumentmassen distribueres fra tilfeldig utvalgte noder. Recall og hvor mange hopp til tjener er de viktige parametrene testalgoritmen skal produsere. Testalgoritmen leser inn søkeord fra en tekstfil og søker så fra alle noder for hvert søkeord. For hver node lagres hvor mange søkeresultater hver node får tilbake og hvor mange hopp i nettverket søket må ta for å nå frem til tjeneren. For hvert søkeord genereres en gjennomsnittsverdi for antall hopp til tjener, gjennomsnittlig søkeresultat levert, worst-case og best-case levert søkeresultat. Disse lagres som tekstdokumenter for hvert søkeord med menneske- og maskinlesebar informasjon om hvor mange tjenere og noder testen er for.

```
Testing start i brukergrensesnittet:

if(input == teststart){
    input antall noder i routingtabell;
    input antall tjenere;
    listemedtall[] = 10000, 20000, 40000, 60000, 100000, 150000

    for(0 → 5){
        generer nettverk med listemedtall[i] størrelse;
        legg til noder i routingtabellen random;
        sett latency for alle noder;

        for(1→ antall tjenere){
            aktiver tjenere;
        } //end for antall tjenere
        test for brukbarhet;
        if( ikke brukbar) → opprett nettverk på nytt;
        else{
            søkeliste[] = hent søkeord fra fil;
            for(0 → søkeliste.length()){
                genererTestResultat(søkeliste[i]);
                } // end for søkeliste
        } // end else
        fjern nettverket for ”garbage collection”
    } // end for 5
} // end if teststart
```

Figur 5.8: Test Algoritmen

Kapittel 6

Analyse av Testresultat

All science is either physics or stamp collecting

Ernest Rutherford

Simuleringsprototypen har generert statistikk gitt flere variabler for nettverkene. De to viktigste variablene er størrelsen på systemet i forhold til antall tjenere i systemet og kvaliteten på recall i det distribuerte søket. Ved testing på flere verdier produserer simuleringen informasjon om hvor godt nettverket oppfører seg ved større belastning. Den andre og like så viktige parameteren for måling av simuleringsprototypen er gjennomsnittlig recall gitt antallet tjenere. Systemet er testet på den måten at for hvert søkeord så regnes det ut en gjennomsnittlig hopp til tjener og gjennomsnittlig recall. I tillegg legges det til informasjon om worst-case og best-case for de to.

6.1 Evaluering Ping og Routing algoritmene

Ping og routing algoritmene er de viktigste for oppsettet og senere vurdering av løsningsforslaget. Indekseringstjenere kobles til nettverket ved bruk av pingalgoritmen, og routingen til tjener skjer på grunnlag av informasjon gitt av pingalgoritmen.

6.1.1 Ping algoritmen til tjener

Ping algoritmen til tjener har fungert utmerket. Algoritmen gjør jobben med å automatisk lage korteste, kjappeste vei fra hver node i nettverket til tjener. Nodene vet ikke hvor i nettverket tjeneren er, men bare hvilken lokal node som er nærmest tjeneren. Pingalgoritmen i simuleringsprototypen tar ikke

hensyn til sykler i nettverket. Sannsynligheten for at det finnes ett stort antall sykler i nettverket er liten fordi nettverket er så kaotisk satt sammen. Hvor tallet av ping i sykler (dupliserte ping til noder) er *pis* og antall noder i nettverket er *antN* vil penetreringen for pingalgoritmen (hvor mange noder som får ping fra tjener) utrykkes ved $20^{ttl} - pis$. For å få full penetrasjon i nettverket så må $20^{ttl} - antN \geq pis$. Som en kommentar til ovenforstående utsagn skal det sies at algoritmen ikke kan produsere nye veier til tjener i sykler. Om den samme noden får flere ping fra tjener vil den automatisk velge den pingverdien med laves vei til tjener i form av forventet bruk av båndbredde til tjeneren. Effektiviteten til algoritmen kan måles på flere måter. Den første måten å se på algoritmen er hvordan denne pingfloodingen vil påvirke nettverket. Pingpakker er svært små og så lenge tjenere ikke kontinuerlig sender ping til nettverket vil ikke dette utgjøre noe problem. Et annet problem med ping i nettverket er om det finnes svært mange tjenere. Siden pingpakkene fjerner andre pakker fra nettverket implisitt (om det ikke er tjenere i nettverket må klienter bruke for eksempel flooding for å få like gode resultat) er algoritmen effektiv. Den andre måten å evaluere algoritmen er å se på hvor mye prosessorkraft algoritmen belaster klientene med. Det eneste klienten må gjøre er å ta imot ping-meldingen, sammenlikne denne med eksisterende *vtt* verdi og sende ping-meldingen videre med en mindre i *htl*.

En ekstra belastning på nettverket for å opprettholde kravet til anonymisering er utregning av fiktive belastningsverdier for første kanten. For at naboklienter til tjenernoden ikke skal vite at naboen er tjener må tjeneren sende med en liten fiktiv belastning oppå kanten fra tjeneren til naboen. Utregningen av denne fiktive verdien kan effektivt gjøres ved bruk av en tilfeldig verdi i ett visst spekter.

En annen belastning for nettverket er med hensikt feilsendte pakker fra tjener. Pakkeroutingen i Freenet sende med hensikt pakker feil og lenger en de skal for å gjøre det umulig å identifisere mottaker noden. På samme måte må tjenernoden automatisk sende videre flere av søkepakkene slik at det er vanskelig å definere hvilke noder som er tjenere selv om noen overvåker nettverket tjeneren er på. Nabonodene til en tjener vil få oppgaven om å sende videre og terminere slike pakker og dette kan sees på som en belastning for nodene.

6.1.2 Routing til tjener

Hovedhensikten med simuleringsprototypen var å teste ut om løsningsforslaget, med distribuerte tjenere, vil fungere i et kaotisk nettverk. Ping algoritmen er beskrevet i figur 4.1. Nodene har mulighet til å sende pakker i retning av

tjener. Det mest interessante er hvor effektiv overføringen fra søke-, publiseringsnode til tjener kan gjøres. At nettverket i Freenet har en IP lik protokoll lagt over IP for overføring av data har stor påvirkning på effektiviteten. Hvis man antar at Freenet overføringsprotokollen gir like gode resultater som IP protokollen vil forsinkelsen i nettverket være $IPforsinkelsen^2$. En indekseringstjenste i nettverket vil kunne effektivisere gjenfinningen av informasjon, men løser ikke problemet med forsinkelser i nettverket. Forsinkelsene kan i tillegg være positive i at det viser at pakker routes gjennom nettverket for dokument konsument og holders anonymitet.

Figur 6.1 viser hvordan *vtt* utvikler seg opp i mot hvor mange tjenere det finnes i nettverket. Grafen viser at antallet tjenere må økes svært eksponentielt for å redusere den gjennomsnittlige avstanden til tjener. I tillegg viser grafen at antall noder i nettverket ikke har svært stor innvirkning på hvor lang vei det er fra enhver node til en tjener. For eksempel for 1000 tjenere og 150000 noder i nettverket er det i gjennomsnittlig 4 hopp i nettverket fra en node til en tjener. I et nettverket med samme størrelse og bare 100 tjenere er *vtt* bare ett hopp lenger. I følge small-world prinsippet finnes den en kort vei mellom alle to individer i et sosialt nettverk. Selv om det er færre tjenere vil den gjennomsnittlige *vtt* ikke forandre seg svært mye. Større routingtabeller i nodene vil mye mer radikalt forbedre pingalgoritmen.

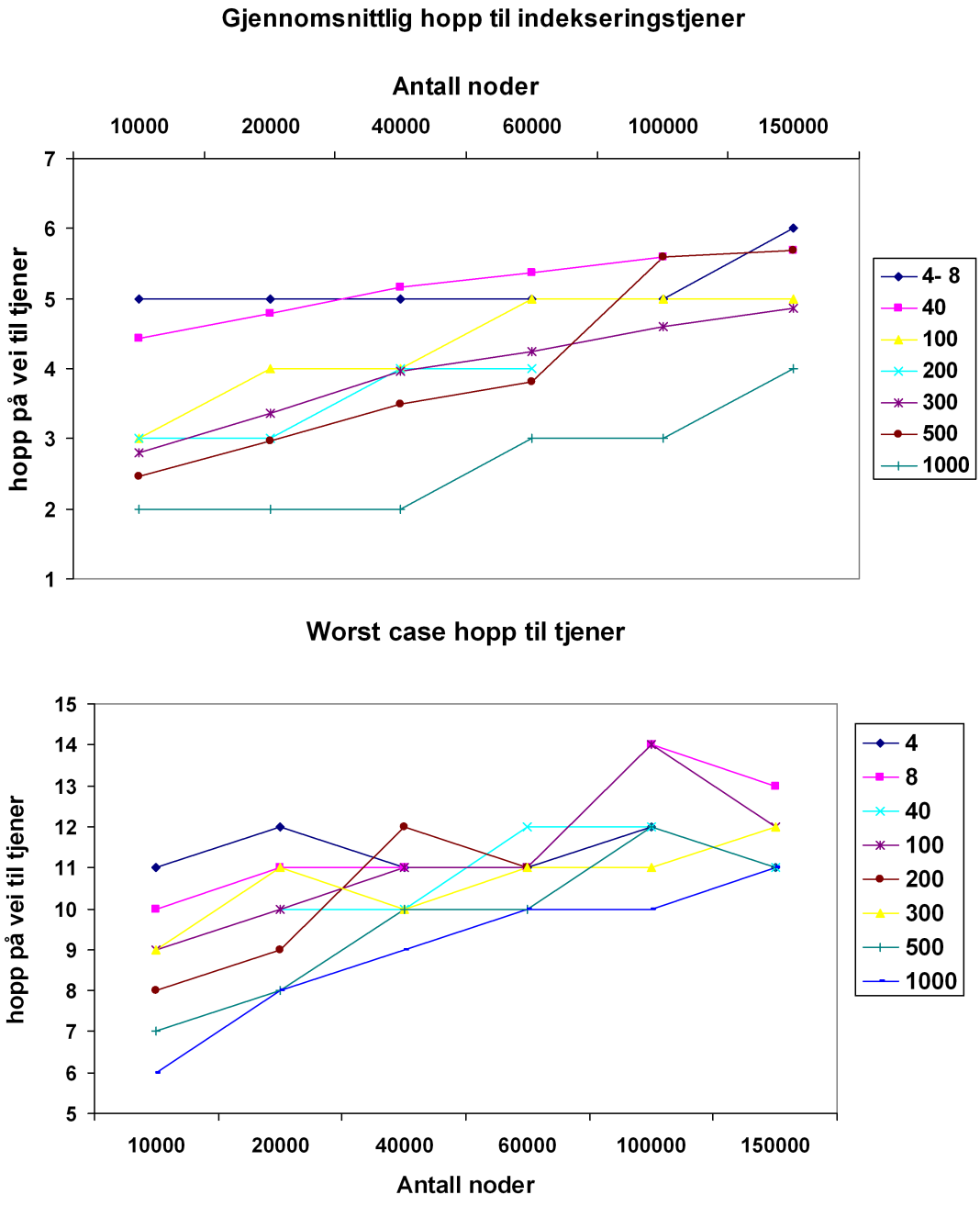
Topologien med distribuerte tjenere fungerer altså med svært få indekseringstjenere, men vil med eksponentiell økning i tjenerantallet levere bedre *vtt*. Informasjonen som skal overføres mellom tjener og klient er likevel ikke svært omfattende.

6.2 Recall i nettverket

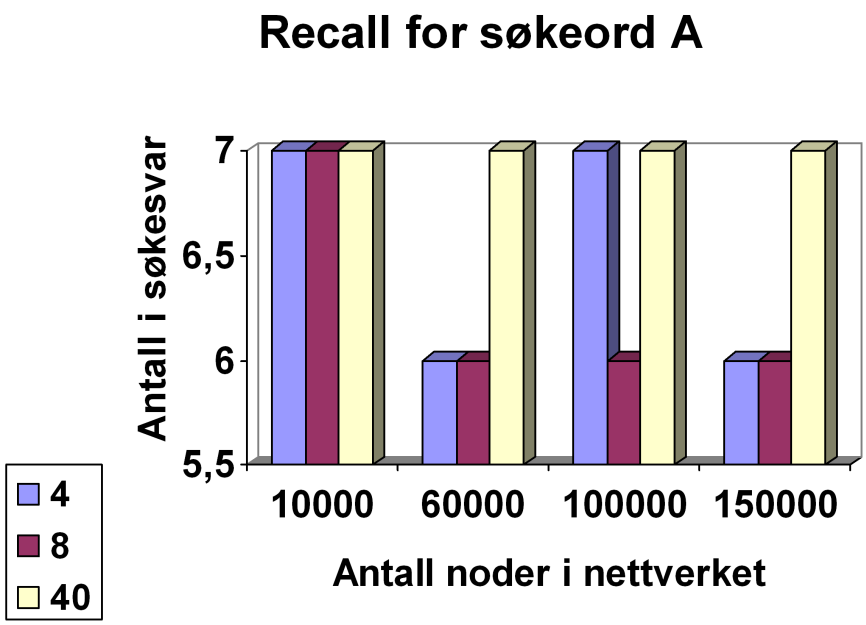
Recall for denne avhandlingen begrenses til hvor mange av alle rette dokumenter svarsettet består av. Recall i nettverket er testet ut med en maksimal svarsettverdi på 150 og tjenere distribuerer bare søket til halvparten av de andre tjenerene i nettverket.

Helt opp til 1 tjener per 500 klienter i nettverket (max på 300 tjener og 150000 noder i nettverket) leverer søket recall på 1. Alle dokumentene blir funnet fra søk fra alle nodene! Slik sett fungerer det distribuerte søket svært godt for slike verdier av tjenere og klienter, men lager en sørgelig kjedelig graf. For søkeord med mer enn 150 relevante dokumenter leverer søkealgoritmen 150 eller noen fler søketreff. Ut ifra søkealgoritmen 4.5 kan vi se at algoritmen kan levere mer en 150 søketreff, men bare noen flere.

For ord A i figur 6.2 finnes det bare 7 dokumenter av 3327 som inneholder søkeordet. For 4 og 8 tjenere med 60000, 100000 og 150000 noder leveres ett



Figur 6.1: Hopp til tjener



Figur 6.2: Recall for søkeord A

mindre i søkesvar til klient. Løsningen lever ikke hundre prosent recall når det er svært få tjenere opp i mot antallet noder i nettverket. Samme søkeordet for 20 tjenere i nettverket leverer hundre prosent recall for alle uttestede nettverksstørrelser.

6.3 Sett fra klienterspektiv

For klienten gjør løsningsforslaget mitt en helt ny tjeneste tilgjengelig. Med distribuerte indekseringstjenere er det mulig for nettverket å søke igjennom dokumenter å få store svarsett tilbake. I sin artikkel om Freenet [28] sier Ian Clarke et. al. at søkefunksjonalitet er en av fremtidige forskningsfokus. Artikler har ikke gitt indikasjon på at tekstsøk er en tjeneste Freenet og liknende systemer leverer. Bruk av tekstsøk på Internett har økt svært mye og er mye klarer fremme i media med tjenester som Google, Yahoo, MSN og Fast. Søk på Internett etter tekstdokumenter kan sies å ha blitt en vanlig aktivitet. En rapport av Nielsen//NetRatings påstår tekstsøk står for de 3 mest besøkte applikasjonene på Internett [15]. I henhold til dokumenter er Freenet laget for å gjøre alle mulige dokumenter tilgjengelig gjennom nettet og på denne måten støtte ytringsfrihet. For å finne tekster bruker man

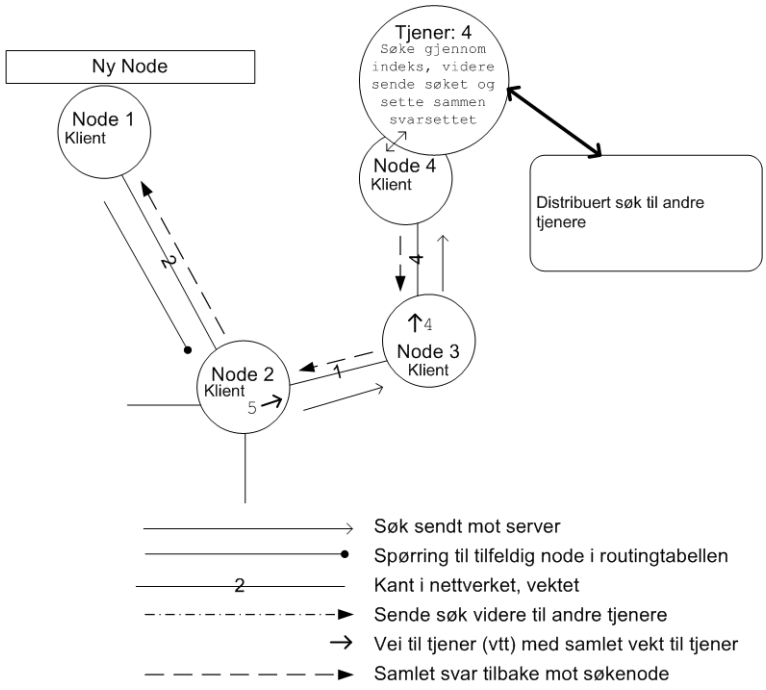
i hovedsak to forskjellige fremgangsmåter. Katalognavigering kan brukes for å finne informasjon om tema brukeren ikke vet så mye om. Katalogiseringen er tidskrevende og leverer ikke rangerte resultat. *Signed subspace* gir en form for innholdssøk i tekstdokumenter. Indekseringstjenerløsningen som er simulert og testet ut vil gi god til å søke på tema man allerede vet noe om. Dette fordi spissing av søk kan gjøres når terminologien for temaet er innlært. Løsningsforslaget vil gi svært god recall for søket, liten belastning på nettverket, mulighet for å søke i gjennom store delere av dokumentmassen og flere utvidelsesmuligheter.

Sett ut ifra figur fra artikkelen Protecting Free Expresssioin Online with Freenet [28] er det lett å se at gjennomsnittlig hopplengde fra informasjonskonsument til informasjonsholder er 10 for 100000 noder i nettverket. 10 hopp for nettverket når routingtabellene er store og konvergeringen av GUID nøkler er operativ. Worstcase ligger på 50 hopp til informasjonsnode og ved videre forbedringer i nettet vil verdien gå ned mot 6 hopp. Simuleringsnettverket leverer mindre en 6 hopp fra søkenode til tjenernoden i gjennomsnitt for alle antall av tjenere. For et nettverk med 1000 tjenere og 100000 noder er gjennomsnittlig hopp fra søkenode til tjener 3. Figur 6.1 viser hvordan en ny node i nettverket enkelt kan bruke søke og indekseringsfunksjonaliteten.

Worstcase for kontakt med tjener ligger opp mot normalen for vanlig dokumenthenting i Freenet.

Bootstrappingen (hvordan koble til når den kommer inn i nettet) til tjener gjøres svært enkelt og er en av styrkene ved løsningsforslaget mitt. Se figur 6.3. For å komme i kontakt med en tjener trenger noden bare å sende søk til en av nabonodenene sine. Mest sannsynligvis vil nabonoden ha en *vtt* eller sende samme melding videre. I figur 6.3 er node 1 ny node i nettverket. Node 1 har bare node 2 i routingtabellen og sender søket til denne noden. Node 2 har fått pingmelding fra tjener og har *vtt*.

På denne måten er indekseringstjenerene automatisk tilgjengelig for klientene helt fra pålogging av systemet. Etter en ukjent kort periode vil noden motta en ping fra tjener og selv vite om korteste vei til tjener.



Figur 6.3: Bootstrapping til tjener

Håndteringen av frafall i P2P nettverk er viktig. Et P2P nettverket er dynamisk og noder kobler seg fra og til hele tiden. Løsningsforslaget må takle at tjenere logger seg av og at noder på veien til tjener kobler seg fra. Tjenere kan koble seg fra uten at indekseringstjenesten forsvinner for klientene. Hvis søket blir sendt fra node til node nærmere tjener og tjeneren er borte fra nettverket kan den siste noden i nettverket sende søket videre til en semi-tilfeldig node i sin routingtabell. Siden det finnes flere distribuerte tjenere vil søket ende opp hos en annen tjener og levere søket tilbake til søkeklienten. En slik fremgangsmåte vil gjøre søkeprosessen midlertidig tregere og belaste nettverket mer en kort stund.

Neste gang en tjener sender ut ping vil *vtt* oppdateres i nodene og tjenesten leveres raskt igjen. En annen måte å løse problemet med frafallende tjenere er å lagre flere *vtt* og ikke bare den raskeste. Hver node kan ha en liste over hvilke noder i routingtabellen som er *vtt*. Om en tjener faller i fra i nettverket kan nettverket levere en melding til søkeklienten om at tjeneren er frafalt og at søket må sendes en annen vei for å nå tjener.

Indekseringstjenerløsningen er tenkt for alltid-påloggede noder i nettverket. Dette reduserer muligheten for at tjenere faller ut, men er ingen garanti siden tjenere kjører på vanlig datamaskiner som krasjer, starter på nytt og mister internettforbindelsen. Noder på veien til tjener kan også falle ifra og

dette er mye mer sannsynlig. Routing rundt "døde" noder kan enkelt gjøres ved at søket sendes til en annen node enn den døde eller at søket returneres til søkenoden med en melding om at veien til tjener denne veien temporært ikke kan brukes. Neste gang en pingmelding kommer til søkeklienten er *vtt* operativ igjen. Her kan også løsningen med å lagre flere rangerte *vtt* fungere. Om søkenoden sender ut et søk og får tilbake en feilmelding kan den nest beste *vtt* brukes og denne metoden gjøres flere ganger til søket returneres. Worst-case for både frafallende tjener og frafallende noder på *vtt* er bare en forsinkelse i levert søkeresultat og ikke et frafall av tjenesten for klienten.

6.4 Sett fra tjenerperspektiv

Tjenere er den nye entiteten i nettverket. Indekseringstjenere gjør at nettverket beveger seg litt bort fra atomisk P2P og mot brukersentrert P2P. Flere aspekter ved tjenere er interessante:

- Frafall av tjenere
- Oppdatert indeks
- Rangering
- Versjonering av filer
- Kontroll over indeksen
- Hvordan svarsettet leveres fra tjener til klient
- Anonymitet

6.4.1 Frafall av tjenere

Frafall av tjenere er mer problematisk sett fra et tjenerperspektiv. For å opprettholde persistentheten i systemet må indekser ikke forsvinne totalt fra systemet. Når tjenere mottar en indeks av en fil fra en publiseringsnode vil indeksen lagres hos tjeneren og distribueres til andre tjenere. Oppsetting av tjenere er tenkt å gjørs for noder med stor routingtabell (godt koblet til nettverket). Tjenere er også tenkt å være satt opp i alltid-online noder. Tjenere kan likevel forsvinne, brukere kan finne ut at de ikke vil være tilkoblet nettverket mer, maskinen kan slutte å fungere og det finnes en myriade grunner til at tjenere faller i fra. Derfor må det finnes en måte å distribuere indeksene på slik at indekser for dokumenter ikke forsvinner fra nettet på grunn av

frafall av tjenere. For løsningsforslaget mitt er det naturlig å la tjenere kommunisere med andre tjenere og utveksle deler av indeksen. Naturlig siden ingen tjenere er antatt å ha svært stor kapasitet og mulighet til å lagre indekser for alle dokumenter. Å holde informasjon om dokumenter persistent må vektes mot muligheten for hvor store indekser det er mulig å lagre hos tjenere.

En mulig løsning er å la tjenere periodisk spre deler av indeksen sin til andre tjenere. Som et andrevalg kan tjeneren selv sende ut søk til andre tjenere for å finne ut hvor godt indeksene er distribuerte og så velge å fjerne visse elementer fra egen indeks og distribuere andre mindre distribuerte elementer. En slik løsning vil skape mer belastning på hele nettverket og for tjenernoden (nettverket må overføre spørringene og tjeneren må kjøre oppdateringsalgoritmen). En eller annen funksjon for verifisering av hvor distribuert indekseringselementer er i tjenere må implementeres for å garantere at alle publiserte objekter er søkbare.

6.4.2 Oppdatert indeks

Ett annet problem med distribuerte tjenere er fjerning av utdaterte dokumenter fra indeksen. Selv om Freenet er laget for å være et persistent system er det mulig for dokumenter å bli faset ut av nettverket. En mulig løsning for verifisering av at dokumentene er tilgjengelige i nettverkte er å sette en tid å leve for alle indekselementene. Hver gang en node henter ett dokument fra nettverket kan informasjonen om hentingen sendes til en tjener. En annen fremgangsmåte er å registrere hvert innkommende søk og sette tid å leve verdien til startverdien hver gang et dokument søkes. Denne metoden vil ikke ta hensyn til at selv om dokumentene blir søkt etter er det ikke sikkert de blir lastet ned i nettverket. Elementer i indeksene kan unngå å bli tatt ut av indeksen selv om de blir faset ut i nettverket.

Problemet med å implementere en løsning for fjerning av indekselementer i Freenet er at det ikke finnes noen statistisk informasjon om hvor fort veldig upopulære dokumenter forsvinner fra nettverket. Lagringsplass blir stadig billigere og Freenet er hovedsakelig laget for tekstdokumenter. Tekst er svært små dokumenter i sammenlikning til film, musikk og bilde dokumenter. For tekstdokumenter er det min påstand at systemet kan lagre total permanent. På 10 gigabyte harddiskplass er det mulig å lagre 4-5 millioner tekstdokumenter på forskjellige formater og 10 gigabyte er ikke lenger mye harddiskplass. Nettverket må selvfølgelig ta hensyn til at det finnes mange datamaskiner med lav kapasitet på nettverket. Ressursnoder i hensyn til lagringsplass kan ta veldig mye av belastningen til nettverket uten at påvirker brukeren av ressursnoden mye.

6.4.3 Rangering

For å rangere dokumenter mot hverandre trenges det informasjon om nettverket. Freenet gir ingen noder mulighet til å ha noen form for global informasjon. Etter at det distribuerte søket er ferdig er det mulig for tjeneren å rangere dokumentene. Siden tjeneren har hele svarsettet og hele svarsettet kan sees som global informasjon. Dokumentene kan så rangeres etter flere forskjellige algoritmer. Forenklede algoritmer fra internettsøk kan benyttes. Forenklet fordi algoritmer for internettsøk gjerne er beregnet for høyressursmaskiner med mye regnekraft. For eksempel kan en algoritme hvor nærheten av ordene eller (om filen er strukturert) høyere rangering for dokumenter med søkeordene i tittel eller sammendrag brukes. Rangeringen kan også distribueres. Tjeneren tar allerede en stor del av belastningen i nettverket, men om en strukturert fil med inverterte indekser sendes til nabonoden kan denne ta seg av selve rangeringsprosessen før den sender søkeresultatet videre til søkenoden.

6.4.4 Versjonering av filer

Ved å bruke distribuerte tjenere til å indeksere og holde orden på dokumentmassen i nettverket er det mulig å få inn en eller flere former for versjonering av dokumenter. I distribuerte systemer hvor alle klienten også leverer en lokalisert indekseringstjeneste (enten bare for sine egne dokumenter, eller for et lite subsett av lokale noder) er versjonering av filer vanskelig. Freenet bruker GUID nøklene for å gi en viss form for versjonering av filene. GUID nøkkelen kan også brukes til versjonering av filer hos indekseringstjeneren. Hvis informasjonpubliserer setter *signed-subspace keyen* til å inneholde versjoneringsnummer kan indekseringstjeneren bare levere den siste og mest oppdaterte versjonen av filen ved søk. En mindre sikker måte versjonere på, men full mulig, er å la informasjonskonsumenten selv versjonere etter mottatt søkeresultat. Om samme dokumentet legges ved siden av andre versjoner av dokumentet rangert etter publiseringsdato kan brukeren selv velge den siste versjonen.

For noen dokumenter som for eksempel utgivelse av programvare eller bøker kan det være ønskelig å hente en gammel versjon av dokumentet. Versjoneringen må legges opp på en slik måte at den ikke sensurerer ut gamle dokumenter. Om indekseringstjenesten utelukkende leverer GUID for det nyeste dokumentet vil dette over tid føre til at gamle dokumenter faller ut av nettverket. En mulig løsning er at informasjonspublisereren kan *tagge*(markere) GUID med informasjon om gamle dokumenter ikke skal vises i søk. Dette vil gi informasjonspubliserer mulighet til over tid å fjerne gamle

versjoner av dokumentet i nettverket. Hvis indekseringen skal støtte fjerning av dokumenter i indeksen må det finnes en måte å absolutt identifisere informasjonspuliserer for å unngå misbruk av løsningen. Et eksempel på misbruk kan være at en node fjerner eller oppdater andres dokumenter. En løsning er at publiserer bruker digital signatur.

Indekseringstjenere kan gjøre funksjonaliteten av versjonering mye sterkere i Freenet. Versjonering av dokumenter vil i tilfelle gjøre at Freenet bryter en av begrensningene ved P2P systemer. Alle begrensningene gjør at bruken av P2P også begrenses. Å løse de innlysende problemene med P2P vil gjøre verktøy basert i P2P tankegang bedre.

6.4.5 Kontroll over indeksen

En sannsynlig situasjon for en indekseringstjener er å gjøre svært uønsket informasjon søkbar. Datamaskinen kan beslaglegges for rettslig forfølgelse og det kan være etiske problemstillinger knyttet til visse klasser av dokumenter. Dette innfører to momenter for sikring av applikasjonen om løsningsforslaget skal implementeres.

Hva tjeneren indekserer kan til en viss grad kontrolleres. Applikasjonen kan implementeres på en slik måte at det er mulig for personen som setter opp tjeneren å sensurere innholdet den indekserer. Dette kan gjøres på flere måter. En mulig måte å gjøre det på er å lage en sentral database koblet til Freenet hvor tjenerapplikasjonen kan laste ned lister over ord tilknyttet diskuterte tema. Tor har en slik type sikre tjenere med høy tillitsgrad [33].

Når tjeneren settes opp kan brukeren selv velge i en liste problematiske tema brukeren ikke vil skal indekseres av tjeneren og applikasjonen laster selv ned ordlister knyttet til temaene. Eksempel på slike tema kan være barneporno, terrorrelaterte dokumenter, hacking, personinformasjon og liknende.

For å unngå mulig rettsforfølgelse kan indeksen krypteres for menneskelesning og utenforstående angrep. Om maskinen blir beslaglagt eller blir tatt over bør indeksen ikke være enkelt tilgjengelig for å beskytte eier av datamaskinen.

Muligheten med å velge bort uønskede tema fører til at brukeren ikke trenger å se hva som indekseres på datamaskinen. Ett hvert beslag av datamaskin kan vil gi brukeren *plausibel fornektelse* (*plausible denial*) angående hva maskinen har indeksert siden denne informasjonen ikke kan dekrypteres og gjøres lesbar. Brukeren kan også vise til at uønskede dokumenter er prøvd fjernet fra tjeneren.

6.4.6 Hvordan svarsettet leveres fra tjener til klient

Om simuleringsprototypen hadde innhold mer funksjonalitet hadde det vært mulig å teste forskjellige måter å levere svarsett. Svarsette kunne blitt rangert og filtrert.

Rangeringsfunksjonen kan enten legges hos tjener eller i alle klientene. Om Tjeneren har samlet ett stor svarsett kunne hele settet leveres til en nabonode. Nabonoden prosesserer svarsettet og gir en rangert versjon videre til søkeklienten. For å kunne rangere må store deler av indeksen være tilgjengelig. Metoder for rangering kan være å velge de dokumentene med flest søkeord i dokumentet. I tillegg kan dokumenter hvor søkeordene er i nærheten av hverandre rangeres høyere enn andre.

Filtrering er et kraftig verktøy for å gjøre svarsettene både mindre og bedre. Ved bruk av indekseringstjenere kan svarsettet automatisk filtreres i at like dokumenter (lik GUID) ikke blir representert mer enn en gang. Systemer som Kazaa og Napster leverer mange filer med samme filnavn og samme binærsignatur.

6.4.7 Anonymitet

Foreslått for Tor nettverket [33] kan plasseringen av en tjener eller klient forandres i nettverket ved å bryte tilkoblingen til de nodene som finnes i routingtabellen og lage en ny routingtabell. Denne fremgangsmåte vil gi større mulighet for å hindre sniklyttere å identifisere eventuelle tjenere. Prosessen med å bytte plass i nettverket er svært krevende. Fordelene ved metoden må settes opp i mot ulempene, men det er innlysende at bytting av plass i nettverket vil gjøre tjenere mindre mulige å identifisere og lokalisere. Hvor stor belastning bytting av plass i nettverket er for tjeneren og klientene rundt tjeneren er det vanskelig å dedusere seg frem til. Dette må i tilfelle testes ut for å kunne stadfeste ressursbruken og hvor fornuftig en slik metode er.

6.5 Legimitet

Det kan fastslås at det ikke er mulig å sikre legimitet i anonymiserte systemer lik Freenet. For nettverket er det ikke hensikten å på noen måte å sensurere de delte dokumentene. Dette gjelder også for dokumenter med opphavsrettigheter. Når man snakker om legimitet er det hensiktsmessig å ta frem de underliggende tankene for legimitet. Legimitet baserer seg i at det finnes en høyere autoritet som indikerer hva som kan deles og ikke deles av dokumenter i et nettverket. Freenet er laget for å unngå at noen høyere autoritet bestemme hvilke typer dokumenter nettverket deler. Ytringsfrihet i sin ytterste form kan ikke implementeres i henhold til eksisterende lover og regler i noen land i verden. Norge for eksempel har strenge medietabu på nazisme og trakassering av privatpersoner. Lovene har gode grunner for å eksistere, men strider litt mot tanken om ultimal ytringsfrihet. Land som Kina har andre lover for ytringsfrihet og Freenet kan her være et verktøy for å kunne diskutere upopulære og ulovlige tema.

6.6 Etterpåklokskap

For å kunne simulere nettverket i det hele tatt måtte ett sett med parametere velges ut. Siden design, programmering av simuleringen og testing ble det ikke tid til å teste ut større funksjonalitet. Alltid er det fornuftig å evaluere et prosjekt i ettertid. Kunnskap om hvordan ting skulle ha vært gjort kan ofte være like verdifull som de faktiske resultatene. For denne simuleringen er det noen få problematiske aspekter å peke på. Valg av parametere, rekkevidde av simuleringen og dynamikk i nettverket.

6.6.1 Valg av parametere

Parametrene for simuleringen var svært viktige for hvilke resultater som kunne trekkes ut. Nedenfor følger en *i ettertid* drøfting av valgene gjort før simuleringen ble utført og under veis.

Nettverksstørrelse: Nettverkstørrelsene har fungert svært godt og gitt gode indikasjoner om fortsettende skalering av løsningsforslaget.

Routingtabell: Routingtabellstørrelsen er satt til 20 noder. Etter å ha satt meg mer inn i funksjonaliteten til Freenet-liknende systemer har det vist seg at 20 noder er svært lite. Andre har også foreslått forbedringer for Freenet og har brukt 240 noder i routingtabellen for simuleringene. Siden min simulering bruker 20 noder i routingtabellen kan resultatene

kategoriseres som *worst-case* resultater. Et nettverk med mer en 20 noder i routingtabellen vil i følge *small-world* prinsippet levere diskuterbart kortere veier gjennom nettverket. Uansett vil økning av antall noder i routingtabellen påvirke *worst-case* vei til tjener resultatene i positiv retning. Større routingtabeller ville også kanskje gi indikasjon om pingalgoritmen er for god. For god i den forstand at ekstremt lave *vtt* verdier kan påvirke anonymiseringen av tjenere i negativ retning.

Distribuering av søk: Søk er i simuleringen satt til å sende søket videre til halvparten av de andre tjenerne. For å dra ut mer informasjon om søketjenesten hadde det i ettertid vært fornuftig å kunne variere videresendingen av søk for å måle hvordan dette påvirket søkeresultatene.

Distribuering av indekser: På samme måte som for distribuering av søk ville det vært interessant å se på flere parametere for videresending og distribuering av indekser. På samme måte ville det vært interessant å se hvordan strukturering av videresendingen ville påvirke indekseringstjenesten og søkefunksjonaliteten. For eksempel å strukturere indekseringstjenesten etter GUID nøkkelverdier og *relativ posisjon* i nettverket.

Pingmelding: Pingmeldinger i sykler kunne også vært unngått og enkelt forbedret. Pingmeldinger mottatt fra samme node med høyere latency kunne termineres og ikke sendes videre.

6.6.2 Dynamisk nettverk

En av de store feilene med simuleringen er at det er statisk når det er satt opp. Noder "dør" ikke, flytter seg ikke i nettverket og forandrer ikke routingtabellen. Oppsetting av et dynamisk nettverk ville gi mye bedre indikasjon om hvor distribuert indeksene må være for å unngå total fjerning av indekselementer. Dynamiske routingtabeller hadde vært interessant siden dette på en mye bedre måte reflekterer det virkelige systemet. I tillegg er dynamikken et viktig aspekt for vurdering av pingalgoritmens effekt i nettverket.

6.6.3 Rekkevidde

Å lage ett forslag for muliggjøring av søk og indeksering i anonymiserte distribuerte nettverket var allerede ambisiøs. Svært mange aspekter har i løpet av design, programmering, test og analyse vist seg å være interessante, men har havnet i *etterpåklokskap*-båsen. Flere *aha!*-opplevelser i løpet av programmeringsperioden førte til større funksjonalitet. I løpet av testingen har hele klasser blitt programmert om og tester forbedret for å gi bredere og mer

nøyaktige resultater. Likevel gjenstår det som nevnt ovenfor flere punkter hvor utviding og tillegging av funksjoner hadde ført til bedre resultater.

Rekkevidden til prosjektet har blitt å bare vise generelt om det er mulig å legge ett virtuelt lag med tjenere over nettverket. Tankene og ideene i programmert i simuleringen gir muligheten for å implementere mer avansert IR-funksjonalitet for distribuerte nettverk.

Det viktigste å peke på som burde vært gjort annerledes er å gjøre simuleringen dynamisk. Den kanskje sterkeste siden ved ett P2P-system er hvor godt systemet takler frafall og omstrukturering. Derfor er det frustrerende at det ikke var tid til å programmere og teste ut hvordan dynamiske variabler påvirket nettverket og søkefunksjonaliteten.

Kapittel 7

Refleksjoner

> *He who fights with monsters should look to it that he himself does not become a monster. And when you gaze long into an abyss the abyss also gazes into you.*
>
> Friedrich Nietzsche

Nietzsche har et poeng i at intens søken etter kunnskap i et område fører til at synes på verden forandres. Brillene"vi ser verden med er formet av vår kunnskap og konklusjoner dra ut på et subjektivt grunnlag. Avhandlingen har til nå fokusert på det tekniske aspektet ved anonymiserte distribuerte systemer. Etiske problemstillinger knyttet til bruk av systemene, hvor reell anonymiteten er og hvorfor det forskes på P2P systemer er også interessante spørsmål. Forhåpentligvis kan denne vinklinen gi andre briller"å se den samme teknologien med.

Anonymiserte distribuerte systemer er et kraftig verktøy for ytringsfrihet, men som alle andre nye teknologier er det mulig å misbruke teknologien. Er så denne anonymiteten virkelig?

7.1 Etiske problemstillinger

Privatsfæren er et viktig konsept i hverdagen. Internett har gått fra å være semi-anonymt til å bil ett kontrollert rom. Selv om brukere har lyst å bruke Internett bør ikke de samme reglene for titting og overvåkning gjelde?

7.1.1 Privat Sfæren

American Civil Liberties Union (ACLU) går 24. mai 2006 ut for å vekke den amerikanske befolkningen. Dokumenter lekket fra telefonselskaper viser at

informasjon om vanlige personer gis fra private firmaer til statlige overvåkningsinstitusjoner.

> *We cannot sit by while the government and the phone companies collude in this massive, illegal and fundamentally un-American invasion of our privacy*
>
> ACLU Executive Director Anthony D. Romero [1]

Samme dag går det norske Datatilsynet ut og advarer mot film og musikkbransjen. For å stagge fildelingstjenester samler musikk- og filmbransjen inn informasjon om nordmenns aktiviteter på P2P-nettverk. Juridisk rådgiver Guro Slettmark i Datatilsynet sier "*at dette handler om folks personvern og hun mener at denne typen overvåkning kan true folks rettsikkerhet*" [6].

Det finnes svært strenge lover for hvordan innsamling av informasjon om privatpersoner kan gjøres. Ved bruk av Internett samles det automatisk inn en mengde informasjon om brukere. Denne informasjonen kan i Norge ikke datamines, men om webtjeneren ligger for eksempel i USA gjelder amerikanske lover. Amazon.com bruker alle informasjonen om brukers oppførsel på deres sider for å sende informert reklame. Nettsidene blir lagt opp for å passe inn til tidligere kjøp. I tillegg sendes det ut rutinemessig e-post med vinklet reklame. Selvfølgelig er det bare å slette *cookie* filen for å unngå problemet. EU har innført strengere krav til lagring av informasjon om brukere [7]. Internet Service Providors (ISP) er påkrevd å lagre informasjon om "Internetttrafikkdata", SMS og informasjon om telefonsamtaler. Informasjonene skal lagres minimum 6 måneder og maksimum 2 år. Disse lovene er vedtatt for å muliggjøre etterforskning av organisert kriminalitet, terrorvirksomhet og nasjonal sikkerhet. Et spørsmål man kan stille seg i denne sammenheng er om brukere har rett til å overføre sin private sfære til Internet? I den virkelige "*ikke cyber verden*" er det nesten helt ulovlig å overvåke personer før de har utført en gal handling.

Preemtive survalance er et passende ord for den lovgivningen EU og USA har laget for Internet. Går så denne overvåkningen ut over brukere? Dette spørsmålet er det svært vanskelig å svare på, men man kan vri spørsmålet og spørre; *hvor* mistet privatpersoner retten til informasjon om seg selv? Lovverket for overvåkning av privatpersoner er ikke strengt uten grunn. I det demokratiske Europa er den underliggende filosofien at informasjon om seg selv er noe enhver privatperson styrer over selv. Freenet kan fungere som en utvidelse av den private sfære til Internet. En menneskerett i det virkelige liv kan overføres til Internet. Selvfølgelig muliggjør krypteringen og anonymiteten i slike systemer at personer med onde hensikter tar i bruk

systemet. Ian Clarke forsvarer Freenet med å si at det allerede finnes bedre systemer for personer med onde hensikter [28].

7.1.2 Mulig ondsinnet bruk

Freenet kan brukes til mye siden nettverket er sikkert anonymisert. Det finnes allerede eksisterende verktøy som er bedre for ondsinnet bruk. Darknets er et kraftig og svært vanskelig overvåkbart verktøy for å overføre informasjon uten å bli oppdaget. Darknets kan bety nettverk uten kontroll eller mer spesifikt nettverk bygget opp på samme måte som Freenet, men hvor noder legges til gjennom informasjon fra person til person kommunikasjon i den virkelige verden. Nodene kan legges til fra andre digitale kanaler eller fra diskett eller minnebrikke.

Det som kanskje er litt overraskende er at denne typer systemer har vært i bruk for militæret i lang tid (helt fra Enigma under andre verdenskrig). At statlige organer ser Freenet som en trussel men enda ikke har fått øynene opp for hvor enkelt det er å implementere ett militær-grad krypografisk Darknet er svært overraskende. Det virker ikke som det finnes strategiske tiltak for å bekjempe en slike systemer i Norge. For privatpersoner kan Freenet vært et godt verktøy om de vil ta vare på sensitiv og triviell informasjon på Internet.

TOR nettverket har blitt brukt til *denial of sercvice* angrep [8]. Noder i TOR nettverket identifiserer seg for tjenestene på Internett og kan enkelt blokkeres. Og informasjon om angripere er svært vanskelig å lokaliserer siden trafikken går gjennom flere andre noder. Freenet har ingen funksjon for å aksessere vanlig Internett gjennom andre applikasjoner.

Motstandere mot anonymisering argumenterer for at verktøyene gir mulighet til kriminelle handlinger. Denne argumentasjonen holder ikke mål da verktøyet ikke utfører de kriminelle handlingene. Brukeren er selv ansvarlig for egen aktivitet, men Freenet er ett kraftig verktøy for anonymisering av kriminelle handlinger.

Den distribuerte tjenerløsningen legger ikke til noen ekstra funksjonalitet for ulovlig aktivitet. Fremtidsvisjonen der tjeneren kan laste ned og fjerne dokumenter (ikke indeksere) med uønsket innhold, gjør at sosiale konvensjoner kan trenge inn i nettverket. For eksempel problemet med barneporno. Dokumenter med innhold om slike saker kan relativt enkelt fjernes ved bruk av flaggede ord. Og store deler av nettverket vil dele den sosiale konvensjonen om at slike dokumenter er galt som vil føre til liten mulighet til å søke i slike filer. Det negative med indekseringstjenerløsningen vil være at det er lettere å søke i gjennom rettighetsbeskyttet materiale og hente disse fra nettverket. Dokumentene vil uansett være til stede i nettverket og tjeneren gjør bare nedlastningen mer aktuell. Forsinkelsen i nettverket vil gjøre at andre verktøy

vil være mer passende for nedlastning av kopibeskyttet materiale.

7.2 Anonymitet en realitet?

Er det virkelig mulig å anonymisere et nettverk? TOR nettverket var teoretisk sikkert og brukt av institusjoner som FBI, CIA med fler, men ble knekt ikke bare på *en* måte [27]. SSH, SSL, Telnet, Javaapplikasjoner, Flash, JavaScript, VBScrip var de forskjellige funksjonene overvåkeren kunne bruke for å identifisere brukeren. I tillegg kunne informasjon om browser, brukernavn og lokalnettverk vises. Kan det samme gjelde for Freenet?

Freenet er et åpen kildekode system og alle har muligheten til å se signaturen på overføringsprotokollene. En overvåker kan sende meldinger for å tvinge frem informasjon fra en datamaskin som bruker Freenet overføringsprotokoll. På denne måten kan informasjon om mange noder vises frem.

Kina som er en av hovedobjektene for ytringsfrihetskonseptet i Freenet vil trolig forby systemet om det kommer i allment bruk. Kina sensurer aktivt informasjon på Internett og Freenet bryter disse reglene. Siden protokollen er åpen for alle kan Kina enkelt identifisere protokollsignaturen, overvåke protokollen, identifisere brukere og stoppe trafikk med Freenet signatur. Da hjelper ikke anonymiteten mye.

Anonyme systemer virker antatt å være anonyme før det publiseres en artikkel som beviser det motsatte. Er dette en sunn antagelse og i tilfelle hvordan skal anonymiteten verifiseres? Antagelsen bør jo alltid være at systemet er *broken* i den forstand at det *er* mulig å overvåke og bruken bør justeres deretter.

7.3 Forskning? Napster var bra nok?

P2P nettverk, utenom IM tjenster, har ofte blitt populær på grunn av muligheten til å enkelt dele musikk, film og bilder. Ett lett søk på "P2P" gir 314,428 treff på ACM [2] 26. mai 2006. Forskningsdisiplinene faller over hverandre for å forske på P2P. Er virkelig P2P den fantastiske måten å gjøre alt på? En påstand er at den populære fildelingen hadde vært lovlig ville Napster vært godt nok. Den siste og mest effektive fildelingsløsningen er BitTorrent [3]. Ideen bak BitTorrent er at dokumenter brytes opp i små deler og distribueres i klientene. Når et dokument hentes fra nettverket lastes den samtidig ned fra alle holdere av filen. En veldig effektiv algoritme og protokoll, men denne brukes i all hovedsak til det samme som Napster. Alt går bare litt raskere.

Ett interesant spørsmål å stille er hvor viktig er akademisk utvikling av P2P systemer? De fleste systemene i utstrakt bruk har ikke kommet fra akademiske kilder. DC++ [9], Kazaa [12], BitTorrent [3], Napster [14], Freenet [10], MSN, Gnutella [11] (og systemer på samme protokoll) og TOR [33] er de mest brukte P2P-systemene. Bare en av disse er ett resultat av forskning og det er TOR. Rammene og ideene for Freenet er funnet på innefor akademiske fora, men utviklet gjennom SourceForge.net i åpen kildekode.

Hva bør akademisk utvikling av P2P systemer fokusere på for å opprettholde at forskningen fører til ny informasjon og kunnskap? Etter min mening er altfor mye av forskningen og publiseringen rundt P2P systemer rettet mot direkte forbedringer av implementerte systemer. All forskning burde heller basere seg i å utvikle konsepter for fremtidens P2P systemer og ikke på direkte forbedring av foreldede systemer. Flooding og Random Walk algoritmene er utdaterte søkealgoritmer og svært inneffektive i de fleste tilfeller. Viktig forskning burde heller fokusere på nye konsepter for nettverkstopologi, IR og forbedring av konsepter. Indekseringstjenere slik presentert i denne oppgaven gir en ide om hvordan en ny type tjeneste kan implementeres for fremtiden. Freenet er bare ett utgangspunkt siden dette systemet virker å være i forkant av annen utvikling innenfor fagfeltet. P2P har muligheten til bli ett svært kraftig verktøy for informasjonsdeling. De nye ideene fra progressive algoritmer bør vurderes for videre kunnskap innen fagfeltet.

Kapittel 8

Oppsummering og konklusjon

No arguing with that. *No* understanding *it either.*

Bill Bryson

I denne avhandlingen har jeg presentert distribuerte anonymiserte systemer og kommet med et forbedringsforslag til den underliggende arkitekturen. Tankene bak Freenet om muligheten for total ytringsfrihet på Internett motiverte meg til å komme med et løsningsforslag for søkefunksjonaliteten. Indekseringstjenere er interessant for systemer lik Freenet fordi det kan gjøre nettverket godt nok for allment brukt. Anonym bruk av Internett har blitt en aktuell problemstilling på grunn av økende overvåkning av privatpersoners aktivitet på Internett.

8.1 Oppsummering

Denne avhandlingen har gjort rede for distribuerte systemer og P2P. Anonymiseringsprosesser og teknikker er presentert for å forklare begrensingene en uttestingsprototyp må implementere. En løsning for søk og indeksering av innhold i tekstdokumenter for distribuerte systemer er presentert. Designet og kravspesifikasjonene er fulgt i implementering av prototyp for simulering.

Løsningen er simulert og testet ved hjelp av en prototyp. Grunnen til valg av simulering i motsetning til implementering av et virkelig system var muligheten for å fjerne noe av kompleksiteten og gjøre uttestingen lettere. Prototypen har sekvensielt bygget nettverk av størrelsene gitt i kravspesifikasjonen, med 8 forskjellige tjener antall, 20 noder i routingtabellen og testet med søkeord fra en søkefil.

Både positive og negative aspekter ved løsningen er trukket frem i analysen. Om det viktigste positive skal oppsummeres i en setning er det at løsningen kan implementeres. Det mest negative å si om prototypen er manglen på dynamikk i simuleringen.

Resultatene viser at pingalgoritimen kobler sammen nodene i nettverket for rask overføring fra klient til tjener. Nettverket blir bundet sammen slik at korteste vei, i følge small-world prinsippet, er mulig å identifisere og benytte seg av for nodene.

Distribuering av indekser og distribuerte søk leverer optimale verdier fra alle noder i nettverket på alle søkeordene simuleringsprototypen er utprøvd med. Det er observert at dårligere enn optimale verdier kan forkomme. Grunnen er svært få tjenere, få dokumenter med søkeordet og veldig stort nettverk. Uttesting med flere variabler i distribusjonen av både indeks og søk ville vært interessant, for å definere verdier hvor søkekvaliteten degenereres.

Søkekvaliteten representert av recall og ranking er presentert. Recall (innholder søkeordet) i nettverket er optimal for uttestede verdier og muligheter for å implementere ranking analysert. Presisjon er ikke diskutert siden ren IR-funksjonalitetet ikke er tilstrekkelig implementert.

Viden bruk av systemet fører til etiske problemstillinger. Anonymiserte distribuerte systemer støtter ytringsfrihet og muligheten for individer å ta med sin private sfære ut på Internett. Anonymiseringen fører også til mindre mulighet for lovlig overvåkning for å beskytte samfunnet mot terror og organisert kriminalitet. Ondsinnet bruk er mulig, men det finnes bedre løsninger for de fleste problematiske brukene av slike systemer.

8.2 Sammenlikning

Ikke så mange prosjekter har utført liknende tester med liknende løsninger. Kazaa [12] sentraliserer noe av belastningen i nettverket ved bruk av supernoder. Alle søk sendes mot noder med høy båndbredde, godt tilkoblet nettverket og med en indeks bygget opp av nabonodenes dokumenter. Supernoder kan sammenliknes med indekseringstjenere, men bare i det at noe av funksjonaliteten blir gitt til færre noder. Kazaa viser en markant forbedring av Gnutella protokollen. Supernodene i Kazaa blir automatisk valgt ut. Indekseringstjenere foreslått i denne avhandlingen blir frivillig satt opp av noder som selv velger å gjøre ressursene tilgjengelige. Indeksene hos supernoder gir lokaliseringsinformasjon om dokumentholder. Dokumentene i svarsettet blir ikke filtrert og indeksene er ikke med hensikt distribuert for å holde informasjonen i nettverket til en hver tid.

Renda og Callan [47] beskriver uttesting av et system med visse likhet-

strekk til indekseringstjenerløsningen. Implementeringen deres organiserer nettverket hierarkisk. Noen noder med mye ressurser (kaldt *Ultra Peers*) sentraliserer søkefunksjonaliteten i deler av nettverket. Deler av nettverket er satt til å lage spesielle deler av dokumentsettet og *Ultra Peers* gir søkefunksjonalitet for en spesiell del av nettverket. Sentraliseringen førte til bedre resultater i form av F-Score

$$\frac{2 * Presisjon * Recall}{Presisjon + Recall}$$

.

Ultra Peers med lik funksjonalitet gir gode resultater selv om mange av *ultra peers* forsvinner fra nettverket. Dynamikk er ikke godt testet for indekseringstjenerløsningen. Likheten med *Ultra Peers* gi støtte for antagelsen om at indekseringstjenere vil kunne levere god søke og indekseringskvalitet selv om flere forsvinner fra nettverket. Et annet punkt er flytting av relativ posisjon i nettverket. Indekseringstjener kan koble av kontakt med den eksisterende routingtabellen og flytter plass. Likevel vil nettverket fungere godt og tjeneren kan få ny relativ posisjon i nettverket. Grunnen til at dette kan være fornuftig er muligheten for å overvåke aktiviteten hos indekseringtjener om overvåker kontrollerer en av nabonodene.

8.3 Konklusjon

Denne avhandlingen har vist at det mulig å implementere indeksering og søk for anonymiserte distribuerte systemer. Slike systemer er muligens fremtidens informasjonssystemer om funksjonaliteten og skalerbarheten muliggjør allment bruk. I løpet av avhandlingen er design, simulering, test og evaluering presentert. De viktigste momentene er:

- Design av et lag med indekseringstjenere.
- Simuleringen følger alle begrensinger for anonymiserte distribuerte systemer.
- Uttesting viste at indekseringstjenere kan muliggjøre
- effektiv nettverksbruk og mulighet for implementering søke og indekseringstjenester.
- I tillegg beholdes anonymitet for publiserer, konsument, holder og overfører i nettverket.

- Avhandlingen har pekt på aktualiteten til anonymiserte systemer for beskyttelse av privatinformasjon.

Løsningen viser en ny måte å implementere distribuert søkefunksjonalitet for anonymiserte nettverk. Helt enkel indeksering og søk er fungerende i prototyp og har gode egenskaper både innen nettverkseffektivitet og recall.

Videre testing med flere variabler må til får å stadfeste kvaliteten til løsningen. Kvaliteten måles i recall, presisjon, ranking og hvordan dynamikk i nettverket påvirker overføring i nettverket.

8.4 Videre arbeid

Distribuerte indekseringstjenere har vist seg å kunne håndtere søk i anonymiserte distribuerte nettverk likt Freenet. Videre arbeid vil være forbedring, implementering av brukbare moduler for Freenet (for full uttesting). Fremtidsvisjoner og videre utvikling av prosjektet kan oppsummeres slik:

Forbeding: Eksperimentet kan forbedres på flere måter:

- ping algoritme som tar hensyn til sykler i nettverket
- oppdatering av *vtt* og routing rundt døde noder
- flere *vtt* for hver node
- bedre håndtering av døde indekseringstjenere og oppdaterte indekser distribuert i nettverket

Lage modul: For å teste ut indekseringstjenere kan en ferdig modul for uttesting i nettverket programmeres. Dette for å se den virkelige oppførselen til indekseringstjenere i Freenet.

IR: Lage god IR-funksjonalitet hos tjener og distribuert til andre noder. Distribuert for eksempel ved rangerign av dokumentsettet i en annen node enn tjeneren. IR kan implementeres som moduler for tillegging i tjeneren. På denne måten kan flere typer algoritmer testes ut. Brukeren kontrollerer tjeneren og kan selv velge hvilken type IR-moduler tjeneren skal ha.

Videre utvikling av distribuerte tjenere er interessant i at det gir muligheten for avansert IR-funksjonalitet og forbedrer brukbarheten av systemeter hvor løsningen er implementert.

Koding

> *All parts should go together without forcing. You must remember that the parts you are reassembling were disassembled by you. Therefore, if you can't get them together again, there must be a reason. By all means, do not use a hammer.*
>
> IBM maintenance manual, 1925

Prototypen er programmert i Java (J2SE). API for koden kan finnes på `http://folk.ntnu.no/tvenning/API/`. Koden, API og testfiler er lagt ved på DVD-ROM og som vedlegg på innleveringstjenesten `daim.idi.ntnu.no` Inneholder både kompilerte .java filer og de kjørbare.class filene. Mainmetoden ligger i StartTestklassen.java.

Bibliografi

[1] ACLU launches nationwide action against NSA snooping on americans phone calls. http://www.aclu.org/safefree/nsaspying/25647prs20060524.html. 24. mai 2006.

[2] Acm portal. http://portal.acm.org.

[3] Bittorrent.org, a forum for developers to exchange ideas about the direction of the bittorrent protocol. http://www.bittorrent.org/.

[4] Citeseer - most sited. http://citeseer.nj.nec.com/articles2000.html.

[5] Citeseer.ist scientific literature digital library. http://www.citeseer.com.

[6] Dagens næringsliv: Datatilsynet advarer mot overvåkning, guro slettemark. http://www.dagensit.no/article790488.ece. 24. mai 2006.

[7] Datatilsynet nyhetsarkiv 21.12.2005 datalagring - eu direktiv. http://www.datatilsynet.no/templates/Page____1290.aspx.

[8] Digi: Slutt på anonym surfing på skjult nett. http://www.digi.no/php/art.php?id=303455.

[9] Direct connect plus plus. http://dcplusplus.sourceforge.net/.

[10] Freenet website. http://freenet.sourceforge.net.

[11] Gnutella website. http://gnutella.wego.com.

[12] Kazaa. http://www.kazaa.com/.

[13] LOV 2000-04-14 nr 31: Lov om behandling av personopplysninger (personopplysningsloven). http://www.lovdata.no/all/hl-20000414-031.html.

[14] Napster website. `http://www.napster.com`.

[15] Nielsen//netratings. `http://www.nielsen-netratings.com`.

[16] Norsk oversettelse av OECDs retningslinjer, kryptopolitikk - retningslinjer og problemstillinger. `http://odin.dep.no/nhd/norsk/dok/regelverk/prinsipputtaleser/024005-990135/dok-bn.html`.

[17] Seti@home. `http://setiathome.ssl.berkeley.edu/`. 25. mai 2006.

[18] Tor: An anonymous internet communication system. `http://tor.eff.org/`.

[19] Philip E. Agre and Marc Rotenberg. *Technology and privacy : the new landscape*. The MIT Press, Cambridge, Massachusetts, 1997.

[20] Alexander Alvaro, Gus Hosein, and Ivan Szekely Meryem Marzouki. Terrorizing privacy? Session Audio, Conference on Computers, Freedom and Privacy, April 2005. `http://www.cfp2005.org/Program.html`.

[21] Stephanos Androutselis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution techonologies. *ACM Computing Surveys*, 36:335 – 371, December 2004.

[22] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley ACM PRESS BOOKS, New York, 1999.

[23] Metthias Bender, Sebastian Michel, Peter Triantoafillou, Gerhard Weikum, and Christian Zimmer. Minerva: Collaborative p2p search. *Proceedings of the 31st VLDB Conference, Trondheim, Norway*, 2005.

[24] Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedin systens 2.0 architecture. *White paper, Zero Knowledge Systems*, Desember 2000.

[25] David M. Burton. *Elementary Number Theory, Fift Edition*. McGraw-Hill, New York, USA, 2002.

[26] David Chaum. Untraceable electronic mail, return address, and digital pseudo-nyms. *Communications of the ACM*, 4, Februar 1981.

[27] Andew Christensen and Dan Faerch. Peeling the onion: Unmasking tor users. `http://www.fortconsult.net/images/pdf/tpr_100506.pdf`.

[28] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with freenet. *IEE Internet Computing*, pages 40 – 49, Januar 2002.

[29] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46–67, 2000.

[30] Brian F. Cooper. A content model for evaluating peer-to-peer searching techniques. *International Federation for Information Processing 2004*, pages 18 – 37, 2004.

[31] Brian F. Cooper and Hector Garcia-Molina. Ad hoc, self-supervising peer-to-peer search networks. *Transactions on Information Systems*, 23(2):169–200, April 2005.

[32] Phillip K. Dick. Letter to patricia warrick. Selected Letters of Philip K. Dick, 1977-1979, 1993.

[33] Roger Digeledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion routing. *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[34] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. *WPES'04*, pages 66–77, November 2004.

[35] Simon Garfinkel. *Database Nation, The Death of Privacy in the 21st Century*. O'Reilly and Associates Inc, Morris Street, Sebastopol, 2000.

[36] Paul Graham. How to make wealth. `http://paulgraham.com/paulgraham/wealth.html`.

[37] C Gulcu and G. Tsudik. Mixing e-mail with babel. *IEEE, Network and Distributed Security Symposium(NDSS 96)*, pages 2–16, Februar 1996.

[38] J. Kleinberg. The small-world phenomenon: an algorithmic perspective. *Cornell Computer Science Technical Report*, pages 99 – 176, 2000.

[39] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummandi, Sean Rha, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. OceanStore: An architecture for global-scale persistent storage. *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS)*, pages 190–202, November 2000.

[40] Bo Leuf. *Peer to Peer, Collaboration and Sharing over the Internet*. Addison-Wesley, 2002.

[41] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstrcutured peer-to-peer networks. *ICS'02, New York*, Juni 2002.

[42] Jens Mache, Melanie Gilbert, Jason Guchereau, Jeff Lesh, Felix Ramli, and Matthew Wilkinson. Request algorithms in freenet-style peer-to-peer systems. *Proceedingsg of the Second International Conference on Peer-To-Peer Computing (P2P'02)*, 2002.

[43] Sun Microsystems. Java - standard edition 1.5 website. `htpp://java.sun.com/javase/index.jsp`.

[44] Ulf Moller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster protocoll - version 2. `http://www.abditum.com/mixmaster-spec.txt`, Juli 2003.

[45] Haruki Murakami. The wind-up bird chronicle. Vintage Books.

[46] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. *SIGCOMM'01*, pages 161–173, August 2001.

[47] M. Elena Renda and Jamie Callan. The robustness of content-based search in hierarchical peer to peer networks. *ACM*, November 2004.

[48] Neil Stephenson. Cryptonomicon, 1993.

[49] Ion Stoica, Robert Morris, David Krager, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet application. *SIGCOMM'01*, pages 149–161, August 2001.

[50] Paul. F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, pages 482–494, Mai 1998.

[51] Olaf Titz. Why tcp over tcp is a bad idea. `http://sites.inka.de/sites/bigred/devel/tcp-tcp.html`.

[52] Dinesh C. Verma. *Legitimate Applications of PEER-TO-PEER Networks*. John Wiley and Sons, Inc, River Street, Hoboken, 2004.

[53] Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. The architeture of robust publishing systems. *ACM Transactions on Internet Technology*, 1:199–230, November 2001.

[54] D.J. Watts. Small-worlds: The dynamics of networks between order and randomness. *Princeton University Press*, 1999.

[55] Aaron Weiss. Hear no evil. NSW, September 2000.

[56] Hui Zhang, Ashish Goel, and Ramesh Govindan. Using the small-world model to improve freenet performance. *Computer Networks 46*, pages 555–574, Mai 2004.