P2P Lab1

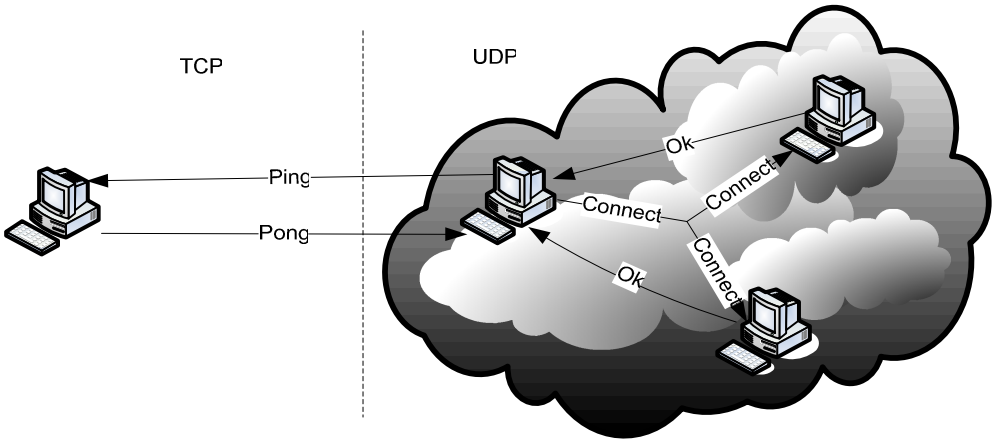
JavaTella – a Java Gnutella Client

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.

I vores løsning kaldet JavaTella, har vi forsøgt at holde os til Gnutella specifikationen, og udviklet en simpel klient til at finde andre peers, både på et LAN samt peers udenfor et LA. Dette løses ved at benytte to forskellige kommunikations former; TCP og UDP. På figur 1 herunder, ses det hvordan systemet er bygget op.



Figur 1 viser forsøgsopstillingen

UDFØRELSE

Vi har forsøgt at implementere klienten som specificeret i Gnutella "standarden". I specifikationen er der specificeret 2 vigtige kommandoer kaldet ping og pong, der har til opgave at få peers til at finde hinanden og udveksle internet adresser.

Et problem ved disse kommandoer er, at det er nødvendigt for en klient at kende til modpartens adresse. Da Gnutella specifikationen ikke beskriver hvordan peers finder hinanden, har vi valgt at implementere en UDP løsning baseret på Broadcast og Multicast. UDP løsningen gør det muligt for peers at finde hinanden uden at kende til hinanden på forhånd, ved løbende at udsende en besked på netværket der fortæller at den er tilstede. Når andre peers hører denne besked, tilføjes klienten til deres interne liste af peers.

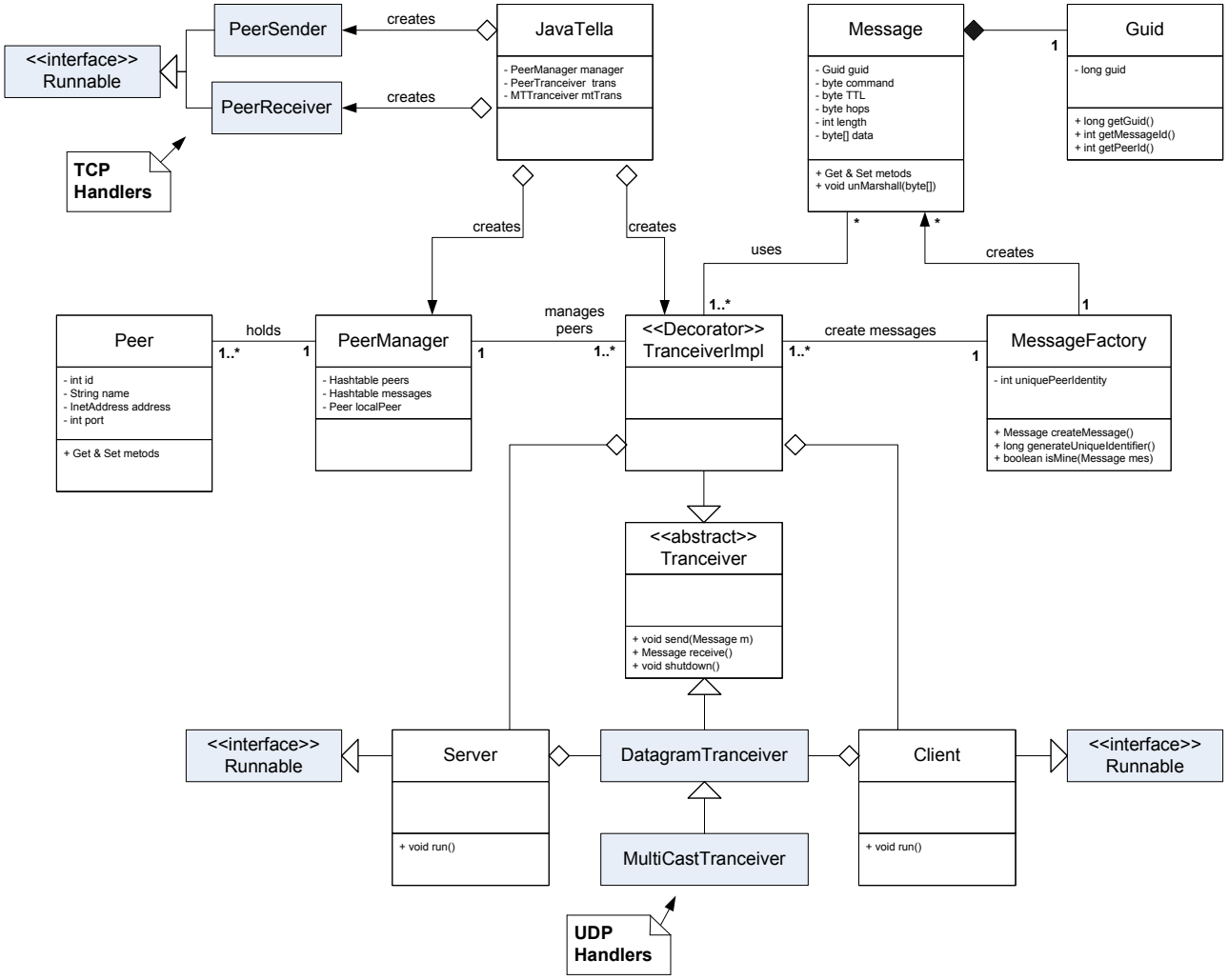
Et problem ved at bruge UDP Broadcast, er at beskederne kun sendes til et subnet på netværket. Hvis 2 klienter befinder sig på 2 forskellige subnet, eller er adskilt af internet, vil de ikke kunne finde hinanden.

En fordel ved IP-Multicast er at den ikke begrænser sig til et subnet, men i stedet bruger en IP-gruppe til at få peers til at kommunikere med hinanden. For at modtage beskeder fra en IP-gruppe, tilmelder en klient sig til gruppen, hvorefter klienten modtager beskeder sendt fra serveren. På et kontrolleret netværk, hvor det er muligt at bestemme og uddele netværksadresser, vil Multicast løsningen være fin løsning, en på et større netværk som Internet, er det ikke muligt for 2 peers at finde hinanden.

Vi har valgt at lave et lag delt design, for at indkapsle funktionaliteten, og gøre det muligt at anvende flere forskellige netværk protokoller.

Til at holde styr på listen af peers og beskeder modtaget har vi lavet en PeerManager klasse. Hver peer i listen er repræsenteret som et Peer objekt, og besidder et unikt ID til at kunne kende forskel på hinanden. Et ID er 64 bit langt og består af 2 elementer: messageID og peerID. Disse 2 elementer gør det muligt at afgøre om en besked er sendt før, samt hvem der har sendt beskeden.

For at sende beskeder til hinanden bruges Message klassen, der specificerer protokol-headeren samt metoder til klargøre beskeder til transmission på netværket. TransceiverImpl klassen er en dekorator for netværksinterfacet, der internt besidder både en klient og server der gør brug af netværket let for klient applikationen.



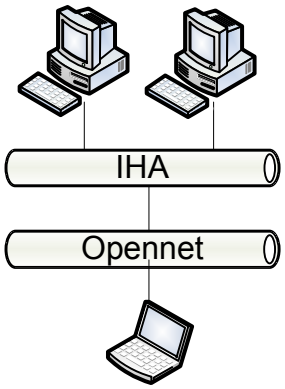
Figur 2

TEST

JavaTella blev testet på Ingeniørhøjskolens netværk, ved brug af tre maskiner fordelt på to forskellige subnets. Forsøgsopstillingen ses på figur 3.

De maskiner på subnettet 'IHA' kunne finde og tilføje hinanden ved hjælp af multicast, mens maskinen på 'Opennet' ikke opfangede disse. Ved at benytte broadcast, kunne maskinen på 'Opennet' ping'e og tilføje maskinerne på 'IHA' til sin liste over andre peers. Ligeledes kunne maskinerne på 'IHA' ping'e og tilføje maskinen på 'Opennet'.

Ved at sætte en firewall på maskinen på 'Opennet', var det ikke muligt at ping'e andre maskiner, da firewallen lukkede af for trafikken på den benyttede port.



Figur 3

KONKLUSION

Vi har udviklet et simpelt program, hvor en peer kan finde og tilføje andre peers, enten ved brug af multicast eller ved at ping'e andre peers.

Designet af programmet kunne sagtens have været pænere, men vi har forsøgt at forberede designet på, at det skal kunne udbygges.

I testen viste det, at programmet på et subnet hurtigt kunne finde andre peers og tilføje disse til peerens egen liste. Ved at benytte to forskellige subnets kunne vi se, at de multicast forespørgsler der blev sendt, ikke kom uden for subnettet. Testen vist også, at hvis vi satte en firewall på, ville denne blokere alt trafik.