Generella Peer-to-Peer plattformar

Författare: Johan Furberg
Johan Ölund

Handledare: Tomas Bjurman (TietoEnator DevCon)
Mikael Rännar (Umeå Universitet)

Abstract

This master's thesis is an inquiry to investigate if general Peer-to-Peer (P2P) platforms are an alternative to the traditional client-server model.

General P2P-platforms are used to build different applications on the same platform, to reduce the time and cost of developing new P2P-applications. General P2P-platforms is a relative new concept and therefore the development is rapid. JXTA is a general P2P-platform and has served as a representative in this report.

A major problem in today's networks is the overload of the servers while resources at the client side remain unused. P2P will reduce the problem by building up a virtual, dynamic and flexible network with no central resources, where clients can communicate directly with each other.

There are three main problems with P2P that have to be solved; security, bandwidth and attitude. The use of general P2P-platforms will probably speed up the development of the P2P-technology and therefore facilitate finding the solution to the main problems.

Given that these problems can be solved, general P2P-platforms will be a competitive alternative to the client-server model. Today, general P2P-platforms are not an alternative to the client-server in practise. None of the platforms that exist today have reached the stability needed to be considered a long time solution.

Sammanfattning

Peer-to-Peer (P2P) är idag ett av de hetaste ämnena inom datorvärlden. För att underlätta utvecklingen av P2P-applikationer kan generella P2P-plattformar utvecklas, ett sådant exempel är JXTA. Är då dess generella P2P-plattformar ett alternativ till den traditionella klient-servermodellen? Idag går utvecklingen med en rasande fart. Men det finns fortfarande ett antal problem som måste lösas. Det största problemet idag är att det inte finns någon plattform som är tillräcklig stabil för att ses som en seriös alternativ till klient-servermodellen.

Innehållsförteckning

1 Inledning

I dagens läge ökar antalet användare av Internet konstant. Kraven på servrarna som tillhandahåller de tjänster som finns tillgängliga på Internet blir allt större. För att avlasta servarna och utnyttja klienternas resurser på ett bättre sätt kan Peer-to-Peer teknologin användas.

Redan i början av datornätverkets tid var Peer-to-Peer teknologin en viktig datakommunikationsteknik. Peer-to-Peer (P2P) är ett begrepp som anger att ett nät är organiserat efter en mer decentraliserad modell. Usenet var en av de första P2P-applikationerna och skapades i slutet av 70-talet. Applikationens uppgift var att distribuera information mellan två datorer. Det som var intressant med Usenet var att det inte fanns någon central server för distributionen av informationen. Kommunikationstekniken P2P används som grund i många Internettjänster så som meddelandetjänster och informationsdelningstjänster.

I mitten av 90-talet skapades ICQ. Med hjälp av ICQ kan användarna på ett snabbt sätt kommunicera med varandra. ICQ är en P2P-applikation men använder en central server för att hantera information om användarna. Men däremot skickas alla meddelanden direkt mellan klienterna.

I slutet av 90-talet blev konceptet P2P allmänt känt genom fildelningstjänster Napster. Napster kom främst att användas för att sprida illegala kopior av mediafiler. Även Napster använder en central server som i det här fallet har till uppgift att lagra vilka filer som finns tillgängliga i nätverket och var de finns lagrade. Det var detta som gjorde att de kom att åtalas för att sprida illegala kopior. Efter domen mot Napster har det kommit ett antal liknande fildelningstjänster som saknar centrala servrar. Exempel på sådana är KaZaA och Gnutella.

Sommaren 2000 startade Sun Microsystems forskningsprojektet JXTA, som resulterade i en generell P2P-plattform. Målet med detta projekt var att skapa ett antal öppna protokoll, för att alla enheter i en distribuerad miljö ska kunna kommunicera på ett enkelt och säkert sätt. Dessa protokoll är definierade i XML och är programspråks- och plattformsoberoende.

2 Syfte

Syftet med examensarbetet är att utreda om generella P2P-plattformar är ett alternativ till den traditionella klient-servermodellen. Detta kommer framförallt att göras genom en granskning av den generella P2P-plattformen JXTA. Examensarbetet utförs på uppdrag av Tieto Enator i Skellefteå.

Rapporten kan användas som en introduktion till P2P i allmänhet och JXTA i synnerhet. Dessutom som en utredning av möjligheterna med generella P2P-plattformar.

Denna rapport vänder sig till personer som har grundläggande kunskap inom programmering och datorkommunikation.

3 Metodbeskrivning

För att kunna utreda om generella P2P-plattformar är ett alternativ till den traditionella klient-servermodellen har utredningen delats upp i två delar. Den första delen innehåller en teoretisk utredning i ämnet och den andra delen består av tester.

Den första delen började med en teoretisk inläsning av P2P-teknologier. Arbetet fortsatte med en teoretisk inläsning av uppbygganden av JXTA, som är en generell P2P-teknologi. Dessa teoretiska inläsningar fick sedan ligga som grund till en dokumentation av P2P-teknologier i allmänhet och uppbyggnaden av JXTA.

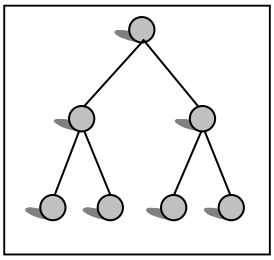
Den andra delen har delats upp i två delar, prestandatester och testning av generella P2P-plattformar i praktiken. Prestandatesterna utfördes för att kunna jämföra prestanda vid informationsöverföring mellan P2P-modellen och klient-servermodellen, där kommunikationsmetod och teknologin varierats. Vid testning av generella P2P-plattformar i praktiken implementerades ett informationsspridningssystem med hjälp av JXTA.

För att sammanfatta utredningen har vi diskuterat om hur generella P2P-plattformar förhåller sig till den traditionella klient-servermodellen, vilka tillämpningsområden som är lämpliga och vad P2P-teknologin har för framtidsutsikter.

4 Topologier för distribuerade system

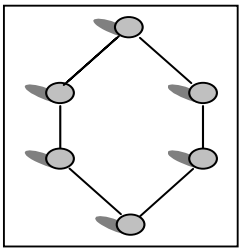
Det finns flera metoder att organisera datorer i ett distribuerat system. De fyra vanligaste metoderna är hierarkiska topologier, ringtopologier, centraliserade topologier och decentraliserade topologier. Det förekommer också kombinationer av dessa topologier.

4.1 Hierarkiska



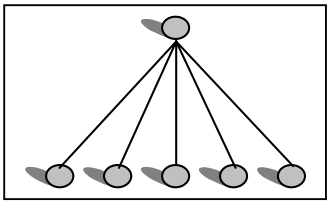
Den hierarkiska topologin är en av de topologier som används mest på Internet idag. Denna typ av topologi distribuerar funktionaliteten och informationen på hierarkiskt sätt. Ett av de mest välkända exemplen på distribuerade system som använder denna topologi är Domain Name System (DNS). Hierarkiska systems största fördel är förmågan att skala. Det är också relativt lätt att lägga till noder på en nivå för att fördela belastningen bättre. Den hierarkiska topologin är delvis feltolerant men om rotnoden slutar att fungera drabbas hela systemet. [TOP101] [TOP202]

4.2 Ring



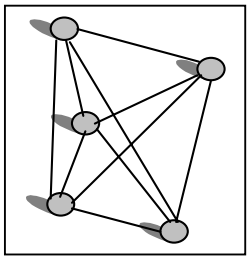
För att avlasta en central enhet kan ringtopologin användas. Ringtopologin fördelar belastningen rättvist på flera centrala enheter. Dessa centrala enheter är då placerade logiskt i en ring. Olikt andra topologier är dessa enheter oftast belägna relativt nära varandra i nätverket. Denna topologi används oftast när hela systemet har samma ägare. [TOP101] [TOP202]

4.3 Centraliserad



Den centraliserade topologin är mer känd under beteckningen klient-servermodellen. Denna topologi är den mest använda på Internet idag. Informationen och funktionaliteten i denna topologi distribueras med hjälp av en central enhet. Fördelen med denna topologi är att all funktionalitet och information är samlat i en central enhet och därmed är lätt att administrera och skydda. Men detta är också en nackdel eftersom funktionaliteten i systemet försvinner om den centrala enheten slutar att fungera. [TOP101] [TOP202]

4.4 Decentraliserad



Den decentraliserade topologin har ingen central enhet utan varje nod i systemet kan kommunicera med alla andra noder. Eftersom ingen av noderna har någon särställning blir decentraliserade system mer feltoleranta. Utbyggnad av decentraliserade system underlättas av att det är lätt att införa nya noder i nätverket. Men det är även ett av de stora säkerhetshoten. Ett annat stort problem med decentraliserade system är svårigheten att administrera dessa. I teorin skalar ett decentraliserat system bra. Men i praktiken uppfylls inte detta om det krävs mer information att hålla systemet sammanhängande när systemet växer. Peer-to-Peer kan beskrivas som en typ av decentraliserad topologi. [GR02] [TOP101] [TOP202]

5 Peer-to-Peer (P2P)

Även om den traditionella centraliserade arkitekturen har fördelar när det gäller administration och säkerhet, finns ett antal nackdelar. Hela arkitekturen bygger på att servern utför det huvudsakliga arbetet och klienten har en passiv roll. När antalet användare ökar så ökar också belastningen på servern. Med denna modell utnyttjas oftast serverns resurser maximalt, däremot är klienternas resurser näst intill outnyttjade. För att avlasta servern och utnyttja klienternas resurser på ett bättre sätt kan P2P-teknologin användas. [WI02]

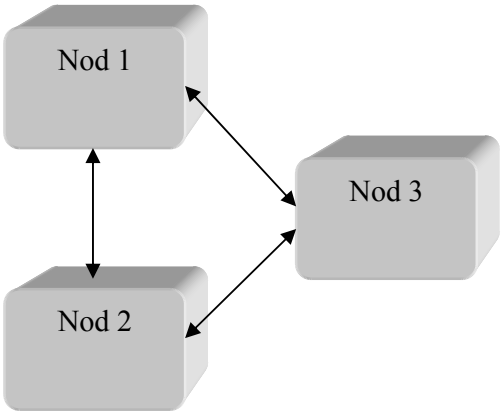
Bokstavligt tolkat är begreppet Peer-to-Peer två jämbördiga noder som kommunicerar fritt med varandra. Men denna tolkning av begreppet uppfyller inte hela definitionen. Det finns ett flertal kommunikationstekniker som uppfyller fri kommunikation mellan två parter, som inte innefattas i begreppet P2P. Ett exempel på en sådan kommunikationsteknik är TCP-protokollet. Definitionen av P2P varierar betydligt beroende på källa. I denna rapport har P2P definierats på följande sätt, med utgångspunkt från artikeln ”What Is P2P ... And What Isn't” av Clay Shirky.

Ett P2P-nätverk ska uppfylla följande krav [WHATISP2P00]:

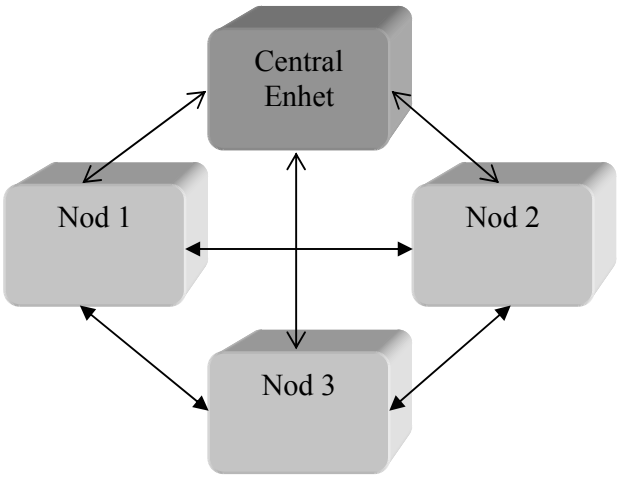
- Noderna i nätverket ska vara signifikant autonoma.
- Alla noder ska kunna kommunicera med varandra.
- Nätverket ska stödja dynamisk anslutning av noder.
- Noderna ska identifieras oberoende av den fysiska adressen.

P2P kan beskrivas som en decentraliserad arkitektur där noderna i nätverket är mer jämställda i jämförelse med den centraliserade arkitekturen. Kommunikationen kan alltså ske mellan alla noder i nätverket vilket inte är fallet i den centraliserade arkitekturen. Funktionaliteten i systemet distribueras inte av en central enhet utan är fördelad mellan noderna i nätverket. Fördelen med denna modell är att resurserna ökar när antalet användare ökar och att modellen är mer feltolerant.

Generellt finns det två möjligheter att bygga upp P2P-system. Den ena formen är en renodlad P2P-modell där det inte finns någon särställning mellan noderna, utan alla noder i nätverket är jämställda (Figur 1). Den andra formen är en hybrid mellan den centraliserade topologin och den decentraliserade topologin, där den centraliserade delarna har samordnande funktionalitet (Figur 2). Ett exempel på detta kan vara att lagra vilka noder som finns tillgängliga i nätverket. Men i övrigt kommunicerar noderna direkt med varandra. [P2PCOMP]



Figur 1 – Den renodlade modellen för uppbyggnad av P2P-system.



Figur 2 – Hybridmodellen för uppbyggnad av P2P-system.

Begrepp

Det finns ett antal viktiga begrepp som förekommer inom P2P-teknologin. Dessa termer är viktiga för förståelsen och beskrivs i detta avsnitt.

5.1.1 Nod (Peer)

En nod i ett P2P-nätverk är den minsta processande delen i nätverket. Eftersom en applikation i ett P2P-nätverk kan sträcka sig över ett flertal noder kan en nod inte förknippas med en applikation. En nod kan inte heller förknippas med en enhet i nätverket eftersom att en enhet kan ha flera noder. Enligt Brendon J. Wilson definieras en nod på följande sätt [WI02:16]:

> *"Any entity capable of performing some useful work and communicating the results of that work to another entity over a network, either directly or indirectly."*

5.1.2 Grupp (Peer group)

En grupp är en mängd noder i ett P2P-nätverk som har ett gemensamt intresse eller mål. En grupp kan erbjuda tjänster som bara medlemmar i gruppen har tillgång till. På detta sätt kan noderna logiskt delas in i grupper. För att öka säkerheten kan en grupp ha regler för vilka som har tillåtelse att bli medlem i gruppen. Enligt Brendon J. Wilson definieras en grupp på följande sätt [WI02:18]:

> *"A set of peers formed to serve a common interest or goal dictated by the peers involved. Peer groups can provide services to their member peers that aren't accessible by other peers in the P2P network."*

5.1.3 Generell P2P-plattform

En generell P2P-plattform innehåller den grundläggande funktionaliteten som krävs för att utveckla P2P-applikationer. Detta innebär att funktionaliteten är gjord på ett sådant sätt att den passar många olika typer av applikationer.

5.1.4 Direkt kommunikation (Point-to-Point)

Direkt kommunikation är det mest naturliga sättet för två noder att kommunicera. Kommunikationen bygger på att två noder i P2P-nätverket kommunicerar direkt med varandra utan att andra noder är inblandade. Denna kommunikationsmetod förutsätter att sändaren har kännedom om mottagaren. Det krävs också att det går att skapa en direkt kommunikationslänk mellan noderna. [BRGOKR02]

5.1.5 Indirekt kommunikation (Cross-Linked)

Indirekt kommunikation används när direkt kommunikation inte kan användas mellan två noder. Denna kommunikationsmetod tillåter två noder att kommunicera med hjälp av andra noder i nätverket. Detta kan vara nödvändigt om det finns hinder i nätverket som hindrar direkt kommunikation mellan noderna eller om noderna inte har kännedom om varandra. [BRGOKR02]

5.2 Kommunikation

För att kommunikation ska kunna ske mellan noder i P2P-nätverket krävs det först och främst att de känner till varandra. Ett annat problem är de hinder som finns i nätverket.

5.2.1 Sökmetoder

För att noder i ett P2P-nätverk ska kunna kommunicera och utföra någon som helst meningsfull uppgift krävs det kännedom om andra noder och tjänster i nätverket. Ett P2P-nätverk är dynamiskt och nya noder kan lätt ansluta. En viktig fråga som då måste besvaras är hur de olika noderna i nätverket ska kunna få kännedom om varandra och de tjänster som erbjuds. Det finns generellt tre olika sökmetoder för att hitta andra noder och tjänster i nätverket, men kombinationer av dessa är vanliga. [WI02]

Passiv sökning

Den passiva sökmetoden innebär ingen sökning i nätverket, utan bygger på att resultat från andra sökmetoder lagras och därigenom kan sökningen ske passivt bland dessa resultat. Denna sökmetod är definitivt den snabbaste av dem tre. För att minska nätverkstrafiken är denna sökmetod nödvändig. Men det finns vissa nackdelar med denna sökmetod. En nackdel är att resultaten från tidigare sökningar blir allt fler och eftersom lagringsutrymmet inte är oändligt är detta ett problem. En annan nackdel är att resultaten från tidigare sökningar kan vara ogiltiga. Om antalet ogiltiga resultat blir för stort ger det

upphov till en ökad nätverkstrafik eftersom att noden först försöker kommunicera med en inte tillgänglig resurs över nätverket. Det finns olika metoder för att minska problemet med ett begränsat lagringsutrymme och ogiltiga poster. En metod kan vara att de äldsta posterna ersätts med nya när lagringsutrymmet är fullt. En annan metod kan vara att ge varje post en tidstämpel som anger hur länge posten är giltig. På så sätt kan de poster som inte längre är giltiga med avseende på tidstämpel raderas. [WI02]

Aktiv sökning

Aktiv sökning bygger på att den sökande noden själv aktivt söker efter resurser i nätverket. Den sökande noden skickar ut en förfrågan på nätverket med hjälp av kommunikationsmetoden broadcast eller multicast. De noder och tjänster som nås av förfrågan skickar tillbaka information om dem själva. Nackdelen med denna metod är att restriktionerna på vissa nätverk ofta är hårda vilket medför att räckvidden på sökningen blir begränsad. [WI02]

Indirekt sökning

Den indirekta sökmetoden bygger på att den sökande noden tar hjälp av noder den redan känner till för att hitta resurser i nätverket. Den sökande noden skickar en förfrågan till dessa noder. Noderna försöker sedan besvara förfrågan genom att skicka tillbaka den information de vet om nätverket. I vissa fall kan de i sin tur skicka vidare förfrågan. På så sätt ökar möjligheten att upptäcka nya noder och resurser i nätverket. Nackdelar med denna sökmetod är att den är långsam och att det krävs kontroller för att förhindra okontrollerad spridning av sökningsmeddelanden. [WI02]

5.2.2 Hinder i nätverket

Ett stort problem för P2P-nätverk är det kan finnas olika hinder i nätverket. De största hindren är brandväggar och Network Address Translation (NAT). Dessa hinder används oftast för att skydda de privata nätverken från otillåten nätverkstrafik. Eftersom ett P2P-nätverk ofta sträcker sig utanför de privata nätverken medför det att noderna inte alltid kan eller har tillåtelse att kommunicera direkt med varandra. [WI02]

En brandvägg kan blockera specifika protokoll, kommunikationsportar, och även speciella mönster i meddelanden. De flesta brandväggar tillåter HTTP-kommunikation som initieras från det privata nätverket. Detta medför att klienter på det privata nätverket kan kommunicera med t.ex. webbservrar utanför det privata nätverket medan nätverket är skyddat från otillåten access utifrån. [WI02]

För att minska användningen av antalet publika IP-adresser och minska kostnaden används tekniken Network Address Translation (NAT). Denna teknik används oftast av en router mellan det privata nätverket och det publika nätverket. Routern fungerar då som en mellanhand för trafiken mellan nätverken. Detta innebär att det endast behövs en publik adress för att representera en grupp av datorer. Tekniken förbättrar då säkerheten genom att erbjuda endast en kommunikationspunkt mellan det interna nätverket och det externa nätverket. [WI02]

Både brandväggar och NAT medför ett problem för P2P-nätverk. Problemet är att en nod utanför det privata nätverket, som skyddas av en brandvägg eller NAT, inte kan initiera kommunikationen med noder på det privata nätverket. En lösning på detta problem är att det finns någon typ av nod som lagrar inkommande meddelanden till noden på det privata nätverket. På så sätt kan noden på det privata nätverket regelbundet kontrollera med den externa noden om det finns några inkommande meddelanden, detta görs idag framförallt med HTTP-protokollet. Men det finns vissa nackdelar med denna metod. En nackdel är att väntetiden för ett meddelande kan bli onödigt långt. En annan nackdel är att onödig nätverkstrafik uppstår när det inte finns några inkommande meddelanden på den externa noden. Förutom dessa nackdelar blir P2P-nätverket mer centraliserat. [WI02]

5.3 För- och nackdelar

Fördelen med P2P-teknologin är att resurserna i nätverket utnyttjas på ett bättre sätt. Dessa outnyttjade resurser finns oftast på klientsidan i klient-servermodellen, exempel på sådana resurser kan vara processorkraft, lagringsutrymme och bandbredd. Belastningen av de centrala delarna av nätverket minskar. Eftersom att det är nodernas resurser som är byggstenarna för den totala resursen i P2P-nätverkt innebär detta att när antalet noder öka, ökar också resurserna i nätverket. Men detta uppfylls inte helt i praktiken eftersom att det då krävs mer resurser för att binda samman ett större nätverk. [BRGOKR02] [WI02]

En annan fördel är att funktionaliteten i nätverket inte är bunden till en specifik nod, som i klient-servermodellen, och är därför inte lika känsligt för störningar. Flera noder kan tillhandahålla samma tjänst vilket gör tjänsten redundant. Noderna i P2P-nätverk är till stora delar oberoende av varandra vilket innebär ett stabilare nätverk.

P2P-nätverket är dynamiskt och kan växa fritt, vilket är en fördel. Men detta medför även en nackdel genom att det blir svårt att kontrollera och administrera nätverket. Den dynamiska strukturen i nätverket innebär också att det icke är deterministiskt. [WI02]

Rena P2P-lösningar innebär en relativt stor säkerhetsrisk eftersom det inte finns någon central enhet som kan bestämma vad som är rätt och fel. Detta går delvis att lösa med hjälp av någon form av demokrati mellan noderna. Hybridlösningar har däremot en större möjlighet att erbjuda säkra system. Detta eftersom att systemet innehåller en samordnande enhet som kan ha bestämmanderätt.

Versionsproblem är också ett problem i P2P-nätverk eftersom att informationen distribueras fritt i nätverket. Detta innebär att det kan finnas många versioner av informationen. Men detta går att lösa med hjälp av synkronisering, vilket lämpar sig relativt bra i P2P-nätverk.

6 JXTA

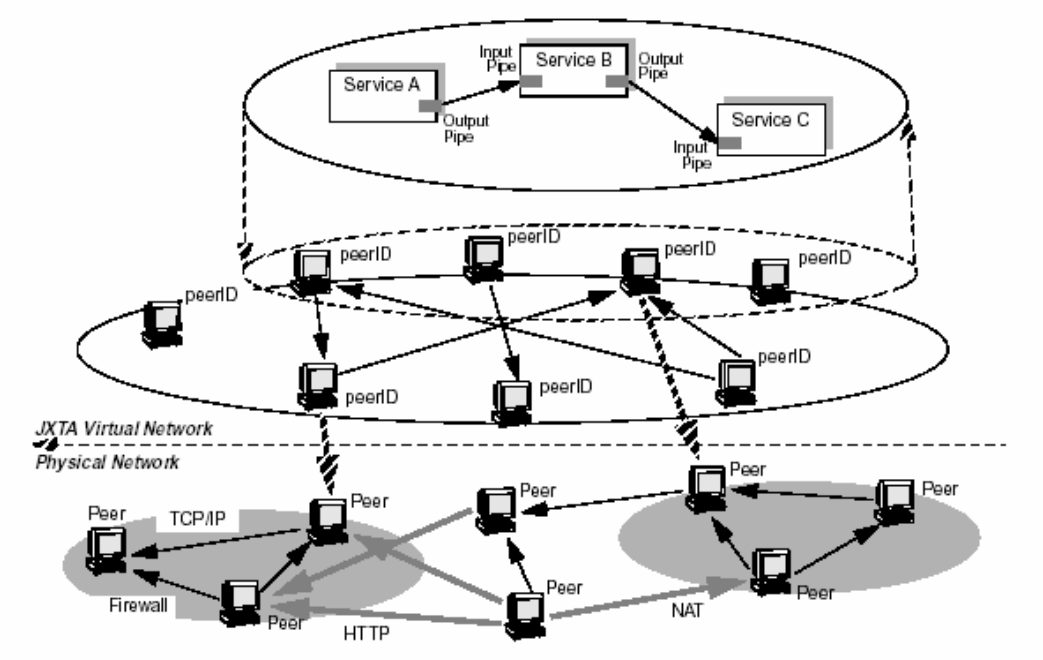
Bill Joy har varit en ledande forskare på Sun Microsystems sedan 1982. Innan dess har han varit med och utvecklat Berkeley UNIX operativsystem som banade väg för konceptet ”öppen källkod”. Bill Joy och hans kollega Mike Clary startade sommaren 2000 forskningsprojektet JXTA. JXTA är en förkortning av det engelska ordet ”juxtapose”, som betyder ”sida vid sida”.

Idag bygger de flesta företag sina egna P2P-plattformar med tillhörande protokoll. Detta innebär att plattformarna inte blir kompatibla med varandra. Målet med JXTA var att försöka lösa detta problem genom att skapa en generell P2P-plattform. Plattformen bygger upp en modell för hur noder i ett nätverk kan kommunicera, dela resurser, grupperas och övervakas.

Sun Microsystems släppte en första version av JXTA specifikationen och en referensimplementation i Java till allmänheten den 25 april 2001. Vid ungefär samma tidpunkt startades också webbplatsen www.jxta.org. Webbplatsen är en samlingsplats för fortsatt utveckling och för att erbjuda utvecklare att delta i projektet JXTA. Samlingsplatsen har en styrelse på tre personer, varav en person utses av Sun.

En av grundtankarna med JXTA är att alla typer av nätverksanslutna enheter, från superdatorer till mobiltelefoner, ska kunna kommunicera med varandra. Teknologin abstraherar bort den underliggande nätverkstopologin och skapar ett virtuellt nätverk där noderna kan kommunicera och samarbeta med varandra (Figur 3). Detta även om det finns hinder i nätverket. JXTA definieras på följande sätt [JXTA02]:

> *“JXTA technology is a set of open protocols that allow any connected device on the network ranging from cell phones and wireless PDAs to PCs and servers to communicate and collaborate in a P2P manner.”*



Figur 3 – Det virtuella nätverket.

Målet med JXTA

Målet med utvecklingen av JXTA är att skapa en generell plattform för P2P-applikationer. Plattformen ska vara programspråksoberoende, operativsystemsoberoende och oberoende av nätverksteknologi. En annan viktig aspekt under utvecklingsarbetet av JXTA har varit att tänka på säkerhetsfrågor. [BRGOKR02]

De konceptuella målen har varit att [BRGOKR02]:

- Använda grupper för att logiskt dela in noderna i nätverket med avseende på deras intresse och mål.
- Erbjuda gruppen möjlighet att skapa sin egen säkerhetspolicy.
- Distribuera information om noder och resurser via nätverket.
- Erbjuda en infrastruktur för kommunikation mellan noder. Oavsett om det finns hinder i nätverket.
- Skapa en möjlighet för noderna att övervaka varandra inom gruppen.

6.1 Begrepp

I detta avsnitt beskrivs de viktigaste begreppen i JXTA. Vissa av dessa har redan beskrivits under P2P-avsnittet men i detta avsnitt beskrivs de i större detalj och med avseende på JXTA.

6.1.1 Nod (Peer)

En nod i ett JXTA-nätverk (P2P-nätverk) är den minsta processande delen i nätverket. En nätverksansluten enhet kan inneha flera noder. Exempel på enheter kan vara servrar, arbetsstationer, mobiltelefoner och andra enheter. En nod kan inte associeras med en användare eftersom en användare kan ha tillgång till flera noder och en nod kan ha flera användare. Det finns tre typer av noder i ett JXTA-nätverk. Dessa är enkla noder, mötesplatsnoder och routernoder. [BRGOKR02] [WI02]

Enkel Nod (Simple Peer)

En enkel nod har endast till uppgift att betjäna användaren. Den ansvarar inte för att hantera information om resurser i nätverket. Noden fungerar inte heller som någon mellanhand mellan andra noder i nätverket. Det vanliga är att denna typ av nod befinner sig bakom olika hinder i nätverket, som brandväggar och NAT. [WI02]

Mötesplatsnod (Rendezvous Peer)

Mötesplatsnodens huvuduppgift är att distribuera information i JXTA-nätverket. Informationen distribueras genom att noder dynamiskt ansluter till mötesplatsnoderna. Mötesplatsnoderna fungerar som en ryggrad vid distribueringen av information och samarbetar med varandra för att upprätthålla informationsinfrastrukturen. Det finns alltså ingen server i JXTA-nätverk som lagrar information om nätverket utan informationen distribueras mellan mötesplatserna i nätverket. Denna typ av nod har alltså andra uppgifter än att bara betjäna användaren. Detta kräver också mer resurser av noden. [SPEC02] [WI02]

Routernod (Router Peer)

En routernod har till uppgift att hjälpa andra noder som inte kan kommunicera direkt med varandra. Det kan till exempel vara noder som inte använder sig av samma transportprotokoll eller om det finns en brandvägg eller NAT som hindrar kommunikationen. En routernod lagrar routerinformation som andra noder i nätverket kan använda sig av när de ska skicka sina meddelanden. [WI02]

6.1.2 Grupp (Peer Group)

En grupp är ett antal noder som har ett eller flera gemensamma intressen. Med hjälp av grupper kan noderna i nätverket delas in i mindre grupper. Orsaken till detta kan till exempel vara av säkerhetsskäl, för att begränsa spridningen av information eller för att erbjuda en speciell tjänst. Alla noder i JXTA-nätverket är medlemmar i huvudgrupperna World Peer Group och Net Peer Group. World Peer Group konfigurerar och definierar de grundläggande egenskaperna hos noden. Net Peer Group är en gemensam grupp, för noderna i nätverket, som tillåter noderna att kommunicera med varandra. Grupperna i JXTA har en hierarkisk struktur där huvudgrupperna har den högsta rangen. [GR02] [WI02]

6.1.3 Ändpunkt (Endpoint)

Lägsta nivån i JXTA är ändpunkter. En ändpunkt är gränssnittet mot ett transportprotokoll som ansvarar för att transportera information över nätverket. En nod kan ha en eller flera ändpunkter, detta beror på att en nod samtidigt kan stödja flera transportprotokoll. [BRGOKR02]

6.1.4 Pipa (Pipe)

En pipa är en virtuell kommunikationskanal mellan två eller flera noder. Pipor används för att skicka information mellan applikationer eller tjänster. En pipa bildar en virtuell kommunikationskanal som döljer den underliggande strukturen. Detta gör att det ser ut som det finns en direkt koppling mellan två ändpunkter medan det i verkligheten kan vara flera noder och typer av nätverk inblandade. En pipa har två ändar, en för att skicka information och en för att ta emot information. Ändarna på piporna binds dynamiskt under exekvering med hjälp av protokollet Pipe Binding Protocol. [BRGOKR02]

6.1.5 Meddelande (Message)

Information som skickas mellan noder i nätverket paketeras i meddelanden. Ett meddelande kan innehålla ett godtyckligt antal element. Dessa element kan lagra alla typer av information. [SPEC01/02]

6.1.6 Annons (Advertisement)

Annonser används för att beskriva resurser i nätverket. Det är med hjälp av annonserna noderna kan upptäcka resurser i nätverket. Alla resurser i JXTA-nätverket måste associeras med en annons. Exempel på sådana resurser är noder, grupper av noder, pipor och tjänster. JXTA specifikationen innehåller följande annonser [SPEC01/02]:

- Peer Advertisement
- PeerGroup Advertisement
- ModuleClass Advertisement
- Module Specification Advertisement
- Module Implementation Advertisement
- Pipe Advertisement
- Rendezvous Advertisement

Alla annonser representeras med hjälp av XML, vilket gör annonserna programspråks- och plattformsoberoende.

6.1.7 Intyg (Credential)

Intyg är ett XML-dokument som används vid identifiering. Till exempel kan det bifogas i meddelanden för att identifiera sändaren och sändarens rättighet att skicka meddelanden. På detta sätt kan mottagaren av ett meddelande identifiera sändaren. Varje nod lagrar sina egna intyg. [SPEC01/02]

6.1.8 Identifierare (Identifier)

Ett P2P-nätverk innehåller ofta en mängd olika typer av resurser. Identifierare används för att unikt identifiera dessa resurser. [SPEC02]

6.1.9 Innehåll (Content)

Ett innehåll i JXTA-nätverket kan t.ex. vara ett textdokument, exekverande processer och programkod. Innehåll i nätverket publiceras och kan delas inom gruppen. JXTA har ingen begränsning när det gäller storleken på innehållet. [SPEC01/02]

6.1.10 Modul (Module)

Moduler är ett grundläggande koncept i JXTA och kan distribueras i nätverket. Funktionaliteten i JXTA är uppbyggd av moduler. Genom att skapa nya moduler kan funktionaliteten i JXTA utökas. Det finns två typer av moduler; tjänster och applikationer. För att noder ska kunna upptäcka rätt moduler delas definitionen av dem in i tre typer av annonser. Dessa annonser är Module Class Advertisement, Module Specification Advertisement och Module Implementation Advertisement. [WI02]

Module Class Advertisement används för att dela in moduler med liknande funktionalitet i samma klass. Module Specification Advertisement beskriver modulens beteende och protokoll på ett programspråks- och plattformsoberoende sätt. Module Implementation Advertisement är en implementation av specifikationen och är programspråksspecifik. Alla dessa annonser är nödvändiga eftersom att JXTA ska vara programspråks- och plattformsoberoende. [WI02]

6.2 Protokollen

JXTA definierar sex fristående protokoll som är designade för att kunna skapa ett decentraliserat, självorganiserat och självkonfigurerat distribuerat system. Genom att använda dessa protokoll kan noderna själva skapa grupper, som står för ordning och konfiguration, oberoende av nodernas position i nätverket. Dessutom kan noderna annonsera resurser, söka resurser och använda resurser. Med hjälp av protokollen kan noderna dynamiskt kommunicera över olika typer av nätverk och genom hinder i nätverket. De protokoll som ingår i JXTA är [SPEC01/02]:

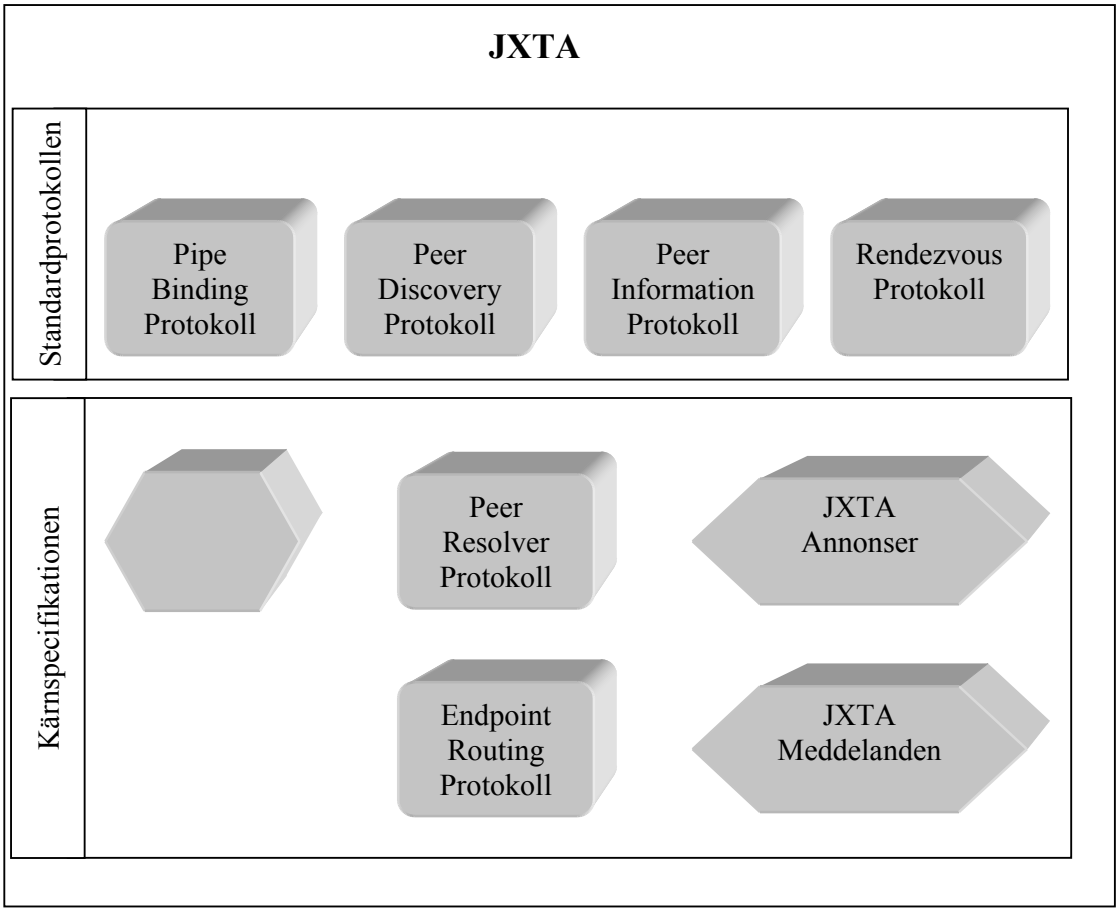
- Endpoint Routing Protocol (ERP) – Detta protokoll gör det möjligt för två noder som inte kan kommunicera direkt med varandra att göra detta med hjälp av andra noder i nätverket. Protokollet definierar en metod att hitta rutten mellan två noder som inte känner till varandra eller kan kommunicera direkt med varandra.
- Peer Resolver Protocol (PRP) – Med hjälp av detta protokoll kan en generell fråga – svar kommunikation upprättas mellan noder inom en grupp.
- Peer Discovery Protocol (PDP) – Detta protokoll används för att noderna ska kunna annonsera om sina egna resurser och hitta varandras resurser inom gruppen. Exempel på resurser är grupper, tjänster, pipor och andra noder.

- Rendezvous Protocol (RVP) – Med hjälp av detta protokoll kan noderna ansluta sig till mötesplatsnoder för att sprida meddelanden till alla ansluta noder. Förutom detta tillhandahåller protokollet möjligheten att annonsera en mötesplats.
- Peer Information Protocol (PIP) – Protokollet används för att få statusinformation, som t.ex. trafikbelastning och kapacitet, om andra noder inom gruppen.
- Pipe Binding Protocol (PBP) – Detta protokoll är en mekanism för att upprätta pipor mellan en eller flera noder inom gruppen.

6.3 Uppbyggnaden

Visioner som skaparna har haft vid designen av JXTA [JXTA02]:

- Använd redan befintliga och beprövade standarder som har fungerat tidigare.
- Engagera olika experter tidigt och ofta under utvecklingen.
- Uppmuntra till öppen utveckling av design, specifikation och kod.
- Behålla kärnan liten och elegant. Alltså gör en distinktion i arkitekturen mellan kärnmekanismer och mekanismer som är valfria.
- Skilja på innehållet i kärnan och valfria tjänster.
- Skapa en plattform för att befrämja kommunikationen mellan olika applikationer.
- Tänka på säkerheten under hela designen.

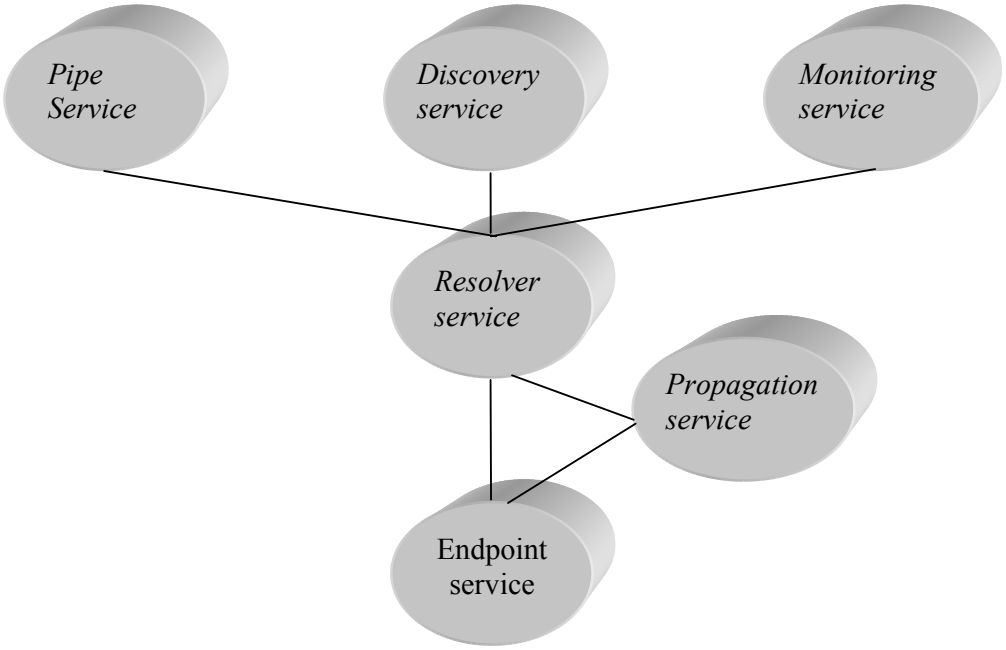


Figur 4 – Delarna i JXTA.

JXTA är en specifikation som består av sex protokoll och ett antal tjänster. Specifikationen kan delas in i två delar, kärnspecifikationen och standardprotokollen (Figur 4). Kärnspecifikationen innehåller de mest grundläggande delarna i JXTA. Dessa delar innefattar tjänster för grundläggande kommunikation mellan noderna. Standardprotokollen utökar kärnans funktionalitet med grundläggande funktioner för P2P-system. Alla protokoll som ingår i JXTA utom Endpoint Routing Protocol implementeras som tjänster och är knutna till gruppen. [SPEC01/02]

- Resolver Service implementerar Peer Resolver Protocol.
- Discovery Service implementerar Peer Discovery Protocol.
- Monitor Service implementerar Peer Information Protocol.
- Pipe Service implementerar Pipe Binding Protocol.
- Propagation Service implementerar Rendezvous Protocol.

Tjänsterna använder sig inbördes av varandra för att bygga upp plattformen JXTA. Förutom dessa tjänster ingår Endpoint Service i JXTA-plattformen. Endpoint Service har till uppgift att skicka information mellan två noder i nätverket. När noderna inte kan kommunicera direkt med varandra används Endpoint Routing Protocol för att hitta rutten mellan noderna. Tjänsternas förhållande till varandra visas i Figur 5.



Figur 5 – Förhållandet mellan tjänsterna i JXTA, där linjerna indikerar tjänsternas utnyttjande av varandra.

Tanken med tjänsterna och protokoll i JXTA är att de ska vara generella och bygga upp en grund för P2P-system. Denna grund kan efter behov byggas ut för att uppfylla de olika krav som P2P-applikationer ställer. Tjänsterna kommer att beskrivas närmare under respektive protokollrubrik.

Säkerheten i JXTA bygger på att det finns möjlighet att bifoga intyg, certifikat, publika nycklar, etc. i meddelanden som skickas mellan noderna. Förutom detta använder JXTA befintliga underliggande säkerhetstekniker.

6.4 Kärnspecifikationen

JXTA är designat för att vara ett litet system med endast ett fåtal obligatoriska egenskaper och komponenter. Kärnspecifikationen är ett krav vid implementationen av JXTA specifikationen. Delarna som ingår i kärnspecifikationen är: identifierare, meddelanden, annonser, Endpoint Router Protocol, Endpoint Service och Peer Resolver Protocol. [SPEC01/02]

6.4.1 JXTA-identifierare

Ett JXTA-nätverk innehåller en mängd olika typer av resurser. För att unikt kunna identifiera och referera resurser i nätverket används JXTA-identifierare. Protokollen i JXTA refererar till exempelvis noder, grupper och pipor. Uniform Resource Names (URN) används som identifierare i JXTA. För ytterligare detaljer se Appendix B. [SPEC01/02]

För att det inte ska uppstå några tvetydigheter mellan olika identifierare måste en JXTA-identifierare uppfylla vissa egenskaper. Identifierarna måste referera till unika resurser och för varje resurs får det endast finnas en identifierare. En annan egenskap identifierarna måste uppfylla är att det inte får finnas några tvetydigheter till vilken resurs de refererar. Sammanhanget ska generellt räcka för att veta vilken typ av identifierare det rör sig om. [SPEC01/02]

Det finns sex olika typer av standardidentifierare. Dessa standardtyper refererar till följande: noder, grupper, pipor, innehåll (content), modulklasser (module classes) och modulspecifikationer (module specifications). [SPEC01/02]

6.4.2 Meddelanden

Informationen som skickas mellan noderna paketeras i meddelanden. Varje meddelande består av ett godtyckligt antal element. Ett element är en namngivet data. Varje element kan i sin tur bestå av ett godtyckligt antal andra element. När ett meddelande lämnar en tjänst för att skickas till en annan nod adderas en eller flera tjänstspecifika element av tjänster på lägre nivå. När ett meddelande mottas av en nod subtraheras tjänstspecifika element av tjänster på lägre nivå än mottagartjänsten. [SPEC01/02]

Varje element innehåller följande attribut [SPEC01/02]:

- **Namnrymd** – Detta attribut är obligatoriskt. Genom att tilldela varje element en namnrymd kan element som tillhör olika meddelanden

organiseras i samma meddelande. Det finns två fördefinierade namnrymder och det är ”” och ”jxta”. Namnrymden ”jxta” är reserverad för användning inom tjänsterna och protokollen i JXTA. Namnrymden ”” används av applikationer och tjänster utanför JXTA. Det finns möjlighet att skapa nya namnrymder. Nya namnrymder behöver inte registreras utan det räcker att de är kända för de inblandade parterna.

- **Namn** – Attributet definierar namnet på elementet. Flera element kan ha samma namn i ett meddelande.
- **Typ** – Specificerar MIME-typen. Används av applikationer och tjänster för att hantera innehållet. Om inget anges antas typen var ”application/octet-stream”.
- **Innehåll** – Detta attribut innehåller elementets data. Data i elementen är inte synligt för de tjänster och applikationer som inte berörs av det.

6.4.3 Annonser

Annonser används för att beskriva resurser i nätverket. Dessa resurser kan vara noder, grupper, pipor, tjänster och andra typer av resurser. Tanken med annonserna är att de ska vara generella och kunna beskriva alla typer av resurser. Annonserna representeras i XML vilket är ett kraftfullt och plattformsoberoende sätt att representera data [SPEC01/02]. Enligt Brendon J. Wilson definieras annonser på följande sätt [WI02]:

> *”A structured representation of an entity, service or resource made available by a peer or peer group as a part of a P2P network.”*

Noderna utbyter information om vad som är tillgängligt i nätverket genom att använda annonser. Noderna kan på så sätt upptäcka de resurser som finns tillgängliga i nätverket. I kärnspecifikationen finns följande annonser; Peer Advertisement, Peer Group Advertisement, Module Class Advertisement, Module Specification Advertisement och Module Implementation Advertisement. För ytterligare detaljer se Appendix C. [SPEC01/02]

Peer Advertisement

Peer Advertisement är en annons som beskriver en nod. Annonsen innehåller alltså specifik information om noden som identifierare, namn och beskrivande information och nodens ändpunktadresser. [SPEC01/02]

Peer Group Advertisement

Peer Group Advertisement är en annons som beskriver en grupp av noder. Annonsen innehåller information om gruppens namn, identifierare, beskrivande information om gruppen och vilka tjänster som gruppen kan erbjuda sina medlemmar. [SPEC01/02]

Module Class Advertisement

En modul i JXTA-nätverket beskrivs av tre annonser. Module Class Advertisement är den mest generella annonsen av dem tre. Denna annons används för att dela in moduler med liknande funktionalitet i samma klass. Annonsen innehåller inte någon information om hur modulen är implementerad. För att unikt identifiera modulklassen innehåller annonsen en identifierare. Utöver identifieraren innehåller annonsen ett modulklassnamn och en generell beskrivning av modulklassen. Informationen i denna annons kan användas av andra noder för att hitta rätt modul. [SPEC01/02]

Module Specification Advertisement

Module Specification Advertisement beskriver modulen på ett mer specifikt sätt än vad Module Class Advertisement gör. Det kan finnas flera annonser av denna typ för varje Module Class Advertisment. Denna annons används för att specificera moduler på ett programspråks- och plattformsoberoende sätt. Annonsen har två huvudsyften. Det första är att tillhandahålla dokumentation om hur modulen ska implementeras. Detta för att var och en ska kunna göra en egen implementation av modulen. Det andra huvudsyftet är att tillhandahålla information om hur kommunikation kan ske med de noder som har implementerat specifikationen. Det finns två element i annonsen för detta ändamål. Det första elementet innehåller en Pipe Advertisement som kan användas för att skapa en pipa till den eller de implementerade modulerna. Om inte kommunikation kan ske direkt med den eller de implementerade modulerna kan det andra elementet användas. Detta element innehåller en modulspecifikation som beskriver en modul som kan användas som mellanhand vid kommunikationen. Det bör påpekas att alla element i Module Specification Advertisement är valfria utom annonsens identifierare och version. [SPEC01/02]

Module Implementation Advertisement

Module Implementation Advertisement är en implementation av en Module Specification Advertisement. Implementationen är programspråkspecifik. Det kan finnas flera implementationer av en modulspecifikation. Därigenom är inte specifikationen bunden till en specifik miljö. Module Implementation Advertisement innehåller all nödvändig information för att hitta, hämta och exekvera modulen. [SPEC01/02]

6.4.4 Endpoint Routing Protocol

Endpoint Routing Protocol (ERP) är ett protokoll som definierar en annons och två meddelanden som används för att hitta rutten mellan två noder som inte kan kommunicera direkt eller inte känner till varandra. Detta kan bero på att noderna inte har ett gemensamt transportprotokoll, befinner sig i olika nätverk eller att det finns hinder i nätverket. [SPEC01/02]

Annons

Protokollet definierar en annons för att beskriva rutten mellan två noder i nätverket. Dessa annonser lagras av de noder i nätverket som har till uppgift att vara routrar. För ytterligare detaljer se Appendix D.

Meddelande

Protokollet består av två typer av meddelanden. Den första typen av meddelande används för att göra förfrågningar till routernoderna om rutten mellan noder. Den andra typen av meddelande använder sig routernoderna av för att svara på förfrågningarna. För ytterligare detaljer se Appendix D. [SPEC01/02]

När en nod ska skicka ett meddelande till en annan nod, som den inte kan skapa en direkt förbindelse med, måste den försöka hitta andra noder som kan vidarebefordra meddelandet till slutdestinationen. Detta gör den genom att skicka en förfrågan till kända routernoder. Om routernoden känner till en rutt mellan startpunkten och slutdestinationen svarar den på förfrågan. De som inte kan svara på förfrågan kan i sin tur skicka den vidare till andra kända routernoder. Svaret på förfrågan innehåller annonsen som beskriver rutten till slutdestinationen. [SPEC01/02]

6.4.5 Ändpunkttjänsten (Endpoint Service)

Ändpunkttjänsten är nodens accesspunkt till de transportprotokoll i JXTA som noden stödjer. Tjänsten är ansvarig för att skicka och ta emot meddelanden mellan två ändpunkter genom att använda ett av de underliggande transportprotokollen. Exempel på transportprotokoll i JXTA är TCP, HTTP, TLS och BEEP. De övriga tjänsterna i noden använder sig av ändpunkttjänsten direkt eller indirekt, för att kommunicera med andra noder och tjänster. Olikt alla andra tjänster i JXTA är inte ändpunkttjänsten begränsad till en grupp, utan alla noder i JXTA-nätverket delar samma ändpunkttjänst. Vilket transportprotokoll som ska användas vid skickandet av ett meddelande bestäms av ändpunktadressen till slutdestinationen. Ändpunktadressen har följande utseende [WI02]:

```
Protocol://address_as_per_protocol/unique_name_of_recipient/
unique_name_inrecipient_context
```

När ett meddelande ska skickas har ändpunkttjänsten till uppgift att tolka ändpunktadressen och överlämna meddelandet till rätt transportprotokoll. Ändpunktadressen kan både vara transportspecifik och transportneutral. Den transportspecifika adressen anger vilket transportprotokoll i JXTA som ska användas när ett meddelande ska skickas. Vid användandet av den transportspecifika adressen måste noderna som kommunicerar ha en direkt förbindelse med varandra via ett gemensamt transportprotokoll. [WI02]

Den transportneutrala adressen, på formen ”jxta://<identifierare>”, aktiverar Endpoint Router Transport Protocol (ERTP) som också är ett transportprotokoll. Detta transportprotokoll har till uppgift att göra de andra transportprotokollen transparenta för andra tjänsterna i noden. Detta görs i första hand genom att försöka hitta ett gemensamt transportprotokoll som kan sammanbinda noderna. Om detta inte är möjligt används Endpoint Routing Protocol för att försöka hitta en möjlig rutt mellan noderna, denna rutt kan därefter bifogas i det meddelande som ska skickas. [WI02]

6.4.6 Peer Resolver Protocol

Med hjälp av Peer Resolver Protocol (PRP) kan en generell fråga – svar kommunikation upprättas mellan noder inom en grupp. Peer Resolver Protocol implementeras av Resolver Service. Denna tjänst används av andra tjänster på högre nivå för att administrera frågorna, t.ex. Discovery Service (Figur 6). Resolver Service skickar frågorna till namngivna hanterare i andra noder inom gruppen. De mottagande nodernas Resolver Service har då till uppgift att överlämna frågan till rätt hanterare. Om noden har en hanterare med samma namn, processar hanteraren frågan för att förhoppningsvis kunna svara på frågan. Om så är fallet överlämnas svaret till Resolver Service som har till uppgift att skicka tillbaka svaret. [SPEC01/02]

Resolver Service använder Propagation Service för att sprida frågorna till flera mottagare. Om frågan ska skickas till en specifik nod används Endpoint Service. [SPEC01/02]

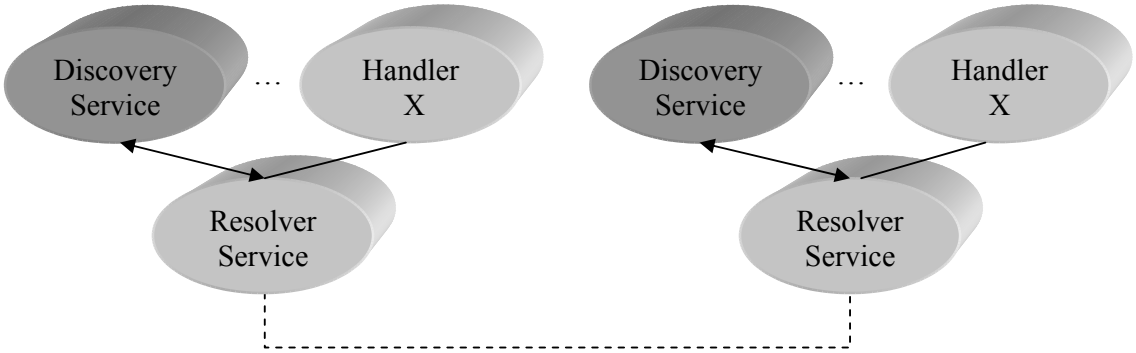
Meddelanden

Peer Resolver Protocol består av ett frågemeddelande och ett svarsmeddelande. Frågemeddelandet används för att skicka frågor till namngivna hanterare i andra noder inom gruppen. Svarsmeddelandet används för att svara på frågemeddelandet. Frågemeddelandet innehåller information om vilken nod som skickar meddelandet, vilken hanterare som ska hantera frågan, en unik frågeidentifierare och ett eventuellt element som auktoriserar sändaren att ställa frågor inom gruppen. Den unika frågeidentifieraren används även i svarsmeddelandet, för att kunna matcha frågemeddelandet med rätt svarsmeddelande. För ytterligare detaljer se Appendix E. [SPEC01/02]

Hanterare

En hanterare är en modul som har till uppgift att behandla meddelanden. Exempel på en hanterare är Discovery Service, som har till uppgift att behandla meddelanden som är specifikt ägnat åt tjänsten. Det är namnet på hanteraren som bestämmer hur meddelandet skall behandlas. Varje tjänst generar ett hanterarnamn som är unikt för noden. Om flera noder har samma

tjänst är dessa hanterarnamn identiska. Konventionen för att skapa hanterarnamn är att lägga ihop namnet på tjänsten, gruppens identifierare och eventuella parametrar med varandra. Hanterarnamnet registreras i Resolver Service tillsammans med tillhörande hanterare. På detta sätt kan frågemeddelanden distribueras till rätt hanterare. [WI02]



Figur 6 – Ett exempel där Discovery Service är registrerad som hanterare i Resovler Service och kan därigenom använda tjänsten.

6.5 Standardprotokollen

Denna del av specifikationen utökar kärnans funktionalitet med fyra grundläggande tjänster och protokoll.

6.5.1 Peer Discovery Protocol

Resurser publiceras med hjälp av annonser i JXTA. För att hitta dessa annonser används tjänsten Discovery Service. Denna tjänst implementerar protokollet Peer Discovery Protocol (PDP). PDP definierar ett frågemeddelande och ett svarsmeddelande. Dessa meddelanden skickas till andra noder inom gruppen med hjälp av Resolver Service. Annonser hittas genom sökning, först lokalt bland resultat från tidigare sökningar. Om denna sökning inte ger upphov till något resultat får noden ta hjälp av andra noder i gruppen för att försöka hitta det den söker. Det är valfritt för de noder som får en fråga att svara på frågan. Detta innebär att det inte är någon garanti att noden som frågar får ett svar. Medan det i andra fall kan ge upphov till ett flertal svar och vissa av dessa kan innehålla redundant information. [WI02]

Meddelande

Peer Discovery Protocol definierar två meddelanden, ett frågemeddelande och ett svarsmeddelande. Frågemeddelandet används för att söka efter annonser och svarsmeddelandet används för att svara på frågan. När en fråga skapas anges bland annat vilken typ av resurs som söks och hur många svar

som önskas. Dessutom finns det möjlighet att söka efter attribut med speciella värden. Svarsmeddelandet innehåller de annonser som efterfrågats. För ytterligare detaljer se Appendix F. [SPEC01/02]

6.5.2 Rendezvous Protocol

Propagation Service är en implementation av Rendezvous Protocol. Propagation Service används för att kunna sprida meddelanden till noder inom gruppen oberoende av underliggande nätverksteknologier. Detta görs genom att noder kan ansluta till mötesplatsnoder och genom dessa skicka och ta emot meddelanden som ska spridas till noderna inom gruppen. Mötesplatsnoderna kan också använda sig av varandra för att öka räckvidden på de meddelanden som ska spridas. Propagation Service har även till uppgift att kontrollera spridningen av meddelanden, detta för att de inte ska fortsätta att spridas i all oändlighet. Kontrollen uppnås med hjälp av teknikerna loopback-detection, Time-To-Live och duplicate-detection. Alltså ett meddelande skickas inte vidare om det har processats tidigare, om livstiden på meddelandet har gått ut eller om meddelandet har mottagits tidigare. [SPEC01/02]

Annons

En nod kan konfigureras till att vara en mötesplatsnod eller dynamiskt tilldelas mötesplatsuppgifterna. När en nod blir mötesplatsnod sker det genom att noden annonserar om sig själv. I annonsen ingår gruppens och nodens identifierare. Förutom dessa unika identifierare ingår eventuellt ett valfritt namn på mötesplatsen. För ytterligare detaljer se Appendix G. [SPEC01/02]

Meddelande

Protokollet definierar inga meddelanden utan bara element som kan adderas till valfria meddelanden för att leasa uppkopplingar av mötesplatsnoder. Noder ansluter till mötesplatsnoder genom att skicka ett meddelande till mötesplatsnoden om att få leasa en uppkoppling. Detta meddelande ska innehålla elementet ”jxta:Connect”, vilket i sin tur ska innehålla nodens annons. Mötesplatsnoden kan då välja att svara genom att skicka tillbaka ett meddelande till noden. Detta meddelande ska innehålla tre element, ”jxta:ConnectedLease”, ”jxta:ConnectedPeer” och ”jxta:RdvAdvReplay”. Dessa element definierar tiden för hur länge leasingen gäller i millisekunder, identifieraren som identifierar mötesplatsnoden och annonsen som beskriver mötesplatsnoden. Om noden vill avbryta leasingen i förtid skickar den ett meddelande till mötesplatsnoden för att meddela detta. Detta meddelande ska innehålla elementet ”jxta:Disconnect”, vilket i sin tur ska innehålla nodens annons. [SPEC01/02]

6.5.3 Peer Information Protocol

Information Service är en implementation av protokollet Peer Information Protocol och har till uppgift att ge övervakningsinformation om andra noder inom gruppen. Exempel på sådan information kan vara nätverks- och processorbelastning. Tjänsten kan användas för till exempel lastbalansering mellan noder med samma tjänst. [WI02]

Meddelande

Peer Information Protocol definierar ett frågemeddelande och ett svarsmeddelande. Både frågemeddelandet och svarsmeddelandet innehåller information om vilken nod som ställde frågan och vilken nod som ska svara på frågan. Frågemeddelandet innehåller dessutom en specifikation om vilken information noden är intresserad av. Svarsmeddelandet innehåller förhoppningsvis svaret på frågan. För ytterligare detaljer se Appendix H. [SPEC01/02]

6.5.4 Pipe Binding Protocol

En pipa är en virtuell kommunikationskanal mellan en eller flera ändpunkter. Pipor används av tjänster och applikationer för att kommunicera med andra noder. En pipa består av två ändar. Den ena änden används för att skicka information och den andra änden används för att ta emot skickad information. Piporna är asynkrona vilket innebär att noderna agerar oberoende av varandra utan någon synkroniseringsmekanism. Specifikationen definierar tre typer av pipor. Den första används för kommunikation mellan två noder och erbjuder ingen säkerhet eller tillförlitlighet. Den andra används också för kommunikation mellan två noder, men erbjuder en säker och tillförlitlig kommunikation. Den tredje typen används för att skicka information från en nod till flera andra noder. När kommunikation ska ske mellan två noder krävs det att den som ska skicka information genom pipan vet att det är någon som är redo att ta emot informationen. För att lösa detta används Pipe Binding Protocol. Pipe Service är en implementation av Pipe Binding Protocol. Pipe Service har till uppgift att upptäcka, skapa och ta bort pipor. [WI02]

Annons

Pipe Advertisement är en annons som används för att annonsera om pipor. Annonsen innehåller en unik identifierare för att unikt kunna identifiera pipan, pipans namn och pipans typ. För ytterligare detaljer se Appendix I. [SPEC01/02]

Meddelande

Pipe Binding protocol definierar ett meddelande som används för att binda en pipa. Meddelandet används både för att fråga andra noder inom gruppen om någon är redo att ta emot information genom pipan och som svarsmeddelande på frågan. För ytterligare detaljer se Appendix I. [SPEC01/02]

7 Referensteknologier

De teknologier som presenteras under detta avsnitt används som representanter för icke P2P-teknologier vid prestandatesterna.

7.1 BEEP – Blocks Extensible Exchange Protocol

Marshall Rose och Carl Malamud började 1998 designen av ett nytt Internetprotokoll som fick namnet Block Extensible Exchange Protocol (BEEP). Resultatet presenterades sedan för Internet Engineering Task Force (IETF) vilket resulterade i standarden RFC 3080. BEEP är ett ramverk som är tänkt att användas vid skapandet av nya Internetprotokoll och löser följande problem.

- Separera meddelanden från varandra.
- Konvertera meddelanden till rätt format.
- Flera kommunikationskanaler över en kommunikationslänk.
- Felrapportering.
- Kryptering.
- Autentisering.

På detta sätt behöver inte utvecklarna lösa de vanligaste uppkommande problemen utan kan koncentrera sig på den unika funktionaliteten i det nya Internetprotokollet. [BEEP01]

När två parter ska kommunicera med varandra, med BEEP-teknologin, måste den ena parten initiera kommunikationen medan den andra parten måste vänta på att kommunikationen ska initialiseras. Efter att en session har skapats kan parterna kommunicera fritt med varandra. Kommunikationen kan ske så länge sessionen inte bryts. Parterna kommunicerar med hjälp av en eller flera kommunikationskanaler som skapas dynamiskt efter att sessionen har skapats. Varje kommunikationskanal har en profil som beskriver syntaxen och semantiken för de meddelanden som skickas genom kanalen. Varje part annonserar de profiler de stödjer vid initieringen av sessionen. [BEEP01]

Application Exchange (APEX) är en profil i BEEP som tillhandahåller en asynkron meddelandetjänst för applikationer (ändpunkter). Tjänsten byggs upp av ett antal noder som sammankopplas till ett APEX-nätverk. Det finns två typer av noder i ett APEX-nätverk. Den första typen är ändpunkter.

Ändpunkterna fungerar som start- och slutdestination för meddelanden. Den andra typen av noder fungerar som vidarebefordrare av meddelanden. Det är med hjälp av dessa noder meddelanden skickas från en ändpunkt till en annan ändpunkt. [BEEP01]

7.1.1 Jämförelse med JXTA

BEEP är ett ramverk för skapandet av nya Internetprotokoll. Ramverkets huvudsakliga uppgift är att definiera hur kommunikationen mellan två parter ska ske. Vid jämförelsen mellan BEEP och JXTA finns likheter. En likhet är att de båda definierar ett generellt sätt för två parter att kommunicera. En annan likhet är att parterna i kommunikationen kan vara jämbördiga. Ingen klient-serverarkitektur behöver alltså följas. Men det finns stora skillnader mellan JXTA och BEEP. JXTA definierar inte bara hur kommunikation kan ske mellan två parter utan bygger upp en modell för hur noder i ett nätverk kan kommunicera, använda varandras tjänster, grupperas, övervakas och dela information. I JXTA finns det också möjlighet att använda ramverket BEEP som transportprotokoll vid kommunikation.

Vid jämförelsen mellan APEX och JXTA finns ytterligare några likheter. APEX är liksom JXTA ett sätt att sammankoppla ett flertal noder till ett virtuellt nätverk, där noder dynamiskt kan ansluta och skicka meddelanden till andra noder genom att enbart känna till deras identitet.

7.2 Socket

Socket är en ändpunkt vid kommunikation över nätverket. Kommunikationen kan ske med hjälp av protokollen TCP eller UDP. När socket omnämns i denna rapport sker kommunikationen med hjälp av TCP-protokollet. Detta eftersom att all implementation sker i Java och där är klassen ”Socket” knytet till TCP-protokollet. [SOCKET]

7.2.1 Jämförelse med JXTA

Socket är ett rent gränssnitt mot TCP eller UDP, som är transportprotokoll. JXTA är en generell P2P-plattform, där en av de transportprotokoll som kan användas är TCP.

8 Tester

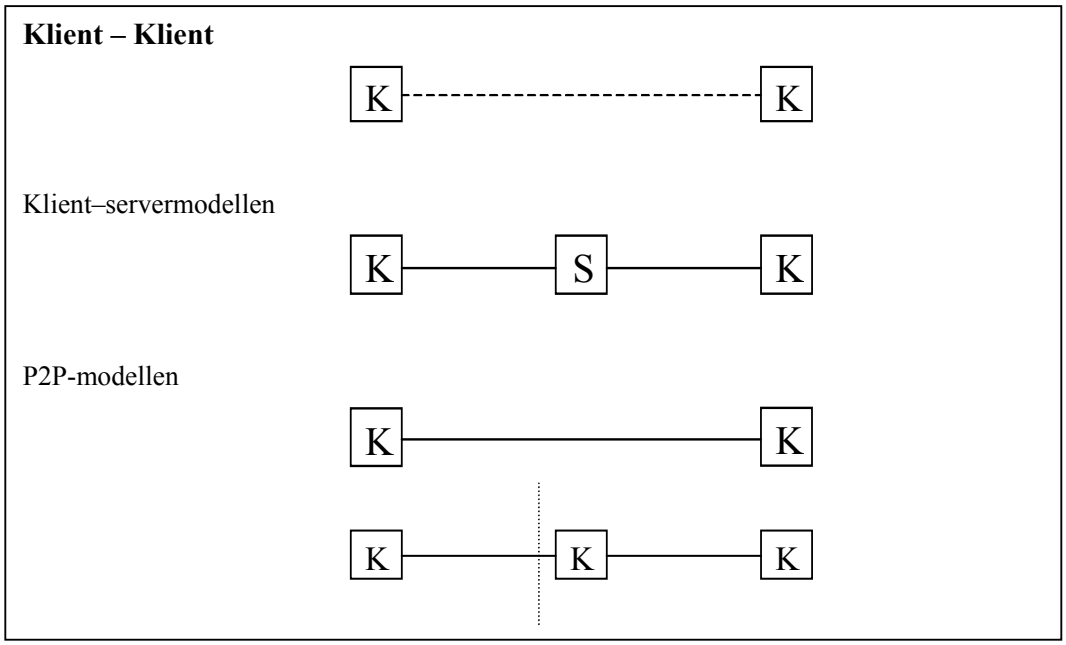
Testningen har delats upp i två delar. Den första delen är en prestandajämförelse mellan den traditionella klient-servermodellen och P2P-modellen. För att kunna göra denna jämförelse har vi använt oss av JXTA-teknologin och två referensteknologier. Valet av referensteknologier har fallit på socket-kommunikation och BEEP. Vid testtillfällena har prestanda i informationsöverföring mätts.

Den andra delen har inriktats på att praktiskt testa hur JXTA-teknologin fungerar i praktiken. Detta har gjorts genom en implementation av ett informationsspridningssystem baserat på JXTA-teknologin.

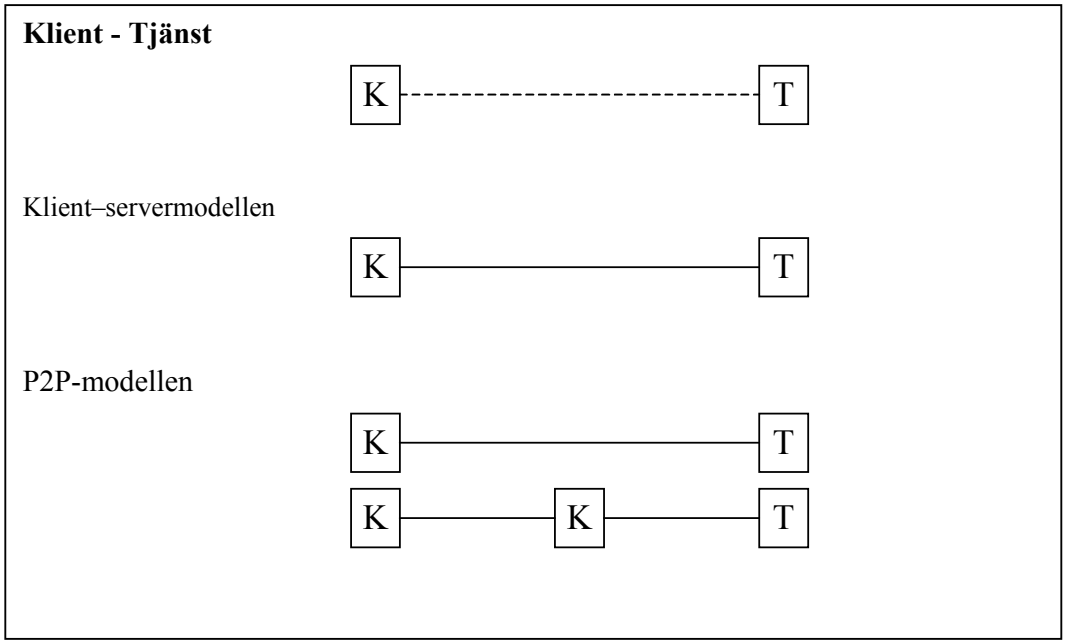
8.1 Prestandatester

Prestandatesterna har gjorts med avseende på prestanda i informationsöverföring. För denna prestandatestning har kommunikationsteknologierna JXTA, BEEP och socket-kommunikation använts. JXTA har valts som representant för P2P-modellen. BEEP har valts för att det, liksom JXTA, är en generell kommunikationsplattform. Den stora skillnaden mellan dessa två plattformar är att BEEP främst är tänkt för direkt kommunikation mellan två parter medan JXTA är en generell P2P-plattform. Socket-kommunikation har valts eftersom den traditionellt använts mycket i klient-serverapplikationer. Detta eftersom socket-kommunikation är gränssnitt mot TCP/IP, som är det nätverksprotokoll som används mest i dagens nätverk. Dessutom kan både BEEP och JXTA köras över TCP/IP. Detta tydliggör eventuella vinster och/eller förluster i prestanda hos dessa två teknologier.

Ett stort problem har varit att hitta rättvisa tester där endast kommunikationsmodellen varierar. Därför har testet delas upp i två testscenarion. I det första scenariot utbyter två klienter information med varandra (Figur 7). I det andra utbyter klient och tjänst information med varandra (Figur 8). För att kunna testa skillnader mellan P2P-modellen och den traditionella klient-servermodellen har två testuppställningar använts. Den första bygger på direkt kommunikation mellan två datorer. Det andra bygger på indirekt kommunikation mellan två datorer, via en tredje dator. Mer information om dessa testuppställningar beskrivs senare i detta avsnitt. Förutom jämförelsen av modellerna kommer prestandaskillnaderna i de olika teknologierna att kunna utvärderas.



Figur 7 – Scenario 1: Utbyte av information mellan två klienter.

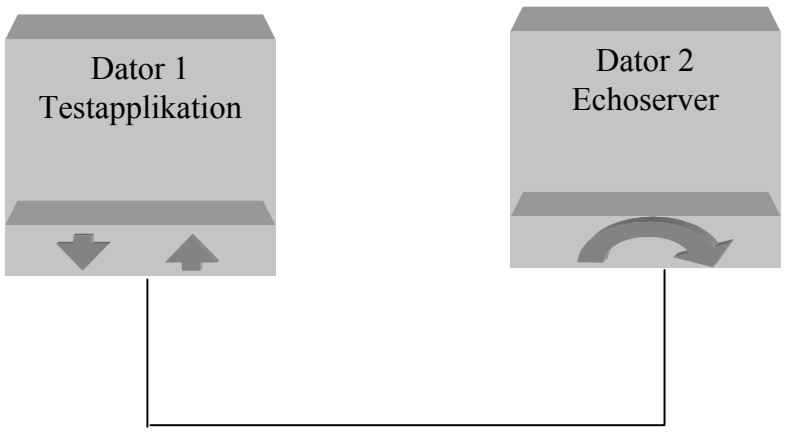


Figur 8 – Scenario 2: Utbyte av information mellan klient och tjänst.

Det som mäts är tiden från att ett meddelande skickas och tills det kommer tillbaka. För att utföra testningen har tre applikationer implementerats. Den första applikationen används för att mäta tiden från det att den skickar meddelandet till dess att meddelandet kommit tillbaka och att lagra tiderna, denna applikation kallas Testapplikationen. Den andra applikationen har till uppgift att ta emot meddelanden och skicka tillbaka dem, kallad Echoserver. Den tredje applikationens uppgift är att vidarebefordra meddelanden, detta både från Echoservern till Testapplikationen och tvärtom.

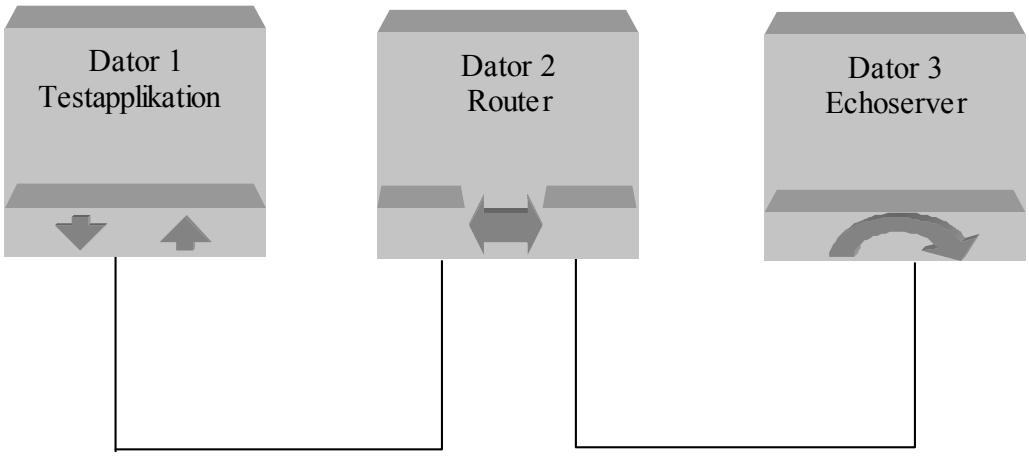
I de två första applikationerna har funktionalitet aktivt programmerats upp för alla tre teknologier. Medan i den tredje har bara funktionaliteten hos socket-kommunikationen och BEEP aktivt programmerats upp. Vid indirekt kommunikation finns en skillnad mellan socket-kommunikation och BEEP. BEEP hanterar informationen som ett paket vilket innebär att BEEP tar emot all information innan den skickas vidare. Socket-kommunikation skickar däremot vidare delar av informationen så fort den är mottagen. Hur JXTA gör med denna detalj är upp till implementatören av plattformen. Ingen aktiv programmering har krävts av JXTA eftersom plattformen har funktionaliteten inbyggd och således har plattformen bara startats. All implementation har skett i programspråket Java.

Testningen har genomförts med två testuppställningar. I den första har Testapplikationen körts direkt mot Echoservern (Figur 9). Dessa applikationer har exekverats på varsin dator.



Figur 9 – Testuppställning: Direkt kommunikation.

I den andra testuppställningen har alla tre applikationerna använts (Figur 10). Applikationerna har exekverats på var sin dator. Testapplikationen och Echoservern har inte kunnat kommunicera med varandra direkt. För mer information om testuppställningarna, se Appendix J.



Figur 10 – Testuppställning: Indirekt kommunikation.

Vid testtillfällena har meddelandestorleken varierats, detta för att se om denna parameter har betydelse. Dessutom har meddelandet skickats upprepade gånger för att ge ett rättvist resultat. Dessa parametrar har varierats enligt Tabell 1. När meddelandestorleken var som störst minskades antalet upprepningar, detta på grund av den ökade tidsåtgången.

Meddelandestorlek	**Antalet upprepningar**
1 B	1000
1 KB	1000
10 KB	1000
100 KB	1000
1 MB	1000
10 MB	100

Tabell 1 – Tabellen visar de meddelandestorlekar och antalet gånger meddelandet har skickats vid testerna.

Under testningen har teknologierna varvats, dvs. en teknologi åt gången har skickat ett meddelande och väntat på svar och därefter har det varit nästa teknologis tur osv.

8.2 Praktisk testning av JXTA

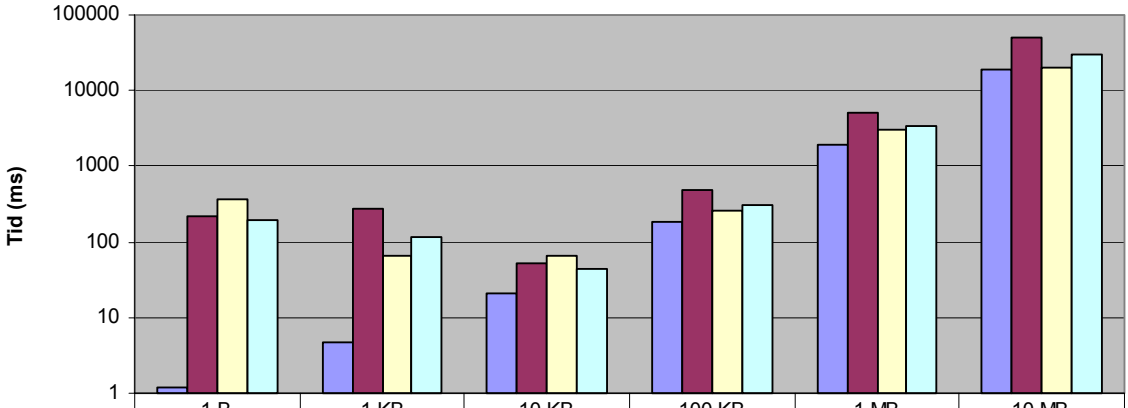
Den praktiska testningen av JXTA-plattformen har gjorts genom en implementation av ett informationsspridningssystem tänkt för skarp användning av utvecklarna på TietoEnator DevCon. Kraven på detta system finns specificerade i Appendix K. Tanken med detta test var att kontrollera om JXTA fungerade som P2P-plattform vid utveckling av applikationer och system.

9 Resultat

Detta avsnitt innehåller resultat ifrån de tester som utförts.

9.1 Prestandatester

Resultaten från testerna kommer att presenteras i diagramform med tillhörande värden, där medeltiderna för varje teknologi och meddelandestorlek kommer att redovisas för varje kommunikationsmetod. Dessutom kommer en medeltid för teknologierna att redovisas för varje meddelandestorlek och kommunikationsmetod. Detta för att en rättvis jämförelse av de två kommunikationsmetoderna ska kunna göras. För att kunna visa på prestandaförluster kommer ett diagram som presenterar skillnaden i procent mellan kommunikationsmetoderna också att redovisas. För att jämföra teknologiernas infrastruktur kommer teknologiernas överföringshastighet i bytes per tidsenhet att redovisas för de två kommunikationsmetoderna.



	1 B	1 KB	10 KB	100 KB	1 MB	10 MB
Socket	1	5	20	189	1904	19299
BEEP	213	270	51	473	5091	51736
JXTA	362	65	64	254	3011	20087
Medel	192	113	45	305	3335	30374

Diagram 1 – Testresultat vid direkt kommunikation.

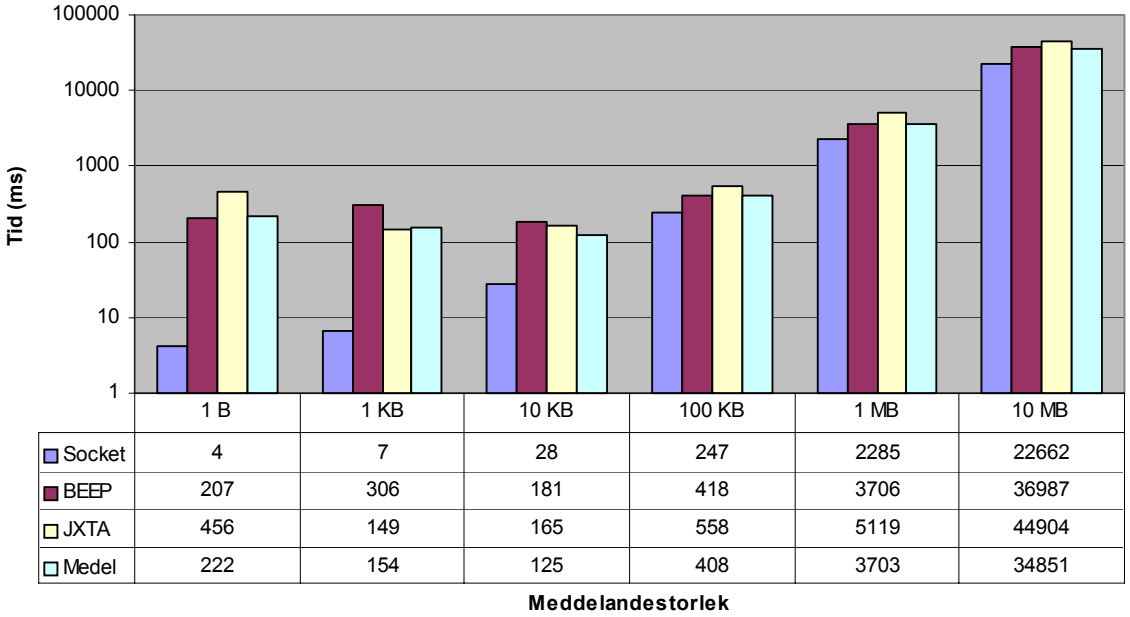


Diagram 2 – Testresultat vid indirekt kommunikation.

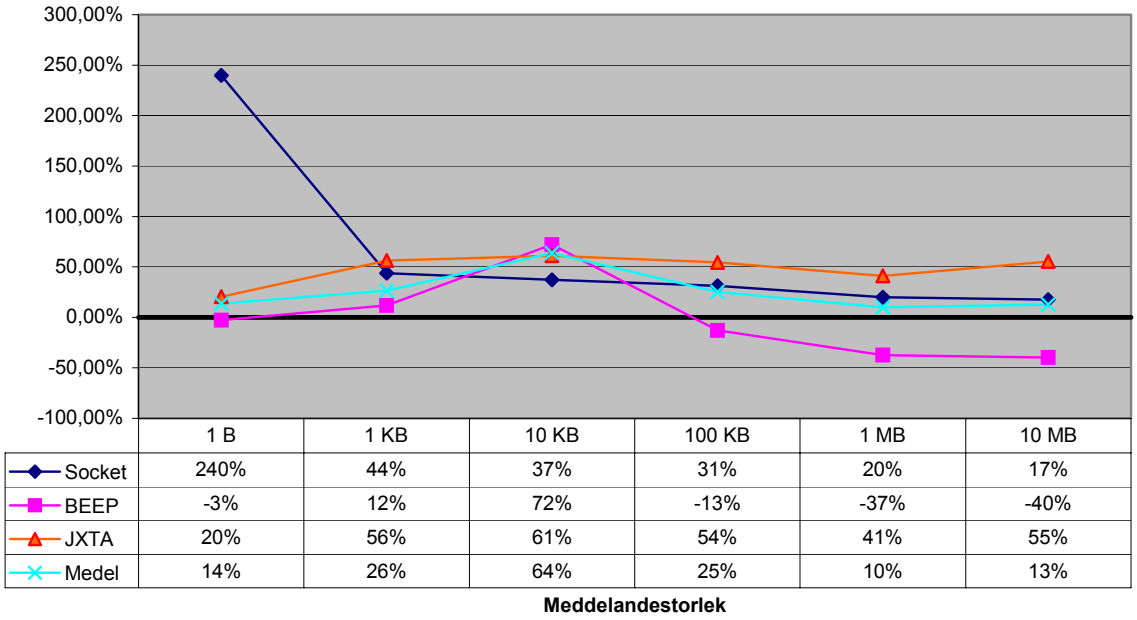
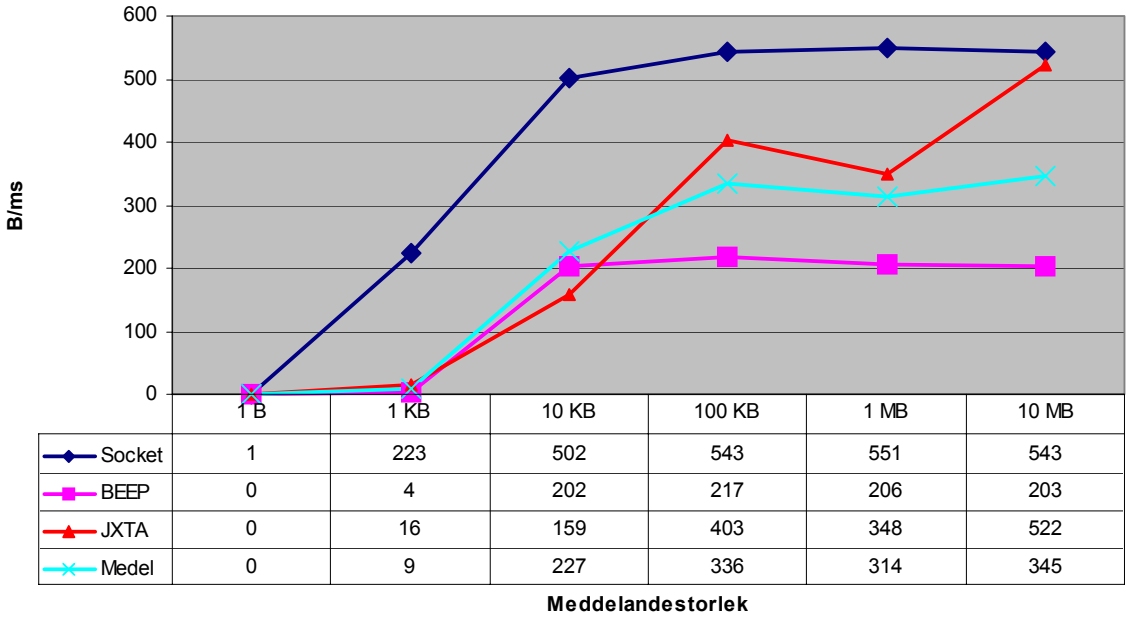


Diagram 3 - Prestandaförlusten vid indirekt kommunikation jämfört med direkt kommunikation.

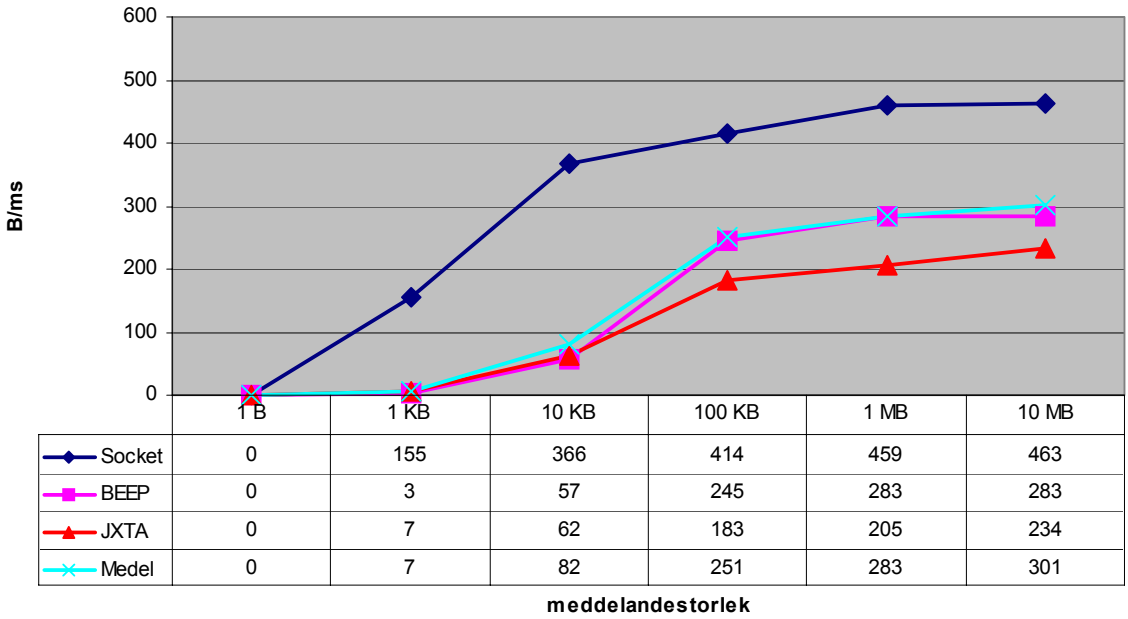
9.1.1 Kommunikationsmetoderna

Medeltiden av teknologierna visar på en prestandaförlust vid indirekt kommunikation jämfört med direkt kommunikation och att förlusten varierar relativt mycket beroende på meddelandestorleken (Diagram 1, Diagram 2 och Diagram 3). Prestandaförlusten varierar från 10 procent till 64 procent beroende på meddelandestorleken. Förlusten är som störst när meddelandestorlek är 10 KB. Vid övriga meddelandestorlekar är förlusten betydligt lägre och lägst är förlusten när meddelandet har storlek 1 MB.



	1 B	1 KB	10 KB	100 KB	1 MB	10 MB
Socket	1	223	502	543	551	543
BEEP	0	4	202	217	206	203
JXTA	0	16	159	403	348	522
Medel	0	9	227	336	314	345

Diagram 4 – Bytes per tidsenhet vid direkt kommunikation.



	1 B	1 KB	10 KB	100 KB	1 MB	10 MB
Socket	0	155	366	414	459	463
BEEP	0	3	57	245	283	283
JXTA	0	7	62	183	205	234
Medel	0	7	82	251	283	301

Diagram 5 – Bytes per tidsenhet vid indirekt kommunikation.

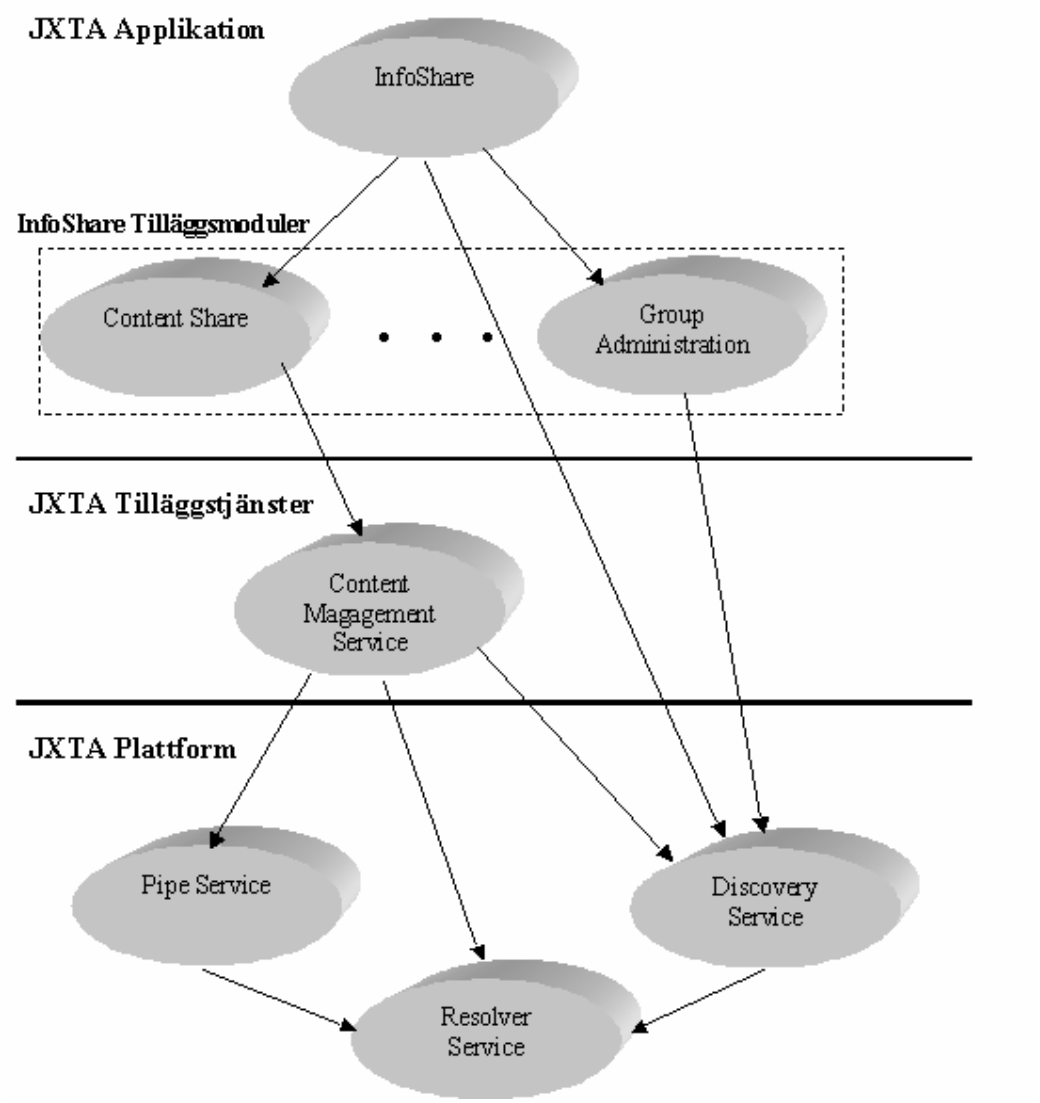
9.1.2 Teknologierna

Vid små meddelandestorlekar har teknologiernas uppbyggnad större betydelse än vid större meddelandestorlekar där överföringshastigheten har större betydelse (Diagram 4 och Diagram 5). Vid jämförelsen mellan referensteknologin, socket-kommunikation, och de övriga teknologierna finns en relativ stor prestandaförlust för BEEP och JXTA i överföringshastighet både vid direkt kommunikation och indirekt kommunikation. Denna förlust är som störst vid små meddelandestorlekar och minskar därefter när meddelandestorleken ökar. Det bör påpekas att vid direkt kommunikation är skillnaden mellan JXTA och referensteknologin vid meddelandestorleken 10 MB marginella, endast 4 procent. Men vid jämförelsen mellan enbart BEEP och JXTA är JXTA generellt snabbare vid direkt kommunikation och BEEP är generellt snabbare vid indirekt kommunikation. Ett annat resultat som bör påpekas är att teknologin BEEP har en högre överföringshastighet vid indirekt kommunikation än vid direkt kommunikation när meddelandestorleken är 100 KB och större.

9.2 Praktisk testning av JXTA

Det praktiska testet av JXTA resulterade i ett informationsspridningssystem. Under utvecklingen av informationsspridningssystemet har Java och JXTA:s referensimplementation i Java använts. Systemet uppfyller sin roll som informationsspridare men på grund av tidsbrist har endast kravet på fildelning implementerats. Detta visar att JXTA och i synnerhet referensimplementationen i Java uppfyller sin funktion som generell P2P-plattform.

Informationsspridningssystemet har byggts upp i form av en applikation som nyttjar JXTA-plattformen och den tilläggstjänst till plattformen som utvecklats (Figur 11).



Figur 11 – Systemöversikt över informationsspridningssystemet.

Informationsspridningssystemet består av en JXTA-applikation, InfoShare, och ett antal tilläggstjänster. Tanken med InfoShare är att den ska fungera som en skalapplikation som dynamiskt ska kunna ladda ett godtyckligt antal tilläggsmoduler. Applikationen har till uppgift att starta JXTA-plattformen. Förutom detta ska JXTA-applikationen hantera grupperna, dvs. hålla reda på vilken grupp som är den aktuella och starta upp alla grupper som användaren är medlem i. Sist men inte minst ska den tillhandahålla det grundläggande användargränssnittet. I detta examensarbete har framförallt en tilläggsmodul utvecklats, en tjänst för fildelning.

Fildelningsmodulen har till uppgift att hantera de filer som finns delade i nätverket och pressentera de för användaren. Dessutom har modulen till

uppgift att tillhandahålla funktionalitet knuten till fildelning, t.ex. nerladdning och utdelning av lokala filer. Detta gör främst genom nyttjande av en JXTA tilläggstjänst som har utvecklas (mer om tjänsten senare). Denna tilläggstjänst arbetar på gruppnivå, detta innebär att modulen måste knyta en tilläggstjänst till varje grupp som användaren är medlem i. Detta gör det något mer komplicerat att nyttja tjänsten eftersom rätt instans måste användas, den som hör till den aktuella gruppen.

Utvecklingen av en ny tilläggstjänst till JXTA-plattformen innebar en inblick i hur funktionaliteten i plattformen kan utökas. Den tilläggstjänst som utvecklades var en tjänst för att dela filer inom gruppen. Det finns en befintlig informationsdelningstjänst (Content Management Service) utvecklat av JXTA community som har stöd för fildelning. Men den befintliga tjänsten har en brist när det gäller informationsspridning och versionshantering. Därför har en ny informationsdelningstjänst utvecklats med hjälp av idéer från den befintliga informationsdelningstjänsten för att öka informationsåtkomsten, som är en viktig del i ett informationsspridningssystem. Denna nya tjänst är tänkt att vara generell för att kunna återanvändas i andra system. Men det som har implementerats är stödet för fildelning. Informationen i tjänsten beskrivs av två typer av annonser för att hantera versioner på ett generellt sätt. Tjänsten använder Discovery Service för att sprida och upptäcka annonser inom gruppen. Resolver Service används för att hitta de noder som har informationen som annonserna beskriver. Pipor används för att överföra information mellan noderna.

Den enda förändring som gjorts i JXTA-plattformen är att utöka funktionaliteten i Discovery Service med aktiv sökning efter annonser inom gruppen. Detta för att hela tiden kunna upprätthålla den aktuella strukturen över grupperna som användaren är medlem i och att noderna hela tiden har den mest aktuella informationen.

10 Diskussion

Antalet klienter som ansluter till Internet ökar i snabb takt. Detta innebär att belastningen av den centrala delen av nätverket ökar och att stora mängder resurser hos klienterna förblir outnyttjade. Denna utveckling innebär att Internet blir mer och mer asymmetriskt. I framtiden kommer dessutom 3:e världen att på allvar börja ansluta sig till Internet. Det kommer att medföra en ännu större ökning av antalet användare. Om klient-servermodellens stora dominans fortsätter kommer det att medföra en överbelastning av de centrala delarna i nätverket, medan det finns enorma mängder outnyttjade resurser i de periferiska delarna. Denna överbelastning kommer att medföra ökade kostnader. Om däremot P2P-modellen får ett stort genomslag kommer resursutnyttjandet i nätverket att bli rättvisare och därigenom avlasta de centrala delarna i nätverket.

Klient-servermodellen bygger på att klienterna utnyttjar den centrala enhetens funktionalitet och information, vilket innebär att klienterna blir beroende av den centrala enheten. Detta är inte fallet i P2P-modellen, där funktionaliteten och informationen distribueras fritt i nätverket. Den centrala enheten i klient-servermodellen identifieras med en statisk nätverksadress. Medan parterna i P2P-modellen identifieras med en unik identifierare som är oberoende av nätverksadress. Att identifiera resurser i nätverket oberoende av nätverksadressen innebär ett mer dynamiskt och flexibelt nätverk som nog är mer anpassat till det sätt som Internet utnyttjas idag. För att P2P-modellen ska kunna få ett stort genomslag krävs det att ett antal problem löses.

Ett av de största problemen med P2P-modellen är säkerheten. I rena P2P-nätverk saknas det någon som har det övergripande ansvaret och kan bestämma vad som är rätt och fel. Detta gör att det är svårt att bestämma vilka som får ingå i nätverket och vilka regler som ska gälla. Klient-servermodellen består däremot av en central enhet som har det övergripande ansvaret i nätverket. En annan fördel är att det bara existerar en central enhet som tillhandahåller funktionaliteten och informationen, vilket gör den lättare att skydda. I P2P-modellen måste alla inblandade parter skyddas.

Förutom säkerheten finns det stora problem med den mängd bandbredd som krävs för att hålla ihop ett P2P-nätverk. Vid kommunikation genom hinder i nätverket är detta problem som störst. Oftast krävs det regelbunden kommunikation mellan noderna på ömse sidor om hindret för att kontrollera om det finns något att överföra. En lösning på detta problem är att anpassa brandväggar och andra hinder till P2P-modellen. Detta problem kompenseras

till viss del av att bandbreddsutnyttjandet fördelas bättre i nätverket med P2P-modellen.

Förutom dessa tekniska problem måste det till en attitydförändring om konceptet P2P. Alla måste vara beredda att dela med sig av sina resurser och inte bara utnyttja andras resurser. För att lösa detta problem måste nog säkerhetsfrågorna lösas först, ty vem vill dela med sig av sina resurser om det inte kan göras på ett säkert sätt.

En lösning på dessa problem kan vara att skapa en generell P2P-plattform som kan användas som grund för P2P-system i allmänhet. Detta skulle leda till att mycket tid och kunskap skulle ägnas åt varje del i plattformen, som i sin tur borde leda till en påskyndning av P2P-utveckligen. När det väl finns en standard så innebär det att den grundläggande P2P-funktionaliteten inte behöver utvecklas på nytt varje gång. En annan fördel med generella plattformar är att olika typer av applikationer kan använda samma plattform och därigenom samma P2P-nätverk. Detta bör minska den bandbredd som krävs för att hålla ihop P2P-nätverket.

I denna rapport har den generella P2P-plattformen JXTA granskats. JXTA är en öppen specifikation som kan implementeras oberoende av programspråk och plattform. JXTA är relativt nytt och första versionen släpptes under våren 2001. Utvecklingen har definitivt inte stagnerat vilket innebär att det inte är någon stabil plattform ännu. En fördel är att plattformen är väldigt generell och det mesta kan anpassas för sitt ändamål, men det innebär också en hög inlärningströskel för utvecklarna. Men det finns pågående projekt för att göra gränssnittet mot plattformen enklare för utvecklarna. För att JXTA ska kunna slå igenom på bred front krävs det att dokumentationen förbättras.

För att kunna avgöra om generella P2P-plattformar är ett alternativ till klient-servermodellen har prestandatester gjorts med avseende på informationsöverföring. För att kunna jämför modellerna har testerna delats in i två scenarion, kommunikation mellan två klienter och kommunikation mellan klient och tjänst. Dessa två scenarion är bland de vanligaste förekommande i Internet idag.

I det första scenariot skiljer sig kommunikationen mellan klienterna åt beroende på om det är klient-servermodellen eller P2P-modellen som används. Klient-servermodellen bygger på att informationen mellan klienterna utbyts med hjälp av en central server.

I det andra scenariot kommunicerar klient och tjänst med varandra. Skillnaden mot det föregående scenariot är att i klient-servermodellen sker kommunikationen direkt mellan klient och tjänst, eftersom att servern i det här fallet tillhandahåller tjänsten.

P2P-modellen bygger på att kommunikationen mellan noderna sker direkt eller indirekt beroende på nätverkets struktur. Detta är också fallet i de två scenarierna. Klient-servermodellen bygger på att en central server ska användas för all funktionalitet. Detta innebär att om två klienter ska kommunicera måste det ske via den centrala servern, vilket också återspeglas i det första scenariot. I det andra scenariot tillhandahåller den centrala servern den tjänst som klienten utnyttjar. I det första scenariot har P2P-modellen en fördel eftersom att direkt kommunikation kan utnyttjas om möjlighet finns. I det andra scenariot är modellerna mer likvärdiga eftersom att klient-servermodellen använder direkt kommunikation. När detta är fallet borde P2P-modellen också kunna kommunicera direkt med den andra parten, som i det här fallet är tjänsten. Resultaten visar att direkt kommunikation generellt har bättre prestanda än indirekt kommunikation. Prestandaförlusten mellan indirekt kommunikation jämfört med direkt kommunikation är försumbar för de flesta applikationer. Men vid vissa meddelandestorlekar ökar prestandaförlusten kraftigt och kan vara en faktor att räkna med.

Eftersom att testerna är gjorda i en kontrollerad testmiljö, utan påverkan av yttre faktorer, finns det dock andra aspekter att ta hänsyn till vid jämförelsen av modellerna i praktiken. En aspekt är att klienterna i P2P-modellen kan utbyta information med hjälp av ett godtyckligt antal noder i nätverket. Detta medför att klienterna kan kommunicera med varandra trots att det finns hinder i nätverket. Men denna indirekta kommunikation innebär en högre prestandaförlust jämfört med direkt kommunikation med en större flexibilitet. En annan aspekt är att kommunikationen i klient-servermodellen alltid sker till eller via samma centrala enhet, medan i P2P-modellen sker kommunikationen till eller via godtyckliga noder i nätverket. Med hjälp av P2P-modellen blir alltså belastningen av nätverket och noderna rättvisare.

Ytterligare en aspekt är vilken teknologi som används vid informationsöverföring. Resultaten visar att socket-kommunikation har överlägset bäst prestanda vid jämförelsen med BEEP och JXTA. Detta resultat är inte så anmärkningsvärt eftersom att socket-kommunikation är en ren kommunikationsteknik mellan två parter, medan BEEP är ett ramverk för skapandet av nya Internetprotokoll och JXTA är en generell P2P-plattform. Både BEEP och JXTA är transportprotokollsoberoende, men vid testerna använder båda teknikerna socket-kommunikation som grund vid kommunikation. Vid jämförelsen mellan BEEP och JXTA skiljer de sig åt beroende på kommunikationsmetod. JXTA har bättre prestanda vid direkt kommunikation jämfört med BEEP. Vid indirekt kommunikation har däremot

BEEP bättre prestanda. Detta kan verka egendomligt eftersom att JXTA är en P2P-teknik och borde då vara mer lämpad för indirekt kommunikation. Detta är kanske inte så egendomligt som det först kan synas eftersom JXTA inte bara har till uppgift att överföra information i den mellanliggande noden, utan är även ansvarig för att hålla ihop det virtuella nätverket. Medan den mellanliggande noden i BEEP har speciellt programmerats för att överföra information från det första nätverket till det andra nätverket, vilket ger BEEP en fördel prestandamässigt. Vid en närmare granskning av resultaten för BEEP så visar det sig att de förväntade prestandaförlusterna vid indirekt kommunikation har bytts mot prestandavinster. Detta är ett svårförklarat fenomen och författarna har inte hittat någon rimlig förklaring till detta.

Om teknologierna och modellerna förs samman visar det sig att socket-kommunikation är en lämplig teknologi som grund i både P2P-system och klient-serversystem, eftersom tekniken har bra prestanda när det gäller informationsöverföring. Det finns dock inget stöd för att upprätta det virtuella nätverket som ligger till grund för P2P-system. Detta innebär en ökad utvecklingskostnad eftersom att en plattform måste skapas, men kan ändå vara nödvändig om systemet ska specialanpassas för speciella krav. I övriga fall kan det vara mer lämpligt att använd en generell P2P-plattform vid utveckling av P2P-system. BEEP är ingen P2P-plattform och har ingen större fördel när det gäller prestanda vid informationsöverföring. Men eftersom att BEEP löser de vanligaste problemen vid nyutvecklig av kommunikationsprotokoll kan det med fördel användas vid utvecklig av klient-serversystem och P2P-plattformar.

P2P-modellen passar generellt de system där information och resurser ska delas mellan alla parter i nätverket. Meddelandetjänster, informationsdelning och distribuerade beräkningar är exempel på applikationer som bygger på informations- och resursdelning. P2P-modellen är generellt inte lämplig när systemet kräver hög säkerhet och permanent lagring av information.

11 Slutsats

Generella P2P-plattformar är relativt nytt och är fortfarande under utveckling till skillnad mot den beprövade klient-servermodellen. Idag finns det ett antal generella P2P-plattformar som är försök till att skapa en enhetlig standard. Ett sådant försök är JXTA. I skrivandestund finns det ingen plattform som har kommit så långt att den på allvar kan ses som ett seriöst alternativ i praktiken. Men utvecklingen går fort och inom en snar framtid bör det finnas ett antal fullgoda kandidater, kanske redan idag för vissa tillämpningar. Kan dessa då konkurrera med klient-servermodellen? På den frågan blir nog svaret: det beror på tillämpningsområdet. Med den rena P2P-modellen är det svårt att bygga upp system med bra säkerhet eftersom det inte finns någon som bestämmer. Men med hybrider finns det stora möjligheter att skapa system som kan mäta sig med klient-servermodellen, eftersom i hybridsystem nyttjas både den renodlade P2P-modellens och klient-servermodellens starka sidor. Denna möjlighet kan även nyttjas i generella P2P-plattformar. Detta medför att generella P2P-plattformar kan ses som ett alternativ till klient-servermodellen inom en snar framtid.

12 Tack

Författarna vill tacka alla anställda på TietoEnator DevCon i Skellefteå, särskilt Håkan Åhlund och handledaren Tomas Bjurman, för en trevlig tid.

Därefter vill författarna tacka Mikael Rännar som har varit handledare vid Umeå Universitet.

Sist men inte minst vill författarna tacka Per Edlund och Anna Lindgren för hjälp med granskningen av rapporten.

13 Källförteckning

13.1 Litteratur

[BRGOKR02] Daniel Brookshier, Darren Govoni och Navaneeth Krishana, *JXTA: Java P2P Programming*, 2002

[GR02] Joseph D. Gradecki, *Mastering JXTA: Building Java Peer-to-Peer Applications*, 2002

[WI02] Brendon J. Wilson, *JXTA*, 2002

13.2 Internet

[BEEP01] The Facts On BEEP, http://www.beepcore.org/beepcore/docs/wp-facts.jsp, 2001, hämtad 2002-10-25

[JXTA02] Project JXTA, http://www.jxta.org, 2002, hämtad 2002-09-09

[P2PCOMP] Peer to Peer Computing, http://www.eas.asu.edu/~p2pcom/p2p/index.htm, hämtad 2002-10-11

[SOCKET] Socketar i ett nötskal, http://www.ida.liu.se/~TDDB28/printout/Socketar.pdf, hämtad 2002-11-04

[SPEC01/02] JXTA v1.0 Protocols Specification, http://spec.jxta.org/source/browse/spec/www/v1.0/docbook/JXTAProtocols.html, 2001/2002, hämtad 2002-11-06

[TOP101] Distributed Systems Topologies: Part 1, http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html, 2001, hämtad 2002-10-24

[TOP202] Distributed Systems Topologies: Part 2, http://www.openp2p.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html, 2002, hämtad 2002-10-24

[WHATISP2P00] What Is P2P ... And What Isn't, http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html, 2000, hämtad 2002-09-09

Appendix A – Ordlista

ABNF Augmented Backus-Naur Form används för att definiera syntaxen för t.ex. programmeringsspråk och kommandon.

Broadcast En kommunikationsmetod där sändaren skickar ett meddelande till alla ansluta mottagare.

DNS Domian Name System är en Internettjänst som har till uppgift att översätta domännamn till IP-adresser.

HTTP HyperText Transfer Protocol är protokollet som används av WWW (World Wide Web). Protokollet definierar formatet på de meddelanden som skickas mellan webbservrar och webbläsare.

IETF Internet Engineering Task Force är ett konsortium av tekniker, tillverkare och forskare som arbetar för att göra Internet mer effektivt och lätt att använda. De ger ut egna standarder som i allmänhet följs.

IP Internet Protocol är ett nätverksprotokoll i Internetprotokollstacken. IP anger formatet för paketen som skickas i nätverket och hur de ska adresseras.

MIME Multipurpose Internet Mail Extensions är en standard för att skicka data innehållande text, bilder, och ljud i e-postmeddelanden.

Multicast En kommunikationsmetod där sändaren skickar ett meddelande till en känd mängd mottagare.

NAT Network Address Translation är en teknik för att översätta ett antal IP-adresser på det interna nätverket till ett antal IP-adresser utanför det interna nätverket.

Protokoll En uppsättning regler som anger hur kommunikationen mellan olika enheter ska utföras.

Router En router används för att dirigera trafiken i nätverket.

TCP Transport Control Protocol är ett transportprotokoll i Internetprotokollstacken. TCP erbjuder en förbindelseorienterad och tillförlitlig kommunikationslänk mellan två processer i nätverket.

TLS Transport Layer Security är ett transportprotokoll som erbjuder autentisering och kryptering.

URI Universal Resource Identifier är en abstraktion som innefattar både URL och URN. URI är det allmänna namnet på alla typer

av namn och adresser som globalt och unikt identifierar resurser i nätverket.

URL Uniform Resource Locator används för att globalt och unikt identifiera resursers lokalisering i nätverket.

URN Uniform Resource Name används för att globalt och unikt identifiera resurser med resursens namn, oberoende av lokalisering i nätverket.

XML Extensible Markup Language är ett enkelt och flexibelt textformat för att representera information på ett program- och plattformsoberoende sätt.

Appendix B – JXTA-identifierare

JXTA-identifierare beskrivs med standarden Uniform Resource Names (URN). Alla JXTA-identifierare tillhör namnrymden ”jxta”.

B.1 JXTA ID ABNF

Följande avsnitt definierar syntaxen för JXTA-identifierare på formen Augmented Backus-Naur Form (ABNF).

```
<JXTAURN>      ::=  "urn:" <JXTANS> ":" <JXTAIDVAL>
<JXTANS>       ::=  "jxta"
<JXTAIDVAL>    ::=  <JXTAFMT> "-" <JXTAIDUNIQ>
<JXTAFMT>      ::=  1 * <URN chars>
<JXTAIDUNIQ>   ::=  1 * <URN chars>
<URN chars>    ::=  <trans> | "%" <hex> <hex>
<trans>        ::=  <upper> | <lower> | <number> | <other> |
                    <reserved>
<upper>        ::=  "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
                    "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" |
                    "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
                    "Y" | "Z"
<lower>        ::=  "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
                    "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
                    "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
                    "y" | "z"
<hex>          ::=  <number> | "A" | "B" | "C" | "D" | "E" | "F" |
                    "a" | "b" | "c" | "d" | "e" | "f"
<number>       ::=  "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                    "8" | "9"
<other>        ::=  "(" | ")" | "+" | "," | "-" | "." |
                    ":" | "=" | "@" | ";" | "$" |
                    "_" | "!" | "*" | "'"
<reserved>     ::=  "%" | "/" | "?" | "#"
```

Appendix C – Annonser

Annonser beskriver alla typer av resurser i nätverket. Exempel på resurser är: noder, grupper, pipor och tjänster. Alla annonser i JXTA representeras med hjälp av standarden XML.

C.1 Common Advertisement Fragments

Dessa element är gemensamma för alla annonser.

```
<xs:element name="JXTAID" type="jxta:JXTAID"/>

<xs:simpleType name="JXTAID">
  <xs:restriction base="xs:anyURI">
    <xs:pattern value="([uU][rR][nN]:[jJ][xX][tT][aA]:)+\-+"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="serviceParam">
  <xs:all>
    <xs:element name="MCID" type="jxta:JXTAID"/>
    <xs:element name="Parm" type="xs:anyType"/>
  </xs:all>
</xs:complexType>

<xs:element name="Cred" type="jxta:Cred"/>
<xs:complexType name="Cred">
  <xs:all>
  </xs:all>
</xs:complexType>
```

C.2 Peer Advertisement

Denna annons beskriver en nod.

```
<xs:element name="PA" type="jxta:PA"/>

<xs:complexType name="PA">
  <xs:sequence>
    <xs:element name="PID" type="JXTAID"/>
    <xs:element name="GID" type="JXTAID"/>
    <xs:element name="Name" type="xs:string" minOccurs="0"/>
    <xs:element name="Desc" type="xs:anyType" minOccurs="0"/>
    <xs:element name="Svc" type="jxta:serviceParams"
        minOccurs="0" maxOccurs="unbounded"/>
  <xs:sequence>
</xs:complexType>
```

<PID>: Detta element innehåller nodens unika identifierare.

<GID>: Detta element innehåller identifieraren till den grupp som noden tillhör.

<Name>: Element används för att definiera namnet på noden. Det finns inget krav på att detta element ska vara unikt. Det är valfritt att använda detta element.

<Desc>: Elementet innehåller en beskrivning av noden som kan användas för indexering och sökning. Det finns ingen garanti att beskrivningen är unik. Detta element är valfritt.

<Svc>: Elementet används för att associera en grupptjänst med ett antal parametrar. Ett exempel kan vara att associera Endpoint Service med ändpunktadresserna. Detta element kan existera i godtyckligt antal.

C.3 Peer Group Advertisement

Denna annons beskriver en grupp.

```
<xs:element name="PGA" type="jxta:PGA"/>

<xs:complexType name="PGA">
  <xs:sequence>
    <xs:element name="GID" type="jxta:JXTAID"/>
    <xs:element name="MSID" type="jxta:JXTAID"/>
    <xs:element name="Name" type="xs:string" minOccurs="0"/>
    <xs:element name="Desc" type="xs:anyType" minOccurs="0"/>
    <xs:element name="Svc" type="jxta:serviceParam"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<GID>: Detta element innehåller gruppens unika identifierare.

<MSID>: Elementet innehåller en identifierare till modulspecifikation som specificerar gruppen. Specifikation definierar den grundläggande funktionalitet som finns inom gruppen.

<Name>: Elementet definierar namnet på gruppen, är valfritt och behöver inte vara unikt.

<Desc>: Elementet innehåller en beskrivning av gruppen som kan användas för indexering och sökning. Det finns ingen garanti att beskrivningen är unik. Detta element är valfritt.

<Svc>: Elementet används för att associera en grupptjänst med ett antal parametrar. Detta element kan existera i godtyckligt antal.

C.4 Module Class Advertisement

Denna annons beskriver en modulklass. En modulklass används för att dela in moduler med liknande funktionalitet i samma klass.

```
<xs:element name="MCA" type="jxta:MCA"/>

<xs:complexType name="MCA">
  <xs:sequence>
    <xs:element name="MCID" type="jxta:JXTAID"/>
    <xs:element name="Name" type="xs:string" minOccurs="0"/>
    <xs:element name="Desc" type="xs:anyType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

<MCID>: Elementet innehåller modulklassens unika identifierare.

<Name>: Detta element definierar namnet på modulklassen. Det finns inget krav på att detta namn ska vara unikt. Det är valfritt att använda detta element.

<Desc>: Detta element innehåller en beskrivning av modulklassen. Det finns inga krav på att detta element ska finns med i annonsen.

C.5 Module Specification Advertisement

Denna annons beskriver en modulspecifikation. En modulspecifikation används för att specificera moduler på ett plattformsoberoende sätt. Annonsen har två huvudsyften. Det första är att tillhandahålla dokumentation om hur modulen ska implementeras. Det andra huvudsyftet är att tillhandahålla information om hur kommunikation kan ske med de noder som har implementerat specifikationen.

```
<xs:element name="MSA" type="jxta:MSA"/>

<xs:complexType name="MSA">
  <xs:sequence>
    <xs:element name="MSID" type="jxta:JXTAID"/>
    <xs:element name="Vers" type="xs:string"/>
    <xs:element name="Name" type="xs:string" minOccurs="0"/>
    <xs:element name="Desc" type="xs:anyType" minOccurs="0"/>
    <xs:element name="Crtr" type="xs:string" minOccurs="0"/>
    <xs:element name="SURI" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="Parm" type="xs:anyType" minOccurs="0"/>
    <xs:element ref="jxta:PipeAdvertisement" minOccurs="0"/>
    <xs:element name="Proxy" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="Auth" type="jxta:JXTAID" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

<MSID>: Elementet innehåller modulspecifikationens unika identifierare.

<Vers>: Detta element specificerar specifikationens version.

<Name>: Detta element definierar modulspecifikationens namn. Det finns inget krav på att detta namn ska vara unikt. Det är valfritt att använda detta element.

<Desc>: Elementet innehåller en beskrivning av modulspecifikationen. Det är valfritt att använda detta element.

<CRTR>: Detta element innehåller namnet på skaparen av modulspecifikationen. Det är inget krav på att använda detta element.

<SURI>: Elementet innehåller en URI som refererar till specifikationens dokumentation. Det är valfritt att använda detta element.

<Parm>: Parametrar till implementationen.

<jxta:PipeAdvertisement>:	Detta element innehåller en annons som beskriver en pipa som kan användas för att kommunicera med en implementation av denna modulspecifikationen. Detta element är inte ett krav.
<Proxy>:	Elementet innehåller en identifierare som unikt refererar till en proxymodul som kan användas för att kommunicera med implementationer av denna modulspecifikation. Denna kommunikation kan även ske rekursivt via flera proxymoduler.
<Auth>:	Detta element innehåller en identifierare som unikt refererar till en autentiseringsmodul som har till uppgift att autentisera noden innan kommunikation kan ske med implementationer av denna modulspecifikation.

C.6 Module Implementation Advertisement

Denna annons beskriver en modulimplementation. En Modulimplementation beskriver en implementation av en modulspecifikation.

```
<xs:element name="MIA" type="jxta:MIA"/>

<xs:complexType name="MIA">
  <xs:sequence>
    <xs:element name="MSID" type="jxta:JXTAID"/>
    <xs:element name="Comp" type="xs:anyType"/>
    <xs:element name="Code" type="xs:anyType"/>
    <xs:element name="PURI" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="Prov" type="xs:string" minOccurs="0"/>
    <xs:element name="Desc" type="xs:anyType" minOccurs="0"/>
    <xs:element name="Parm" type="xs:anyType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

<MSID>: Detta element innehåller modulimplementationens unika identifierare.

<Comp>: Detta element innehåller en beskrivning av miljön där implementationen kan exekveras.

<Code>: Detta element innehåller information för att ladda och köra implementationen. I vissa fall kan detta element innehålla hela programkoden för modulen.

<PURI>: Detta element innehåller en URI som refererar till paketet som innehåller implementationen av denna modul

<Prov>: Elementet innehåller namnet den som tillhandahåller implementationen.

<Desc>: Elementet innehåller en beskrivning av implemnetationen. Detta element kan användas som föremål för sökning. Det är valfritt att inkludera detta element i annonsen.

<Parm>: Parametrar till implementationen.

Appendix D – Endpoint Routing Protocol

Ett meddelande som ska skickas mellan två noder som inte kan kommunicera direkt med varandra måste ta hjälp av andra noder i nätverket för detta ändamål. För att veta vilka noder som ska ingå i rutten från startpunkten till slutdestinationen används Endpoint Routing Protocol.

D.1 Route Advertisement

Denna annons beskriver rutten mellan två noder.

```
<xs:element name="APA" type="jxta:APA"/>

<xs:complexType name ="jxta:APA">
  <xs:sequence>
    <xs:element name="EA" type="jxta:JXTAID" minOccurs="1"
        maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="RA" type="jxta:RA"/>

<xs:complexType name ="jxta:RA">
  <xs:sequence>
    <xs:element name="DstPID" type="xs:anyURI"/>
    <xs:element ref="jxta:RA"/>
    <xs:element name="Hops" minOccurs="0">
      <xs:sequence>
        <xs:element ref="jxta:APA" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

<DstPID>: Elementet innehåller en identifieraren till noden som utgör slutdestinationen i rutten.

<APA>: Elementet innehåller en lista med ändpunktadresser som är associerad med noden som utgör slutdestinationen i rutten.

<Hops>: Detta element innehåller en lista med ändpunktadresser. Denna lista beskriver rutten till noden som utgör slutdestinationen i rutten.

D.2 Endpoint Router Query

Detta meddelande används för att hitta rutten mellan två noder.

```
<xs:element name="ERQ" type="jxta:ERQ"/>

<xs:complexType name ="jxta:ERQ">
  <xs:sequence>
    <xs:element name="Dst" type="jxta:JXTAID"/>
    <xs:element name="Src">
      <xs:element ref="jxta:RA"/>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

<Dst>: Detta element innehåller den sökta nodens identifierare.

<Src>: Element innehåller annonsen som beskriver noden som gör förfrågan om rutten. Detta för att svaret ska kunna skickas tillbaka.

D.3 Endpoint Router Response

Detta meddelande används för att svara när en rutt mellan två noder har hittats.

```
<xs:element name="ERR" type="jxta:ERR"/>

<xs:complexType name ="jxta:ERR">
  <xs:sequence>
    <xs:element name="Dst">
      <xs:element ref="jxta:RA"/>
    </xs:element>
    <xs:element name="Src">
      <xs:element ref="jxta:RA"/>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

<Dst>: Detta element innehåller den sökta nodens identifierare.

<Src>: Element innehåller annonsen Route Advertisement som beskriver rutten från startpunkten till slutdestinationen.

D.4 Endpoint Router Message Element

Detta element beskriver rutten mellan två noder och adderas till de meddelanden som ska skickas från startpunkten till slutdestinationen.

```
<xs:element name="ERM" type="jxta:ERM"/>

<xs:complexType name="jxta:ERM">
  <xs:sequence>
    <xs:element name="Src" type="jxta:JXTAID"/>
    <xs:element name="Dest" type="jxta:JXTAID"/>
    <xs:element name="LastHop" minOccurs="0"
        type="jxta:JXTAID"/>
    <xs:element name="Fwd">
      <xs:sequence>
        <xs:element ref="jxta:APA" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
    <xs:element name="Rvs" minOccurs="0">
      <xs:sequence>
        <xs:element ref="jxta:APA" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
```

<Src>: Elementet innehåller ändpunktadressen för noden som utgör startpunkten i rutten.

<Dest>: Elementet innehåller ändpunktadressen för noden som utgör slutdestinationen i rutten.

<LastHop>: Ändpunktadressen till den nod i rutten som sist skickade vidare meddelandet.

<Fwd>: Elementet innehåller en lista av annonser som beskriver rutten till slutdestinationen.

<Rvs>: Elementet innehåller en lista av annonser som beskriver rutten till startpunkten.

D.5 Endpoint Address URI ABNF

Ändpunktadresserna i JXTA identifieras med standarden URI. Följande avsnitt definierar syntaxen för ändpunktadresserna på formen Augmented Backus-Naur Form (ABNF).

```
<ENDPOINTADDRESS>  ::= (<URISCHEME> <PROTOCOLADDR>
                       0*1("/" <RECIPIENT> "/"
                       0*1(<RECIPIENTPARAM>))) |
                       (<URNSCHEME> <PROTOCOLADDR>
                       0*1("?" <RECIPIENT> "/"
                       0*1(<RECIPIENTPARAM>)))
<PROTOCOL>         ::=  <URISCHEME> | <URNSCHEME>
<URISCHEME>        ::=  1 (<upper> | <lower>) 0* <URICHARS>
                        <SCHEMESEP>
<SCHEMESEP>        ::=  ":" | "://"
<URNSCHEME>        ::=  "urn:" 1 (<upper> | <lower>)
                        0* <URICHARS> ":"
<URICHARS>         ::=  <upper> | <lower> | <number> |
                        "%" <hex> <hex>
<PROTOCOLADDR>     ::=  0* (<URICHARS> |  <other>)
<RECIPIENT>        ::=  0* (<URICHARS> |  <other>)
<RECIPIENTPARAM>   ::=  0* <URN chars>
```

Appendix E – Peer Resolver Protocol

Protokollet definierar ett generellt sätt för noder att ställa frågor till andra noder inom gruppen. Protokollet består av två typer av meddelande; ett frågemeddelande och ett svarsmeddelande.

E.1 Resolver Query

Detta meddelande används för att skicka en generell fråga till en annan nod inom gruppen.

```
<xs:element name="ResolverQuery" type="jxta:ResolverQuery"/>

<xs:complexType name="ResolverQuery">
  <xs:all>
    <xs:element ref="jxta:Cred" minOccurs="0"/>
    <xs:element name="SrcPeerID" type="jxta:JXTAID"/>
    <!-- This could be extended with a pattern restriction -->
    <xs:element name="HandlerName" type="xs:string"/>
    <xs:element name="QueryID" type="xs:string"/>
    <xs:element name="Query" type="xs:anyType"/>
  </xs:all>
</xs:complexType>
```

<jxta:Cred>: Detta element innehåller sändarens intyg för att få sända frågor inom gruppen.

<HandlerName>: Elementet innehåller namnet på den hanterare ska hantera frågan i mottagaren.

<SrcPeerID>: Detta element innehåller en identifierare som identifierar noden som gör förfrågan.

<QueryID>: Elementet innehåller frågans unika identifierare som används för att matcha frågan med eventuella svar.

<Query>: Elementet innehåller själva frågan.

E.2 Resolver Response

Detta meddelande används för att svara på en fråga.

```
<xs:element name="ResolverResponse" type="ResolverResponse"/>

<xs:complexType name="ResolverResponse">
  <xs:all>
    <xs:element ref="jxta:Cred" minOccurs="0"/>
    <xs:element name="HandlerName" type="xs:string"/>
    <xs:element name="QueryID" type="xs:string"/>
    <xs:element name="Response" type="xs:anyType"/>
  </xs:all>
</xs:complexType>
```

<jxta:Cred>: Elementet innehåller sändarens intyg för att få sända svar inom gruppen.

<HandlerName>: Elementet specificerar hur svaret ska hanteras.

<QueryID>: Elementet innehåller en unik identifierare som refererar till frågan som detta meddelande svarar på.

<Response>: Svaret på frågan.

E.3 Resolver Shared Resource Distributed Index (SRDI)

Resolver SRDI meddelandet används för att kunna skicka frågor till de noder som mest troligen kan svara på frågan.

```
<xs:element name="ResolverSRDI" type="jxta:ResolverSRDI"/>

<xs:complexType name="ResolverSRDI">
  <xs:all>
    <xs:element name="HandlerName" type="xs:string"/>
    <xs:element ref="jxta:Cred" minOccurs="0"/>
    <xs:element name="Payload" type="xs:anyType"/>
  </xs:all>
</xs:complexType>
```

<HandlerName>: Elementet innehåller namnet på den hanterare ska hantera meddelandet i mottagaren.

<jxta:Cred>: Elementet innehåller sändarens intyg för att få sända meddelanden inom gruppen.

<Payload>: Detta element innehåller informationen som ska levereras till hanteraren i mottagaren.

E.4 Listener Element Naming ABNF

Följande avsnitt definierar syntaxen för att namnge hanterare. Syntaxen för hanterarnamnen beskrivs på formen Augmented Backus-Naur Form (ABNF).

```
<JXTARSLVRRSQRY>   ::=  <JXTARSLVRNAM> <JXTAIDVAL>
                        <JXTARSLVRQRYTAG>
<JXTARSLVRRSRSP>   ::=  <JXTARSLVRNAM> <JXTAIDVAL>
                       <JXTARSLVRRSPTAG>
<JXTARSLVRRSSRDI>  ::=  <JXTARSLVRNAM> <JXTAIDVAL>
                        <JXTARSLVRSRDITAG>
<JXTARSLVRQRYTAG>  ::=  "ORes"
<JXTARSLVRRSPTAG>  ::=  "IRes"
<JXTARSLVRSRDITAG> ::=  "Isrdi"
<JXTARSLVRNAM>     ::=  "jxta.service.resolver"
<JXTAIDVAL>        ::=  SEE
```

Appendix F – Peer Discovery Protocol

Peer Discovery Protocol definierar meddelanden för att söka efter annonser i nätverket. Protokollet består av två meddelanden, ett för att söka annonser och ett för att skicka svar på sökningen.

F.1 Discovery Query

Detta meddelande används för att söka efter annonser.

<xs:element name="DiscoveryQuery" type="jxta:DiscoveryQuery"/>

```
<xsd:simpleType name="DiscoveryQueryType">
  <xsd:restriction base="xsd:string">
    <!-- peer -->
    <xsd:enumeration value="0"/>
    <!-- group -->
    <xsd:enumeration value="1"/>
    <!-- adv -->
    <xsd:enumeration value="2"/>
  </xsd:restriction>
</xsd:simpleType>

<xs:complexType name="DiscoveryQuery">
  <xs:sequence>
    <xs:element name="Type" type="jxta:DiscoveryQueryType"/>
    <xs:element name="Threshold" type="xs:unsignedInt"
        minOccurs="0"/>
    <xs:element name="Attr" type="xs:string" minOccurs="0"/>
    <xs:element name="Value" type="xs:string" minOccurs="0"/>
    <!-- The following should refer to a peer adv,
         but is instead a whole doc for historical reasons -->
    <xs:element name="PeerAdv" type="xs:string"
        minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

<Type> : Elementet definierar vilken typ av annons som sökes. Möjliga värden på elementet är :

"0": Nod annonser (Peer Advertisements)

"1": Grupp annonser (Peergroup Advertisements)

"2": Andra annonser

<Threshold> : Elementet anger det maximala antalet svar som önskas från varje nod. Om både elementet ”Type” och elementet ”Threshold” har värdet 0, betyder det

	att alla noder som nås av frågan ska svara även om svaret inte innehåller någon annons.
<PeerAdv> :	Detta element innehåller annonsen som beskriver noden som skickar förfrågan.
<Attribute>, <Value>:	För att specificera sökningen kan dessa attribut användas. Elementet ”Attribute” definierar namnet på det attribut som ska finnas i de annonser som söks och elementet ”Value” värdet på attributet. Om elementen inte används sker ett slumpmässigt urval av annonserna. Det finns möjligheter att använda sig av ”*” i början eller slutet av ett värde. Detta gör att matchningen sker fram till ”*” eller efter ”*”. Om ”*” anges som värde är det inte definierat vad det matchar.

F.2 Discovery Response

Detta meddelande används för att svara på en sökning. Det är frivilligt att svara på en sökning.

```
<xs:element name="DiscoveryResponse"
type="jxta:DiscoveryResponse"/>

<xs:complexType name="DiscoveryResponse">
  <xs:sequence>
    <xs:element name="Type" type="jxta:DiscoveryQueryType"/>
    <xs:element name="Count" type="xs:unsignedInt"
        minOccurs="0"/>
    <xs:element name="Attr" type="xs:string" minOccurs="0"/>
    <xs:element name="Value" type="xs:string" minOccurs="0"/>
    <!-- The following should refer to a peer adv,
         but is instead a whole doc for historical reasons -->
    <xs:element name="PeerAdv" minOccurs="0">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="Expiration"
                type="xs:unsignedLong"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="Response" maxOccurs="unbounded">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="Expiration"
                type="xs:unsignedLong"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

<Type> :	Elementet definierar vilken typ av annons som returneras.
<Count> :	Detta element innehåller antalet annonser som detta meddelande returnerar.
<PeerAdv> :	Elementet innehåller annonsen som beskriver noden som returnerar annonsen eller annonserna.Detta element är valfritt.
<Attribute>, <Value>:	Dessa element innehåller de kriterier som ställs på svaret.
<Response> :	Elementet innehåller de annonser som matchar sökkriterierna och har den typ som är specificerad i frågan.

Appendix G – Rendezvous Protocol

Protokollet används för att kunna sprida meddelanden till noder inom gruppen.

G.1 Rendezvous Advertisement

Annonsen används för att annonsera beskriva en mötesplatsnod.

```
<xs:element name="RdvAdvertisement"
    type="jxta:RdvAdvertisement"/>

<xs:complexType name="RdvAdvertisement">
  <xs:sequence>
    <xs:element name="Name" type="xs:string" minOccurs="0"/>
    <xs:element name="RdvGroupId" type="jxta:JXTAID"/>
    <xs:element name="RdvPeerId" type="jxta:JXTAID"/>
  </xs:sequence>
</xs:complexType>
```

<Name>: Detta element användas för att associera mötesplatsnoden med ett namn. Detta element är valfritt.

<RdvGroupId>: Detta obligatoriska element innehåller en unik identifierare som identifierar gruppen som mötesplatsnoden agerar inom.

<RdvPeerId>: Detta obligatoriska element innehåller möteplatsnodens unika identifierare.

G.2 Rendezvous Propagate Message

Detta element adderas till meddelanden för att hindra meddelanden att spridas i all oändlighet.

```
<xs:element name="RendezVousPropagateMessage"
type="jxta:RendezVousPropagateMessage"/>

<xs:complexType name="RendezVousPropagateMessage">
  <xs:element name="MessageId" type="xs:string"/>
  <!-- This should be a constrained subtype -->
  <xs:element name="DestSName" type="xs:string"/>
  <xs:element name="DestSParam" type="xs:string"/>
  <xs:element name="TTL" type="xs:unsignedInt"/>
  <xs:element name="Path" type="xs:anyURI"
      maxOccurs="unbounded"/>
</xs:complexType>
```

Appendix H – Peer Information Protocol

Peer Information Protocol används för att noder ska kunna övervaka andra noder inom gruppen.

H.1 PIP Query Message

Detta meddelande används för att fråga efter information om en annan nod.

```
<xs:element name="PeerInfoQueryMessage"
    type="jxta:PeerInfoQueryMessage"/>

<xs:complexType name="PeerInfoQueryMessage">
  <xs:sequence>
    <xs:element name="sourcePid" type="jxta:JXTAID"/>
    <xs:element name="targetPid" type="jxta:JXTAID"/>
    <!-- if not present then the response is the general peer
         info -->
    <xs:element name="request" type="xs:anyType"
        minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

<sourcePid>: Elementet innehåller en unik identifierare som identifierar noden som gör förfrågan.

<targetPid>: Detta element innehåller den förfrågade nodens identifierare.

<request>: Elementet innehåller eventuella frågestrukturer.

H.2 PIP Response Message

Detta meddelande används för att svara på en informationsfråga, dvs. PIP Query Message.

```
<xs:element name="PeerInfoResponse"
type="jxta:PeerInfoResponse"/>

<xs:complexType name="PeerInfoResponseMessage">
   <xs:sequence>
     <xs:element name="sourcePid" type="jxta:JXTAID"/>
     <xs:element name="targetPid" type="jxta:JXTAID"/>
     <xs:element name="uptime" type="xs:unsignedLong"
         minOccurs="0"/>
     <xs:element name="timestamp" type="xs:unsignedLong"
         minOccurs="0"/>
     <xs:element name="response" type="xs:anyType"
         minOccurs="0"/>
     <xs:element name="traffic" type="jxta:piptraffic"
         minOccurs="0"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="piptraffic">
  <xs:sequence>
    <xs:element name="lastIncomingMessageAt"
        type="xs:unsignedLong" minOccurs="0"/>
    <xs:element name="lastOutgoingMessageAt"
        type="xs:unsignedLong" minOccurs="0"/>
    <xs:element name="in" type="jxta:piptrafficinfo"
        minOccurs="0"/>
    <xs:element name="out" type="jxta:piptrafficinfo"
        minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>

<xs:complexType name="piptrafficinfo">
  <xs:sequence>
    <xs:element name="transport" maxOccurs="unbounded">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:unsignedLong">
            <xs:attribute name="Expiration" type="xs:anyURI"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

<sourcePid>: Elementet innehåller en unik identifierare som identifierar noden som gör förfrågan.

<targetPid>: Detta element innehåller den förfrågade nodens identifierar.

<uptime>: Elementet definierar hur lång tid den förfrågade noden har exekverat. Detta element bör finnas med i alla svar, men kan utelämnas i implementationer där information inte finns eller där det kan påverka säkerheten.

<timestamp>: Den absoluta tid när detta meddelande generades. Detta mätt i millisekunder sen den första januari, 1970, 00:00:00 GMT. Detta element bör finnas med i alla svar, men kan utelämnas i implementationer där information inte finns eller där det kan påverka säkerheten.

<response>: Detta element innehåller svaret på informationsfrågan.

<traffic>: Elementet innehåller information om nätverkstrafiken hos den svarande noden. Det är valfritt att använda detta element. Elementet kan ha följande underelement:

- <lastIncomingMessageAt>: Den absoluta tid när noden sist tog emot ett meddelande.
- <lastOutgoingMessageAt>: Den absoluta tid när noden sist skickade ett meddelande.
- <in>: Detta element beskriver den inkommande trafiken.
- <out>: Detta element beskriver den utgående trafiken.
- <transport>: Elementet innehåller antalet byte som skickats eller mottagits från en specifik ändpunktadress.

Appendix I – Pipe Binding Protocol

Pipe Binding Protocol används för att upprätta virtuella kommunikationskanaler, kallade pipor, mellan en eller flera noder. När kommunikation ska ske mellan två noder krävs det att den som ska skicka information genom pipan vet att det är någon som är redo att ta emot informationen. För att lösa detta används Pipe Binding Protocol. Pipe Binding Protocol definierar en annons och ett meddelande.

I.1 Pipe Advertisment

Pipe Advertisement är en annons som används för att annonsera om pipor.

```
<xs:element name="PipeAdvertisment"
type="jxta:PipeAdvertisment"/>

<xs:complexType name="PipeAdvertisement">
  <xs:element name="Name" type="xs:string" minOccurs="0"/>
  <xs:element name="Id" type="JXTAID"/>
  <xs:element name="Type" type="xs:string"/>
</xs:complexType>
```

<Name>: Detta element innehåller ett valfritt namn som associeras med pipan. Det finns inget krav på att namnet ska vara unikt.

<Id>: Elementet innehåller pipans unika identifieraren.

<Type>: Detta element beskriver pipans typ. Följande typer finns definierade:

"JxtaUnicast":	En enkelriktad, opålitlig och osäker pipa mellan två noder.
“JxtaUnicastSecure":	En enkelriktad, opålitlig och säker pipa mellan två noder.
"JxtaPropagate":	En enkelriktad, opålitlig och osäker pipa mellan en nod till flera andra noder.

I.2 Pipe Resolver Message

Med hjälp av detta meddelande kan en nod ta reda på om det är någon som är redo att ta emot information i andra ändan av en specifik pipa.

```
<xs:element name="PipeResolver" type="jxta:PipeResolver"/>
<xs:complexType name="PipeResolver">
  <!-- MsgType should be an enumeration choice -->
  <xs:element name="MsgType" type="xs:string"/>
  <xs:element name="PipeId" type="JXTAID"/>
  <xs:element name="Type" type="xs:string"/>
  <xs:element name="Peer" type="JXTAID" minOccurs="0"/>

  <!-- following are used in the query -->
  <xs:element name="Cached" type="xs:boolean"
      default="false"minOccurs="0"/>

  <!-- following are used in the answer -->
  <xs:element name="Found" type="xs:boolean" minOccurs="0"/>
  <!-- This should refer to a peer adv,
       but is instead a whole doc -->
  <xs:element name="PeerAdv" type="xs:string" minOccurs="0"/>
</xs:complexType>
```

<MsgType>: Detta element används för att definiera om meddelandet ska vara ett frågemeddelande eller ett svarsmeddelande. Om elementet innehåller texten ”Query” är meddelandet ett frågemeddelande. Om elementet innehåller texten ”Answer” är meddelandet ett svarsmeddelande.

<PipeId>: Elementet innehåller identifieraren som unikt identifierar pipan som ska bindas.

<Type>: Detta element innehåller typen på den pipa som meddelandet avser.

<Cached>: Detta element bestämmer om svaret får komma från tidigare lagrade svar i det lokala lagringsutrymmet, detta innebär att svaret kan vara inaktuellt. Om elementet har värdet ”True” får svaret vara ett tidigare lagrat svar. Om elementet har värdet ”False” måste svaret vara aktuellt, vilket innebär att det lokala lagringsutrymmet inte kan användas som källa.

<Peer>: Detta element används för att definiera till vilken nod förfrågan ska skickas. Alltså det är bara denna nod som ska svara på förfrågan. Elementet innehåller nodens identifierare. Men detta garanterar inte att ett svar kommer från den aktuella noden. Det är alltid frivilligt att svara på en fråga. Om detta element ingår i svarsmeddelande refererar det till alla noder som har bundit en mottagarända av pipan.

<Found>: Elementet används för att indikera om motagarändan fanns på den specificerade noden.

<PeerAdv>: Elementet innehåller annonsen som beskriver noden som har bundit motagarändan av pipan. Identifieraren till denna nod kan finnas med i elementet ”Peer”, men det kan inte tas för givet.

Appendix J – Testuppställning

Detta avsnitt innehåller specifikationen för testuppställningen vid prestandatesterna. Följande versioner av programvara har använts i båda testfallen:

- **JXTA Version** - JXTA Project Stable Builds (STABLE_20020924T1446PDT, 09-24-2002)
- **BEEP -** Perma BEEP Version 0.8 beta

J.1 Testfall 1

Ett 10 Mbits Ethernet har använts vid kommunikationen mellan datorerna.

J.1.1 Dator 1

Hårdvara

Intel 400 MHz

64 MB Internminne

Operativsytem

Microsoft Windows 2000 5.00.2195

Java

java version "1.4.0_01"

Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0_01-b03)

Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)

J.1.2 Dator 2

Hårdvara

Intel 300 MHz

128 MB Internminne

Operativsytem

Microsoft Windows 2000 5.00.2195 Service Pack 3

Java

java version "1.4.0_01"

Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0_01-b03)

Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)

J.2 Testfall 2

Ett 100 Mbits Ethernet har använts vid kommunikationen mellan dator 1 och 2 och ett 10 Mbits Ethernet har använts vid kommunikationen mellan dator 2 och 3.

J.2.1 Dator 1

Hårdvara

Intel 400 MHz

64 MB Internminne

Operativsytem

Microsoft Windows 2000 5.00.2195

Java

java version "1.4.0_01"

Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0_01-b03)

Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)

J.2.2 Dator 2

Hårdvara

2 stycken Intel 200 MHz

128 MB Internminne

Operativsytem

Linux

Java

java version "1.4.0_01"

Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0_01-b03)

Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)

J.2.3 Dator 3

Hårdvara

Intel 300 MHz

128 MB Internminne

Operativsytem

Microsoft Windows 2000 5.00.2195 Service Pack 3

Java

java version "1.4.0_01"

Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0_01-b03)

Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)

Appendix K – Kravspecifikation

Förslag till JXTA-implementation

Ämne: Informationsspridning

Målgrupp: I första hand utvecklare

Uppgift: Designa och implementera ett system för informationsspridning.

Exempel på information:

- Fildelning
    - Klienten notifierar användaren att ny version av en fil finns alt. att filen automatiskt tankas ned (valbart per klient).
- Länkar - förmodligen en typ av fil med speciell hantering
    - Man vill skicka en länk och beskrivning, jmfr. ICQ URL's.
- Bulletin Board
    - Typ usenet
    - Inlägg/Svar på inlägg
    - Grupp-/ämnesindelning
    - Klienten bestämmer hur länge inläggen ska cachas lokalt.
- IRC
    - Till alla i gruppen, eller enskild klient

Generellt:

- Pling - händelser visas på varje klient det berör.
    - På Win32-implementationen så kan man tänka sig en liten systray-ikon...
- "Pusha" filer till gruppmedlemmar. Här bör man tänka sig för och kanske inte trycka ut filerna utan bara information om filerna typ version/storlek etc. och sedan låta klienten välja om det ska tankas ned eller ej.
- Varje "lyckad" nedladdning av ett objekt ska generera en "annonsering"...
- "Central" administration