P2P Lab2

JavaTella – a Java Gnutella Client

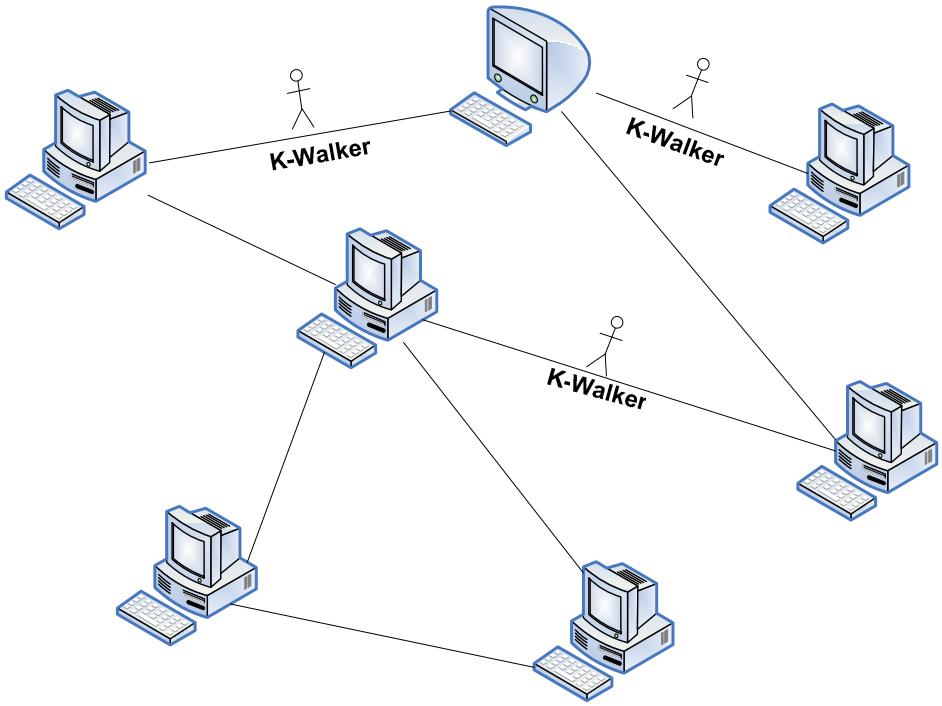
Claus Kirkegaard Clausen [20034435]
Frank Henningsen [20034454]
Jens Peter Troelsen [20034441]

INDLEDNING

I faget Peer to Peer skulle der udvikles et program, der kunne lytte og finde andre peers på netværket. Der skulle implementeres tilsvarende funktioner som 'Ping' og 'Pong' fra Gnutella protokollen. Det skulle ligeledes også være muligt at kunne interagere med systemet. Dette program kaldte vi JavaTella.

Denne opgave er en udvidelse til JavaTella, med mulighed for at kortlægge hvilke filer der er til rådighed i et system ved brug af en k-walker så søgninger kan udføres hurtigere, mere effektivt og uden at floode et netværk. Flooding betyder at der sendes beskeder til alle maskiner på et netværk.

Det skulle ligeledes være muligt at søge efter filer hos andre peers, samt at hente filer fra andre peers.



Figur 1: Forsøgsopstillingen

UDFØRELSE

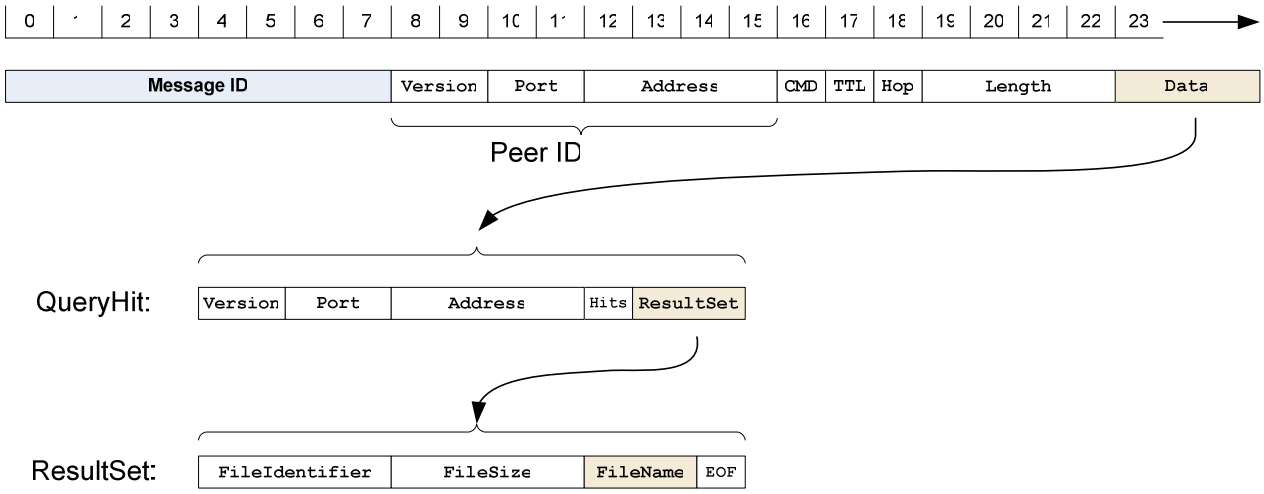
Vi har implementeret funktioner der gør det muligt at søge efter filer hos de peers den enkelte peer er forbundet til, samt hente disse filer. Ligeledes kan der udveksles tekstbeskeder mellem peers og det er muligt at se de peers den enkelte peer er forbundet til.

Vi har ligeledes implementeret en k-walker.K-walkeren gør, at hvis en peer får en søgning den ikke har et svar til, kan den videresende denne søgning til sin super peer. Denne super peer er kendetegnet ved, at være den peer der har flest delte filer. Det er derfor ikke sikkert at super peeren

har den ønskede fil, men sandsynligheden for det er større end hvis der var blevet valgt en tilfældig peer.

Pakkeformater

Vi har lagt os tæt op af Gnutella protokollen, med vores implementationer af bl.a. Query og QueryHit. Vi har en besked, Figur 2, der indeholder 22 faste bytes, og et variabelt data felt. Dette datafelt indeholder informationer om hvilken pakke der sendes, da Data f.eks kan indeholde en QueryHit.
QueryHit pakken er vist under, og indeholder adressen og porten på den peer der udsendte Query samt et ResultSet. ResultSet pakken indeholder en FileIdentifier der unikt beskriver filens id, filens størrelse samt filens navn og en EOF karakter. Filnavnet er lige som Data og ResultSet felterne har variable længde. EOF karakteren benyttes i UnMarshall() til at finde slutningen på filnavnet.



Figur 2

Hver besked indeholder en unik kode – en Global Unique Identifier – GUID. Denne GUID er opbygget af Message ID og Peer ID, tilsammen 16 bytes. Peer ID'en er, som det ses på figur 2, opbygget af JavaTella Version, Port samt senderens IP-adresse. Dette gør, sammen med Message ID'en, at hver GUID er unik, da der ikke er to peers der kan køre på samme maskine og benytte samme portnummer. Det er dog muligt at kunne køre flere klienter på samme maskine, men hver klient skal benytte et unikt portnummer.
Ved at benytte en GUID er det muligt at skelne beskeder fra hinanden. Når der sendes en besked, skiftes den oprindelige Peer ID ud med afsenders, mens Message ID'en er den samme. På denne måde kan en peer skelne om beskeden har været ved peeren før eller er ny.

.

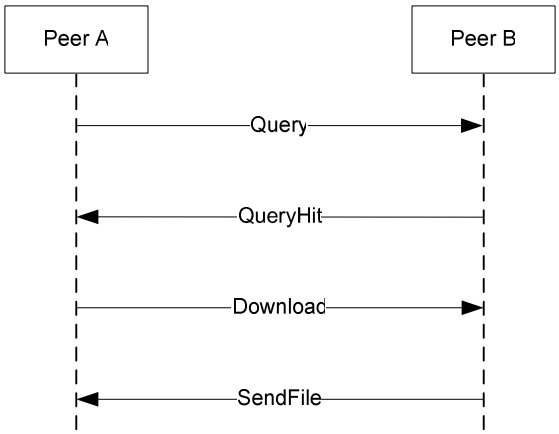
Design

Igennem forløbet med udviklingen af JavaTella har der været lagt vægt på at opbygget et modulært design der ville gøre implementering af nye udvidelser simpel. Et komplet diagram over JavaTellas design ligger som bilag 1.

Når JavaTella startes, oprettes en instans af Repository-klassen, der indeholder peer'ens delte filer. Det er denne klasse der benyttes til at lave en QueryHit pakke, når JavaTella modtager en Query på en bestemt fil. Når en Query modtages, oprettes en QueryResult der kan indeholde en eller flere ResultSet. For hvert hit der findes i Repository, oprettes en instans af ResultSet og Hits-feltet i QueryHit'et tælles en op.

Kommunikation mellem peers

Den simple kommunikation mellem to peers, hvor Peer A søger efter en fil Peer B har, forløber således:



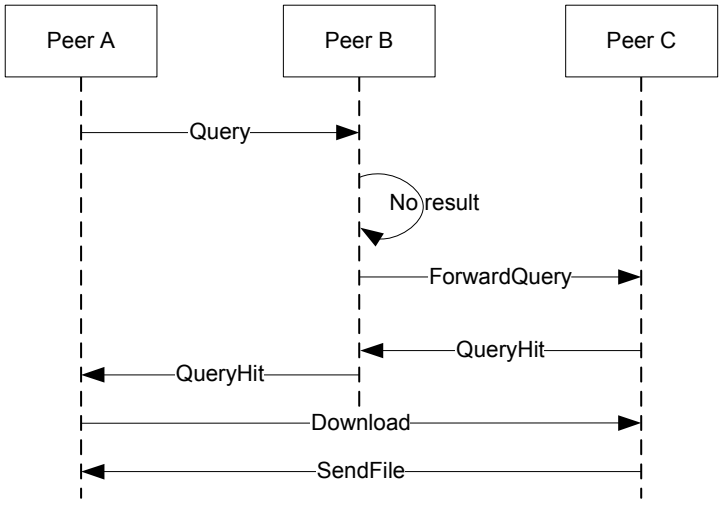
Figur 3

På figur 3 vises sekvensdiagrammet for kommunikation mellem to peers. Det antages at begge peers kender hinanden inden dette scenarie forløber.

Der bliver først sendt en Query fra Peer A til Peer B, der indeholder den tekststring der skal søges på. Peer B leder sit repository igennem, og hvis filen findes hos B, sendes et QueryHit tilbage til Peer A. Peer A kan nu vælge mellem de QueryHits der er kommet, hvis flere filer matcher kriteriet, eller han kan vælge at downloade filen fra Peer B.

På figur 4 vises sekvensdiagrammet for kommunikation mellem to peers og en Super Peer.

Peer A søger efter en fil hos Peer B, men denne fil har Peer B ikke. I stedet videresender Peer B søgningen til sin Super Peer, Peer C. Peer C har den ønskede fil, og sender denne tilbage til Peer A, gennem Peer B. Peer A kan herefter kontakte Peer C og downloade filen direkte.

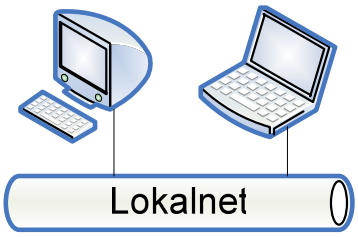


Figur 4

TEST

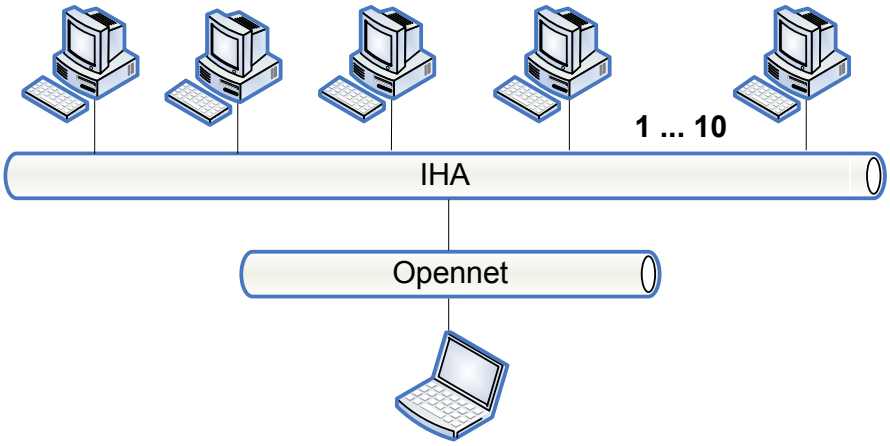
For at teste JavaTellas nye funktioner, har vi opstillet en række testcases der skal finde mulige styrker samt svagheder i programmet.

JavaTella blev testet på et lokalnet med to maskiner begge på samme subnet, en iMac med Mac OS X samt en Dell med Windows XP. Denne forsøgsopstilling ses på figur 5.



Figur 5

Den anden testopstilling foregik på Ingeniørhøjskolens netværk, ved brug af to maskiner fordelt på to forskellige subnets, begge Dell maskiner med Windows XP. Den ene maskine emulerede 10 klienter, mens den anden maskine kørte en enkelt klient. Forsøgsopstillingen ses på figur 6 .



Figur 6

Hvis en test ikke kan godkendes, bliver det markeret med et : ✗

Hvis en test kan godkendes, bliver dette markeret med et : ✓

Test cases

#1 Overførsel

Disse testcases blev udført med et scenarie som vist på figur 5.

Case	Forventet Resultat	Faktisk Resultat	Godkendt
En søgning er foretaget, en peer har den ønskede fil og overførslen kan starte.	Filen bliver korrekt overført til den peer der søgte efter filen.	Den ønskede fil overføres korrekt.	✓
En søgning er foretaget, en peer har flere filer der matcher søgningen, og	Brugeren bliver vist en liste af mulige filer han kan hente, og vælger filen,	De mulige filer nummereres og vises i en liste, hvor brugeren vælger	✓

brugeren ønsker at hente en bestemt.	som overføres.	hvilken fil der ønskes overført	
En søgning er foretaget og en peer har den ønskede fil. Overførslen startes, afbrydes kort på grund af netværksfejl, men fortsættes igen.	Filen overføres korrekt, da TCP-protokollen benyttes til filoverrførsler.	Filoverførslen stopper midlertidigt, men når netværksforbindelsen er genetableret fortsættes overførselen og filen overføres uden fejl.	✓

#2 K-walker

Disse testcases blev udført med et scenarie som vis på figur 6.

Case	Forventet Resultat	Faktisk Resultat	Godkendt
Udfør en søgning på en fil der eksistere hos en peer	Peer'en med den ønskede fil svarer at den har filen	Der kommer svar fra den peer der har den ønskede fil.	✓
Udfør en søgning på en fil der eksistere hos mere end en peer.	De peers der har filen svarer tilbage og disse resultater præsenteres så brugeren kan vælge hvilken fil der skal hentes.	Der kommer svar fra de peers der har den ønskede fil, og brugeren kan vælge hvilken fil der ønskes downloaded.	✓
Udfør en søgning på en fil der ikke findes hos nogle peers.	Brugeren bliver præsenteret for en forklarende fejlbesked.	Intet.	✗
En peer søger efter en fil hos en anden peer. Denne peer har ikke den efterspurgte fil, men videresender forespørgslen til sin super peer, der har denne fil.	Super peer'en modtager denne søgning, og søger sit repository igennem, finder den ønskede fil og sender et svar tilbage.	Super Peer'en modtager forespørgslen og finder den ønskede fil i sit repository og sender en QueryHit tilbage.	✓
En peer har en super peer. En ny peer kommer på netværket, og denne peer er bedre end den hidtidige super peer	Denne peer vælge som super peer.	Den nye peer bliver valgt til super peer.	✓

KONKLUSION

Vi har fået implementeret søgemekanismer samt en meget simpel k-walker algoritme. Det er muligt at søge og hente filer fra andre peers, over flere subnets. Hvis en peer ikke kan svare på en forespørgsel, sendes denne videre til en Super Peer fundet ved hjælp af den simple k-walker funktion.

Vi testede JavaTella ud fra en række testcases der skulle vise systemets styrker samt svagheder. En enkelt test fejlede, da der ikke kommer en besked til brugeren om at søgningen ikke gav et resultat. Dette kunne dog simpelt implementeres ved at benytte en tråd med en timer, og når denne timer udløber og der ikke er kommet svar, skulle beskeden vises til brugeren.

JavaTella er åben for mange nye mulige udvidelser. En implementation af en adaptiv k-walker algoritme, der basere sit output på tidligere søgningers resultater, en brugergrænseflade som brugere kunne betjene programmet gennem og bedre error-og flowcontrol er bare nogle af disse.

Bedre håndtering af peer-listen vil også være en vigtig funktion at implementere. Denne skulle sende keep-alive pakker til de peers den enkelte peer var forbundet til, og på denne måde også få opdateringer om peers nye filer i repository.

En centraliseret lookup-server ville også hjælpe JavaTella til at opnå en bedre performance da peers nemmere ville kunne finde andre peers at komme i kontakt med. En centraliseret lookup-server har tidligere vist at være et singel-point-of failure for systemer der distribuere materiale med copyright. For legale systemer, der f.eks. kunne bruges i en større virksomhed til at distribuere opdateringe eller nightly-builds ville sådan en server gøre nytte.