Avdelning för datavetenskap

Anders Broström och Niclas Kihlstadius

Prototyp av en VoIP/PSTN-gateway

Prototype of a VoIP/PSTN gateway

Examensarbete 10 p
Dataingenjör

Datum:	07-06-05
Handledare:	Donald F. Ross
Examinator:	Martin Blom
Löpnummer:	C2007:03

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Prototyp av en VoIP/PSTN-gateway

Anders Broström och Niclas Kihlstadius

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Anders Broström och Niclas Kihlstadius

Godkänd, Datum

Handledare: Donald F. Ross

Examinator: Martin Blom

Sammanfattning

Under de senaste åren har Internettelefonin varit på frammarsch, och i takt med att tekniken mognat har fler och fler börjat se den som ett alternativ till att ringa via telefonnätet. Förutom att det är billigare att ringa över det förstnämnda, så erbjuder Internettelefonin också en rad revolutionerande tjänster. Det är dock troligt att telefonnätet kommer att få tjänstgöra i många år till, och det erbjuder fortfarande överlägset bäst stabilitet och har stor acceptans. Om de två telefoninätverken ska existera sida vid sida, med varsina användarbaser är det lämpligt om de kan fås att samverka, så att användare av det ena kan ringa användare av det andra, och vice versa. Detta kan göras med en VoIP/PSTN-gateway, som översätter kontrollinformation och rösttrafik mellan de två nätverken.

Uppsatsen handlar om det arbete vi har utfört år TietoEnator i Karlstad. Uppgiften bestod i att utveckla en prototyp av en VoIP/PSTN-gateway. Från början var det avsett att systemet skulle klara uppringning från endera en ”vanlig” telefon, eller en så kallad IP-telefon. Därtill skulle rösttrafiken överföras genom ändamålsenlig hårdvara. För att utföra arbetet behövde vi först studera relevanta kommunikationsprotokoll både för telefonnätet och för Internet, för att se hur dessa kunde fås att samverka. Vi behövde också lära oss tillgängliga system, bibliotek och verktyg för att förstå hur vi skulle skapa vårt eget system i den efterkommande implementeringsfasen. På grund av en lång inläsningsperiod och inledande tekniska problem, samt att nödvändig hårdvara för översättning av rösttrafiken inte anlände i tid begränsades arbetet till att innefatta samtal initierade från den vanliga telefonen till ip-telefonen, utan röstöverföring. Likväl har ett resultatgivande arbete utförts, och det beskrivs i detalj i rapporten.

Abstract

During the past few years Internet telephony has advanced rapidly, and as the technology has evolved, more and more have come to consider it an alternative to making phone calls through the telephone network. Besides being cheaper, Internet telephony also provides several revolutionary services. It is likely though that the telephone network will remain in use for several years to come, and it still offers by far the best stability and is accepted by most people. If the two networks are to coexist, with their respective users, it would be useful if they could be made to interact, so that users of one network can call users of the other, and vice versa. This can be done with a VoIP/PSTN gateway, which translates control information and voice traffic between the two networks.

Our dissertation is about the work we have performed for TietoEnator in Karlstad. The assignment was to develop a prototype of a VoIP/PSTN gateway. Initially the system was meant to support phone calls initiated either from an “ordinary” phone or from an IP telephone. Also the voice traffic was supposed to be translated with the use of appropriate hardware. To manage this we first needed to study all the relevant protocols for communication used in the telephone network and on the Internet, to get an idea of how these could be made to interact. We also had to learn existing systems, libraries and tools in order to see how we could create our own system. Due to a long learning period and technical problems in the beginning, and because the necessary hardware equipment for translation of voice traffic did not arrive in time, the assignment was limited to include only calls initiated from the ordinary phone to the IP telephone, without voice transmission. Never the less, the efforts have produced results, and our work is explained in detail in this dissertation.

Innehållsförteckning

Figurförteckning

Tabellförteckning

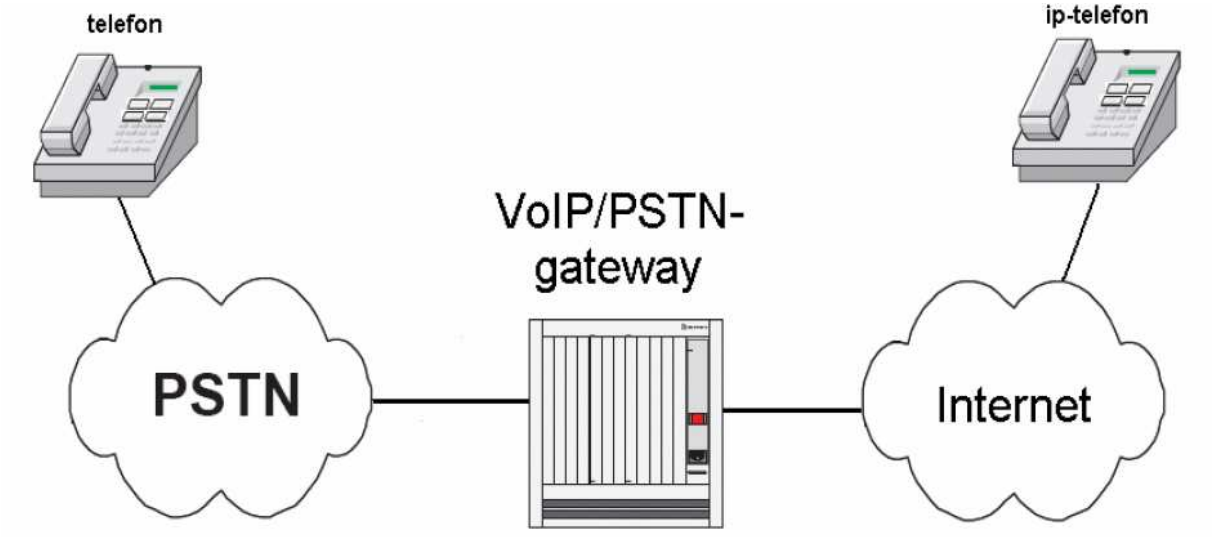
1 Inledning

Telekommunikationsindustrin som vi känner den håller på att förändras. Från att ha varit hårt knuten till det klassiska, världsomspännande telefonnätet (Public Switched Telephone Network, PSTN) har telefoni på senare år även blivit en tillgänglig tjänst som kan erbjudas via Internet (ofta kallat Voice over IP, VoIP, ip-telefoni eller internettelefoni). Detta är en intressant utveckling av flera skäl, men en av huvudanledningarna till övergången är utan tvekan de stora ekonomiska besparingar som kan göras. Exempelvis är ett långdistanssamtal över Internet billigare än motsvarande över telefonnätet. Den ekonomiska aspekten speglas även i att resurserna nyttjas effektivare, vilket är en direkt följd av de principiella skillnaderna mellan kretskopplade nät som telefonnätet, och paketförmedlande nät som Internet. När en användare av det förstnämnda ringer till en annan användare tilldelas samtalet dem emellan en på förhand bestämd kapacitet för överföring av media (röst). Storleken av den tilldelade kapaciteten förblir alltid den samma, oavsett om andra användare nyttjar nätet eller inte. Den resterande kapaciteten går helt enkelt förlorad om den inte används. Det vore därför mer effektivt om man istället alltid utnyttjade resurserna optimalt, vilket skulle leda till att ett ensamt samtal fick tillgång till all tillgänglig kapacitet. Rent praktiskt skulle det yttra sig i bättre ljudkvalité för deltagarna i samtalet, eftersom mer information kan överföras under lika lång tid. Och tack vare den underliggande nätverksarkitekturen är detta fullt möjligt när telefonsamtal förs över Internet.

Men att använda Internet istället för telefonnätet har även en rad andra intressanta fördelar, inte minst de nya tjänster som erbjuds användarna. Istället för ett telefonnummer kopplat till en enhet eller plats används adresser (inte helt olikt e-mailadresser) kopplade till användare, och genom ett snillrikt nätverk av servrar med information om vart dessa adresser leder kan en användare nås på sin adress oavsett var denne än må befinna sig och vilken ip-telefonenhet han eller hon använder för tillfället – förutsatt givetvis att personen har tillgång till Internet. Lika så kan en användare mycket enkelt förflytta sig eller växla till en annan enhet, ställa in vidarebefordring av samtal, konfigurera närvaromeddelanden som kan ses av andra användare, så som ”jag är upptagen, var snäll ring senare” och dylikt. Att nämna alla finesser med Internettelefoni ligger utanför den här rapportens räckvidd, och i synnerhet skulle det ta alltför stor plats. En sak står dock säker, internettelefonin är här för att stanna.

Även om det klassiska telefonnätet i jämförelse kanske framstår som omodernt är det långt i från besegrat av den nya trenden. Det är trots allt fortfarande det dominerande nätet för telefoni, och lär så förbli i flera år framöver. Det har också en stabilitet som än så länge saknar motstycke hos Internet. Telefonnätets stora nackdel, att det inte effektivt utnyttjar överblivna resurser, kan också ses som dess styrka – om än indirekt. Där Internet erbjuder sin ”best effort”-tjänst, och inte garanterar minimum av vare sig fördröjningar eller förluster, är telefonnätet så gott som alltid pålitligt. Att ringa över Internet kan under förhållanden med stor belastning vara besvärligt, med irritationsmoment som avklippta ord och långa pauser av tystnad. Så till dess att Internet har mognat till den grad att det kan erbjuda lika bra tjänstekvalité som telefonnätet, eller åtminstone bli mer allmänt accepterat som ”telefonnät” (något vi ska återkomma till), så är det troligt att det nuvarande telefonnätet får tjäna vidare.

Således har vi två olika nätverk, med vilt skilda arkitekturer och tekniker, som kan användas för att genomföra telefonsamtal. Dels det välanvända, stabila telefonnätet, och dels Internet med den smått revolutionerande internettelefonin på stadig frammarsch. Det är troligt att systemen kommer att leva sida vid sida en lång tid framöver, med varsina förespråkare och användarbaser. För att en ny telefonimetod som internettelefoni ska bli allmänt accepterad är det viktigt att den kan samverka med telefonnätet. Hur varmt hade mobiltelefonen mottagits om man exempelvis inte hade kunnat ringa till det fasta telefonnätet? Den naturliga frågan blir därför om det går att sammankoppla internettelefonin med den sedvanliga telefonin, med andra ord – om det går att ringa via Internet till en telefon kopplad till det vanliga telefonnätet, och vice versa. Och svaret är ja, det går. Och det har gjorts tidigare. Ett särskilt gateway-system, en så kallad VoIP/PSTN-gateway, ansluten både till Internet och till telefonnätet, hanterar samtalsuppkoppling, -hantering och -nedkoppling så att användare i det ena nätet kan nå användare i det andra. Principen visas i Figur 1.1. Gateway-systemet översätter informationen så att telefonisystemen förstår varandra. Det handlar dels om överföring av signalering, så att ett samtal kan kopplas upp från ena sidan till den andra (genom en rad olika kommunikationsprotokoll) och dels om att överföra media så att deltagarna i samtalet faktiskt kan prata med varandra. Detta är ingalunda en trivial uppgift med tanke på tidigare nämnda arkitektoriska skillnader mellan näten, som grundar sig på det faktum att telefonnätet redan från början skapades för att användare skulle kunna genomföra samtal, medan Internet togs fram i mer generella syften. De två nätverken använder även olika protokoll, som måste kunna tolkas och sammankopplas av gateway-systemet.



Figur 1.1 - Principen för att ringa mellan telefonnätet och Internet

Vi har utvecklat en prototyp av ett sådant gateway-system åt vår uppdragsgivare TietoEnator i Karlstad. I den här rapporten presenterar vi arbetet, hur vi gick till väga, vilka problem vi stötte på, hur det resulterande systemet kom att bli, samt vad vi kom fram till för slutsatser i en avslutande utvärdering. Arbetsmomenten kan sägas ha bestått i en inläsningsperiod, för att förstå alla ingående system och tekniker tillräckligt för att kunna utföra uppgiften, samt en utvecklingsperiod då vi implementerade gateway-systemet i fråga. Den teoretiska bakgrunden som är nödvändig för att förstå resten av rapporten läggs till största del fram i kapitel 2, Bakgrund. Här presenteras först principerna bakom telefoni i största allmänhet, för att sedan övergå i en introduktion av telefoni i telefonnätet och genom Internet, respektive. Arkitekturerna beskrivs, och de protokoll som används för signalering gås i genom. Detta är viktigt då det är just dessa protokoll som sedan ska mappas till varandra av gateway-systemet. Även mediaöverföring jämförs, i Internets fall förklaras även kortfattat hur man kan handskas med problem som fördröjningar och dataförlust för att minimera påfrestningen på användarna. Därefter visas väldigt enkelt hur översättning kan fungera i teorin, genom att ställa de olika systemen mot varandra. Vidare beskrivs hur vi genom för ändamålet tillgänglig hårdvara och mjukvara, samt våra egenutvecklade program, kunde skapa en prototyp. Kapitlet avslutas med scenarier för uppringning från ena sidan till den andra, steg för steg genom systemet.

I kapitel 3, Implementation, lägger vi fram de designmodeller vi använt vid utveckling av systemet, samt i generella drag utformandet av programmen för överföring av såväl signalering som media. I kapitel 4, Resultat och utvärdering, presenteras därefter resultaten av

de tester som systemet fått genomgå, för att på så vis ge en fingervisning om vad det är kapabelt till. Både ”simulerade” scenarier (tester i mjukvara) och resultaten från den faktiska provkörningen på målhårdvaran presenteras och utvärderas. I samma kapitel görs även en utvärdering av arbetet i stort, och vi resonerar kring vad som gick bra, vad som gick mindre bra, och vad som kunde ha gjorts annorlunda och inte minst bättre.

Kapitel 5, Sammanfattning, avrundar rapporten och sammanfattar projektet med återblickar och framtida utsikter. Systemet granskas och problem diskuteras, och avslutas med våra egna slutsatser av arbetet.

2 Bakgrund

2.1 Introduktion

Som nämndes i kapitel 1 har en prototyp av en VoIP/PSTN-gateway utvecklats i syfte att möjliggöra telefonsamtal mellan en vanlig telefon, och en så kallad ip-telefon. De två telefontyperna är kopplade till två olika nätverk, vars fysiska och logiska arkitekturer skiljer sig åt. Å ena sidan finns telefonnätet, eller PSTN, som byggts med telefoni i åtanke. Det är ett kretskopplat nätverk där data överförs i bestämda tidsluckor. Å andra sidan finns Internet som är ett paketförmedlande nätverk, och som ursprungligen inte skapades specifikt för överföring av strömmande media, såsom tal.

Syftet med det här kapitlet är att introducera de mekanismer som möjliggör telefonsamtal i de båda nätverken, samt hur de via en VoIP/PSTN-gateway som agerar översättare kan fås att samverka. På så vis skapas en grundläggande förståelse för utvecklingen av ett sådant system - varför det är nödvändigt, hur det kan göras, och vilka följder det får. Kapitlet är relevant då det presenterar idéer som utgör grunden för det utförda experimentet, och som senare i rapporten kommer att användas för att motivera tillvägagångssättet vid implementeringen.

Först introduceras gateway-systemets roll som översättare, och hur det förhåller sig till de två nätverken. Därefter förklaras grundläggande tekniker för att utföra telefonsamtal, såsom signalering och överföring av media – och sambandet dem emellan, rent generellt. Dessutom beskrivs det hur dessa mekanismer realiseras i PSTN, genom att påvisa relevanta protokoll och metoder för dataöverföring. Efter en kort introduktion till VoIP, eller internettelefoni, beskrivs även de protokoll och tekniker som ger motsvarande tjänster möjliga på Internet.

Slutligen beskrivs hur signalerings- och mediatrafik kan översättas mellan de två nätverken. Några redan existerande system som klarar av denna översättning nämns. Detta leder in på hur den aktuella prototypen i fråga var tänkt att fungera, och dess fundamentala komponenter beskrivs, både i form av hårdvara och mjukvara. För att underlätta förståelsen beskrivs två scenarier, dels uppringning från en vanlig telefon till en ip-telefon, från PSTN genom gateway-systemet och till Internet, och dels det motsatta förfarandet, från en ip-telefon till en vanlig telefon.

2.2 Definitioner

I det här avsnittet presenteras de definitioner av termer som används i resten av rapporten.

ANSI – American National Standards Institute är en amerikansk organisation som verkar för att standardisera produkter så att dom passar både den amerikanska och den internationella marknaden. ANSI bildades 1918 av ett antal ingenjörsorganisationer såväl som den amerikanska staten. Mer information finns på [17].

ETSI – European Telecommunications Standards Institute är en organisation bestående av telekommunikationsaktörer på den europeiska marknaden. ETSI har som officiellt uppdrag att standardisera informations- och kommunikationsteknologier i Europa. Mer information finns på [16].

ISDN – Integrated Services Digital Network. ISDN är ett kretskopplat nätverk som är det system som används i telefonisystem (PSTN). Det är ett system som möjliggör digitala sändningar över det koppartrådsnät som traditionellt sett endast kunde användas för rösttrafik.

ITU-T – International Telecommunication Standardization Sector är en organisation inom förenta nationerna som arbetar med att standardisera telekommunikationer världen över. ITU gick innan 1992 under namnet CCITT (Comité Consultatif International Téléphonique et Télégraphique) och bildades ursprungligen för att standardisera telegraftrafiken länder emellan. Mer information finns på [15].

ISUP – Integrated Services Digital Network (ISDN) User Part. Ett protokoll som används mellan switchar i PSTN-nätet för samtalssignalering [1]. Ingår i SS7-stacken.

PCM – Pulse Code Modulation. En digitaliseringsteknik för röst som digitaliserar röst till 64Kbps. Varje röstnivå representeras av ett åtta bitars kodord [6].

PSTN – Public Switched Telephone Network. Det publika telefonnätet genom vilket lokal tillgång till telefontjänster ges [6]. (Termen PSTN används i denna rapport för att benämna det stora nätverk av telefonnät, som innefattar allt från fasta nät, mobila nät till interna nät).

RTCP – RTP Control Protocol. Ett RTP-relaterat protokoll definierat i samma RFC, som tillåter deltagare i en RTP-session att skicka rapporter och statistik till varandra [1].

RTP – Real-Time Transfer Protocol. Ett transportprotokoll i TCP/IP-stacken som används för att transportera strömmande media [5].

SDP – Session Description Protocol. Ett protokoll i TCP/IP-stacken som används för att beskriva multimediasessioner. Sessionsrapporteringar, sessionsinbjudningar och andra former av initieringar av multimediasessioner. [16]

Signaleringsprotokoll - Ett protokoll vars huvudsakliga funktionalitet ligger i att lokalisera ändpunkter, kontakta dessa för att avgöra önskan om att upprätthålla en session, utbyte av mediainformation för att tillåta att sessionen upprätthålls, modifiera en existerande mediasession, samt nedbrytning av existerande mediasessioner [1].

SIP – Session Initiation Protocol. SIP är ett signalerings/kontrollprotokoll på applikationslagernivå i TCP/IP-stacken som kan upprätthålla, modifiera och avsluta multimediasessioner, såsom internettelefonisamtal [3].

SS7 – Signaling System #7. En standard för ”out-of-band”-signalering och vanligt förekommande protokollstack som beskriver hur switchar i SS7-nätverk ska kommunicera signaleringsinformation [6].

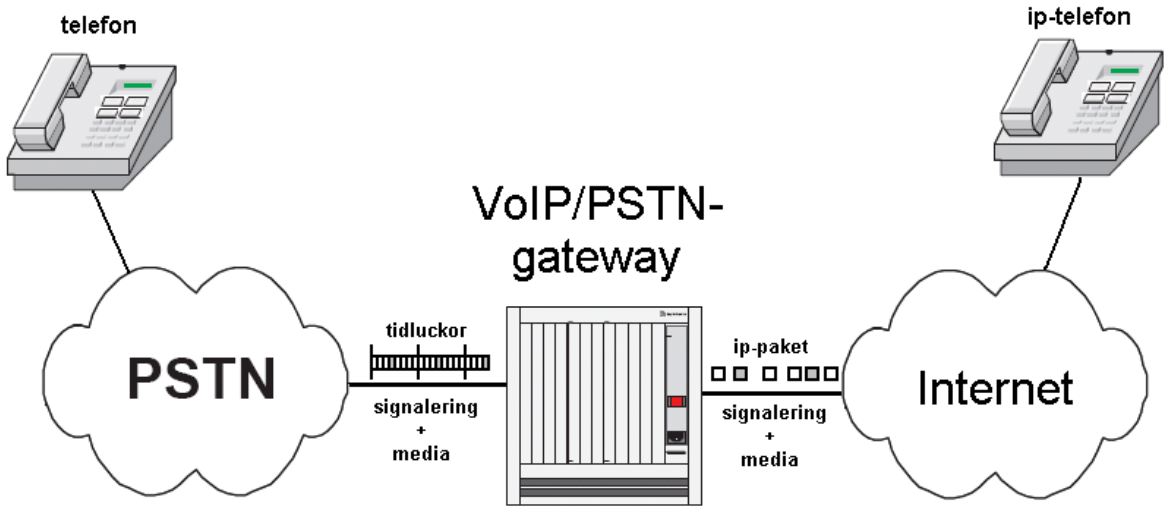
SS7-nätverk. Ett paketförmedlande nätverk baserat på OSI-modellen. Används för signalering i PSTN, och är fysiskt separerat från det kretskopplade nätverket som bär den faktiska rösttrafiken. Varje nod i PSTN är kopplad till båda näten [6].

TDM – Time Division Multiplexing. En teknik där tid divideras till ramar av en fix längd, och varje ram divideras ett fixt antal tidsluckor. När nätverket senare upprättar en uppkoppling över en länk så dedikeras en tidslucka i varje ram till just den kopplingen [5].

VoIP – Voice over IP. Refererar till godtycklig teknik som används för att skicka röst över godtyckligt nätverk som använder IP-protokollet.

2.3 Överblick

Än så länge har beskrivningen av gateway-systemet legat på en ganska hög nivå. Man ska kunna ringa från en ”vanlig” telefon till en ip-telefon, och vice versa. Den vanliga telefonen är kopplad till telefonnätet PSTN, den andra till Internet. (Nätverken har för enkelhetens skull symboliserats med moln i Figur 2.1.) Och mellan dessa nätverk sitter alltså en VoIP/PSTN-gateway som möjliggör tjänsten i fråga.



Figur 2.1 - Gateway-systemets roll och plats

Låt oss börja med att först betrakta nätverken var för sig. För att upprätthålla ett telefonsamtal mellan två ändpunkter i endera nätverket så måste två typer av information kunna utbytas, nämligen signalering (eller kontrollinformation) och röst. Nästa avsnitt beskriver detta mer i detalj, då signalering och mediaöverföring introduceras. Signalerings- och mediatrafik kommer därför att passera genom gateway-systemet när man ringer från det ena nätet till det andra, och det är det som ska översättas. Detta är dock inte helt trivialt då informationen skickas på olika sätt i PSTN och Internet, vilket till stor del beror på att nätverken är så pass olika. PSTN är ett kretskopplat nätverk byggt med telefoni i åtanke och skickar information i så kallade tidsluckor genom tidsmultiplexering, medan Internet är paketförmedlande. I figuren har tidsluckor och paket med signaleringsdata gråmarkerats.

Signaleringsinformationen överförs med hjälp av protokoll. Vi ska senare i kapitlet komma att se att både PSTN och Internet har ändamålsenliga protokollstackar, nämligen Signaling System #7 (SS7) respektive multimediaprotokollen i TCP/IP, som kommer att beskrivas mer

ingående i avsnitten dedikerade till respektive nätverk. Internet har därtill särskilda protokoll för överföring av media vilka vi ska återkomma till. Innan översättningen i gateway-systemet beskrivs är det lämpligt att studera dessa protokoll inom ramen för de två nätverken. Till att börja med beskrivs PSTN då det för de flesta förmodligen känns mer bekant i telefonisammanhang. I synnerhet kommer signaleringsprotokollet ISUP (ISDN User Part) att förklaras. Dessutom beskrivs hur media överförs i nätverket. Därefter presenteras protokoll för Internet som kan användas för att förverkliga ip-telefoni. Vikten läggs på signaleringsprotokollet SIP (Session Initiation Protocol) och sessionsbeskrivningsprotokollet SDP (Session Description Protocol), därutöver visas även mediaöverföringsprotokoll såsom Real-time Transfer Protocol (RTP) och RTP Control Protocol (RTCP). Det är gateway-systemets uppgift att översätta signalerings- och mediainformation. Ur nätverkens perspektiv ser gateway-systemet ut som en vanlig nätverksnod.

2.4 Grundläggande tekniker för telefoni

Låt oss först gå igenom telefonins grunder genom att hålla oss på en abstrakt nivå. Istället för att gå in i detalj på hur ett nätverk för telefoni kan vara uppbyggt, och hur information som skickas i ett sådant nätverk kodas, så fokuserar vi istället på de generella tekniker som förekommer i olika telefonisystem. Förklaringen görs enklast genom ett exempel. Antag att en person ringer till en annan person. För att de ska kunna genomföra ett samtal måste förstås media (läs: röst) kunna överföras över nätverket mellan de två telefonerna, i båda riktningarna. Två telefonister som för ett samtal kan sägas vara delaktiga i en ”mediasession”. Dessutom behövs ett sätt att initiera, hantera och slutligen avsluta denna överföring, och det är här signalering kommer in i bilden. Nätverksnoderna i telefonnätverket kommunicerar signalinformation sinsemellan. Generellt sett kan signalering sägas vara överflyttandet av kontrollinformation för att stödja överflyttandet av något annat, i det här fallet alltså media. Vi ska i detta avsnitt ge en introduktion till signalering och mediaöverföring, och se hur de relaterar till varandra. På så sätt skapas en grundförståelse som är till hjälp när vi undersöker hur de båda teknikerna har förverkligats i PSTN och över Internet.

2.4.1 Signalering

[ITU-T] definierar signalering som ”utbytandet av information (annan än genom röst) specifikt engagerat med upprätthållning, frigöring och annan kontroll av samtal, och nätverksadministration, i automatisk telekommunikationsfunktion”... [19]

Det huvudsakliga syftet med signalering i sedvanliga telefoninätverk är att tillåta överförandet av kontrollinformation mellan noder [19], i samband med uppsättning, övervakning och nedkoppling av samtal och tjänster, databaskommunikation (noder kan skicka databasförfrågningar rörande särskilda tjänster) och nätverksadministration (exempelvis blockering och avblockering av signaleringslänkar). För att utföra signalering används signaleringsprotokoll. De huvudsakliga funktionerna för signaleringsprotokoll är enligt [1] följande:

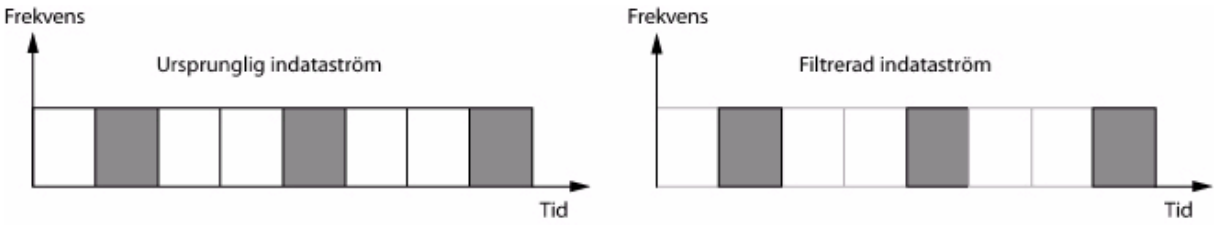
- Lokalisering av en ändpunkt (en telefon)
- Att kontakta ändpunkt för att avgöra dess vilja att etablera en session (överföring av media)
- Utbyte av mediainformation (exempelvis hur röst kodats) för att kunna etablera sessionen
- Modifikation av existerande mediasessioner
- Nedbrytning av existerande mediasessioner

2.4.2 Mediaöverföring

Överföringen av röst kan som synes ske först när en mediasession initierats genom signalering, då det står klart när, var och hur media ska skickas. Idag är det vanligt att röst samplas och digitaliseras innan den skickas ut på telefoninätverket. Fördelarna med detta är främst att digital utrustning är billigare än analog, samt att man slipper problem med brus över längre avstånd. Dock tar bearbetningen av digitala signaler längre tid än för de analoga motsvarigheterna [2]. Det ska påpekas att signalerings- och mediatrafik inte behöver åka längs samma väg genom nätverket. Enligt [19] behöver ”vägen som tas av signaleringstrafiken för en specifik bärartrafik [alltså mediatrafik] inte vara fix”, och signaleringen kan ta många olika vägar för ett och samma samtal.

Efter denna introduktion till viktiga begrepp som signalering, signaleringsprotokoll och mediaöverföring ska vi nu se hur de appliceras i praktiken. Vi börjar med att i nästa avsnitt undersöka det klassiska telefonnätet, PSTN.

2.5 Signalering och mediaöverföring i PSTN

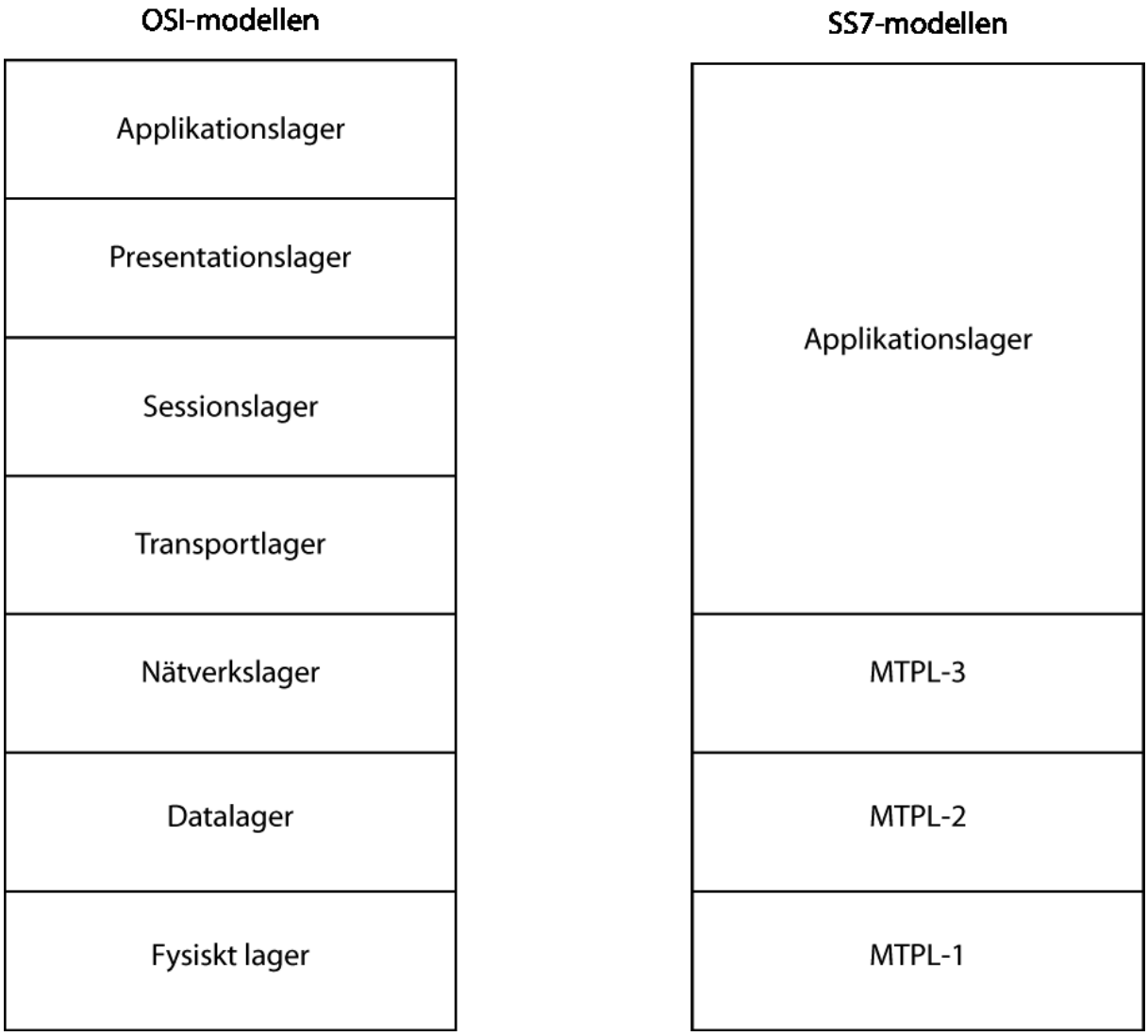


Figur 2.2 - Principen för det TDM-system som används i PSTN

Den information som skickas genom det allmänna telenätet är tidsdivisionsmultiplexerat, och kontrollinformationen och röstinformationen i detta system använder separata tidsluckor. En anledning till att använda tidsdivisionsmultiplexering är att man på så sätt kan öka effektiviteten genom att flera användare kan dela samma kommunikationslina. En nackdel är dock att ingen användare kan skicka en kontinuerlig ström av information, utan måste invänta sin tur för att få skicka information.

Den röstinformation som skickas över telenätet är PCM-kodad (Pulse Code Modulation), en teknik som används för att konvertera analoga signaler till sina digitala motsvarigheter (bortsett från det kvantiseringsfel som introduceras varje gång som en konvertering från en analog signal till en digital signal sker). Denna konvertering sker via en codec (COder-DECoder) [18]. Vår prototyp kom att från PSTN-sidan mottaga digitala signaler som var i så kallat G.711-format. Den codec som användes för detta format använde sig av en semilogaritmisk skala, och detta förfarande kallas ”companded PCM” (från engelskans ”compression expanding”) vilket innebär att signalerna moduleras så att de bättre ska passa (och ge mer detaljer till) människans hörsel. Det finns idag två olika system vars skillnad ligger i antalet samplingsnivåer. A-law har 256 olika nivåer medan μ–law har 255 stycken. Kanada, USA och Japan är de största användarna av μ-law, och A-law används i resten av världen med några undantag [14].

2.5.1 Signaling System #7 (SS7-stacken)



Figur 2.3 - Jämförande bild mellan OSI-modellen och SS7-modellen

Det finns idag flera olika standarder för SS7 (Signalsystem 7), bland annat så har ANSI (American National Standards Institute), ITU (International Telecommunication Union) och ETSI (European Telecommunications Standardization Institute) egna standarder som avviker från varandra mer eller mindre. Prototypen kommer att använda sig av den standard som ITU specificerar då det är ett krav från uppdragsgivaren. Det understa lagret i SS7-stacken är ”Message Transfer Part Layer 1”, detta lager motsvarar det fysiska lagret i OSI-modellen [8]. Det näst lägsta lagret är ”Message Transfer Part Layer 2”. Denna tjänst tillhandahåller feldetektering och felkorrigering, samt sekventiell leverans av alla SS7-paket [8]. Precis som datalagret i OSI-modellen hanterar MTPL-2-lagret överföringar noder emellan och har ingen uppfattning om den slutgiltiga destinationen. Detta lager har också som uppgift att garantera

att en överföring sker. Om ett paket blir deformerat eller tappas bort på vägen till nästa nod så är det MTPL-2-lagret som ansvarar för att återsända paketet tills dess att det överförs utan fel. Om antalet paketfel blir för stort så instrueras MTPL-2 av MTPL-3 att inleda diagnostiska tester.

Det tredje lagret är nätverkslagret, MTPL-3. Nätverkslagret tillhandahåller tre funktioner: routing, meddelandesärskiljning och distribution [8]. Meddelandesärskiljningen avgör om meddelandet är ämnat åt den lokala noden eller inte och skickar meddelandet vidare (till routingfunktionen) om så inte är fallet. Routingfunktion avgör vilken fysisk adress som meddelandet bör skickas till. Distributionsfunktionen anropas ifall meddelandet är ämnat till den lokala noden och dirigerar då meddelandet vidare till den interna användare som meddelandet är adresserat till. Det fjärde och sista lagret är applikationslagret, som består av ett antal olika protokoll, som alla går under namnet användardelar och applikationsdelar [8]. ISUP (ISDN User Part) är ett av de protokoll som finns i applikationslagret.

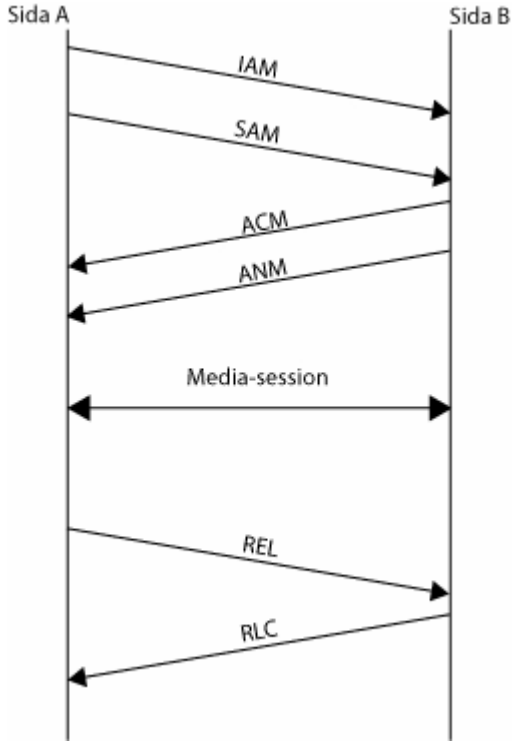
2.5.2 ISDN User Part (ISUP)

ISUP används för att koppla upp och koppla ner kretsar som används för data- eller rösttrafik i det allmänna telenätet (PSTN), och tanken är att ISUP skall vara flexibelt och underlätta kommunikationen mellan det lokala ISDN-nätverket (Integrated Services Digital Network) och det icke-lokala nätverket. Eftersom ISUP är ett protokoll som finns på SS7-stackens applikationsnivå och är ämnat att överföra ISDN-relaterad kontrollinformation så är ISDN-protokollet helt transparent gentemot SS7 då det finns direkta mappningar från ISDN-parametrarna till ISUP-protokollet [8].

ISUP stöder två typer av ISDN-tjänster, de grundläggande och supplementära. Grundläggande tjänster används för att etablera uppkopplingar för kretsar i nätverket. Dessa kretsar används för ljud-, data- och videoöverföringar. Praxis är idag att använda sig av digitala kretsar oavsett vilken typ av information som överförs.

Supplementära tjänster är den funktionalitet som används för att skicka kontrollmeddelanden under ett samtals gång, men används också för mer avancerade procedurer såsom omkoppling av samtal, konferenssamtal etc. Det går också att säga att supplementära tjänster är alla de meddelanden som inte hör till de grundläggande tjänsterna (grundläggande upp- och nedkoppling av samtal).

Det första ISUP-protokollet tillkom 1984 och har sedan dess reviderats flertalet gånger för att kunna tillmötesgå nya tekniska landvinningar inom telekommunikationen. Idag finns ett antal olika standarder för ISUP (ANSI, ETSI, ITU-T etc.) och ett hundratal nationella ISUP-varianter [7]. De primära fördelarna med ISUP är dess hastighet, ökade signalbandbredd, och standardisering av meddelandeväxling [7].



Figur 2.4 - Exempel på en enkel ISUP-session

En session inleds alltid med ett IAM-meddelande (Initial Address Message), som är det meddelande som används för att etablera en koppling över en viss krets i nätverket. I IAM-meddelandet finns information som identifierar bäraren och eventuella specifika krav att beakta i samband med hanterandet av detta samtal. I ett nätverk som följer ITU-T-standarden kan även SAM-meddelanden (Subsequent Address Message) efterfölja IAM-meddelandet (detta är inte specificerat i ANSI-standarden och finns således inte i till exempel amerikanska telefoninätverk) som kan innehålla extra information som krävs för att koppla upp samtalet. Den informationen som skickas kan till exempel vara ytterligare siffror att vidhänga till det telefonnummer som IAM-meddelandet innehöll.

När tillräcklig information för att koppla samtalet har mottagits av den mottagande sidan så skickar den tillbaka ett ACM-meddelande (Address Complete Message) bekräftar att samtalet fortskrider. Det är i detta skede som signaler skickas till destinationen om att ett samtal är på väg att kopplas upp. Redan här kan i vissa situationer ljud överföras från den svarande sidan. När abonnenten på den mottagande sidan svarar på samtalet (lyfter på luren) så skickas samtidigt ett ANM-meddelande (Answer Message) som bekräftar att samtalet fortskrider. Nu börjar rösttrafiken flöda mellan de båda parterna i samtalet (Media-session i Figur 2.4). Det förekommer även att CPG-meddelanden (Call Progress Message) skickas mellan parterna under uppkopplingsfasen såväl som den aktiva fasen av ett samtal. Innebörden av ett CPG-meddelande är olika beroende på situationen, och används för att informera motparten om viktiga händelser. I vårt fall kommer ett CPG-meddelande oftast innehålla information om progress hos motparten, eftersom vi utvecklar en prototyp och således ej berör alla möjliga indikationer som ett CPG-meddelande kan förmedla.

När samtalet sedan avslutas när någon lägger på luren så skickar den avslutande delen ett REL-meddelande (Release Message) som innebär att samtalet är i färd med att avslutas. Den andra sidan i samtalet bekräftar REL-meddelandet genom att skicka ett RLC-meddelande (Release Complete Message), och de resurser som samtalet har tagit i anspråk kan frigöras för andra ändamål. I REL-meddelandet finns även orsakskoder som ger information om anledningen till varför parten vill avsluta samtalet. Exempel på en sådan orsakskod kan vara när en part ringer en annan som är upptagen (till exempel med ett annat samtal), och den uppringda användaren skickar ett REL-meddelande med en orsakskod som indikerar att användaren är upptagen.

I enlighet med önskemål från uppdragsgivaren skulle prototypen använda sig av den ISUP-standard som specificerades av ITU-T.

2.6 Signalering och mediaöverföring på Internet

2.6.1 Voice over IP (VoIP)

Trots det något förvirrande ordvalet VoIP/PSTN-gateway är VoIP till skillnad från PSTN inte något nätverk. Enligt [6] refererar termen istället till ”godtycklig teknik som används för

att överföra röst över godtyckligt nätverk som använder IP-protokollet”, och man nämner topologier som modembaserad punkt-till-punkt-förbindelse, lokala nätverk, intranät och Internet. Det är i synnerhet det sistnämnda fallet som är intressant här, och [5] väljer att kalla realtidsinteraktivt ljud över Internet för ”Internet phone” (internettelefoni på svenska), med motivationen att från användarnas perspektiv är tjänsten som erbjuds snarlik den som ges av PSTN.

Internettelefoni är egentligen ingen ny idé, vilket man kan läsa om i [5]. Prototyper togs fram på 80-talet, och forskningen därkring är ännu äldre. Under 90-talet skapades internettelefoniprodukter för användning mellan PC-datorer av en mängd företag. Även applikationer som gjorde det möjligt att ringa från en PC till en ”vanlig” telefon blev populära vid samma tidpunkt. Dessa tjänster fanns nämligen tillgängliga till en låg kostnad. Tekniken bakom fenomenet var en följd av användandet av multimediaprotokoll för Internet och särskilda gateway-system, vilka är viktiga delar av temat för den här rapporten. Ett annat applikationsområde var att ringa mellan ”vanliga” telefoner över Internet, vilket gav billiga samtalskostnader över långa distanser.

Det finns dock nackdelar. Precis som andra multimediaapplikationer är internettelefoni känslig för fördröjningar. Fördröjningen ska för röst vara mellan 150 och 400 millisekunder för att vara acceptabel [5]. I PSTN är fördröjningarna vanligtvis mellan 50 till 70 millisekunder menar [6], och förklarar vidare att alltför låg bandbredd vid röstöverföring på Internet kan resultera i att röster försvinner och ord ”klipps av”. Givet ”best effort”-tjänsten som IP erbjuder är detta alltså något man som användare av internettelefoni kan få erfara. I gengäld klarar ip-telefoni av en del förluster. (Detta kan ställas i kontrast till andra mer klassiska applikationer, såsom filöverföring, där förhållandena är motsatta.) Mellan 2 och 20 procent förluster kan tolereras enligt [5], beroende på val av röstkodning och -överföring, samt hur förlusterna döljs hos mottagaren.

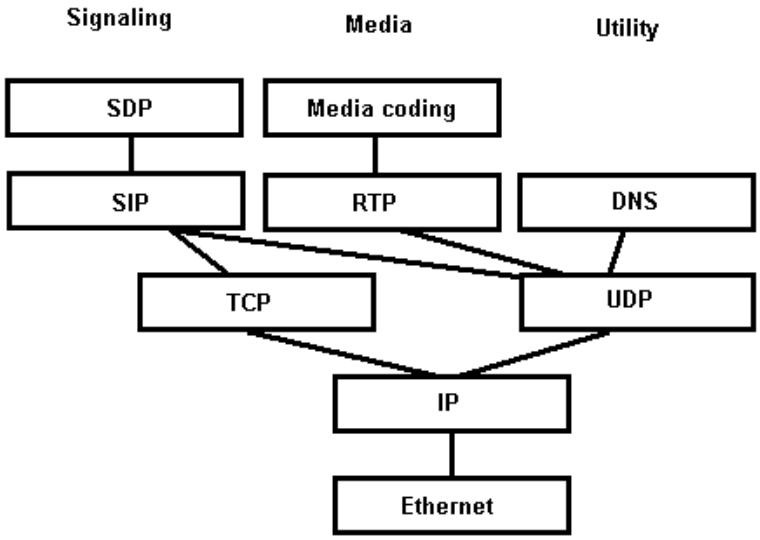
2.6.2 Session Initiation Protocol (SIP)

I RFC 3261 [3] beskrivs Session Initiation Protocol (SIP) som “ett signaleringsprotokoll på applikationslagernivå för att skapa, modifiera och avsluta sessioner med en eller flera deltagare”. Som exempel på en sådan session ges internet-telefonsamtal. Detta avsnitt innehåller en mycket kortfattad beskrivning av protokollet i fråga inom ramen för vad som är relevant för att senare kunna förstå relationen till ISUP (avsnitt 2.5.2), och därigenom

problemet med att låta gateway-systemet översätta mellan de båda signaleringsprotokollen. För en mer utförlig introduktion av SIP hänvisas läsaren till [3].

I [1] kan man läsa om historien bakom SIP-protokollet och anledningen till dess likheter med andra protokoll. Protokollet skapades ursprungligen av IETF Multi-Party Multimedia Session Control Working Group (MMUSIC). Version 1.0 släpptes som Internet-Draft år 1997, och efterföljande version 2.0 publicerades som RFC två år senare. Protokollet hämtar inspiration från två de två välkända internet-protokollen HTTP (Hyper Text Transport Protocol) för webben och SMTP (Simple Mail Transfer Protocol) för e-mail. Från det förstnämnda känner man igen klientserver-designen och användandet av URIs (Unified Resource Identifiers), och från SMTP har man lånat ett schema för textkodning och header-stil. I enlighet med filosofin "ett problem, ett protokoll" används SIP till just bara signalering, och använder i sin tur andra protokoll för transport (TCP och UDP), mediatransport (RTP) och beskrivning av media (SDP).

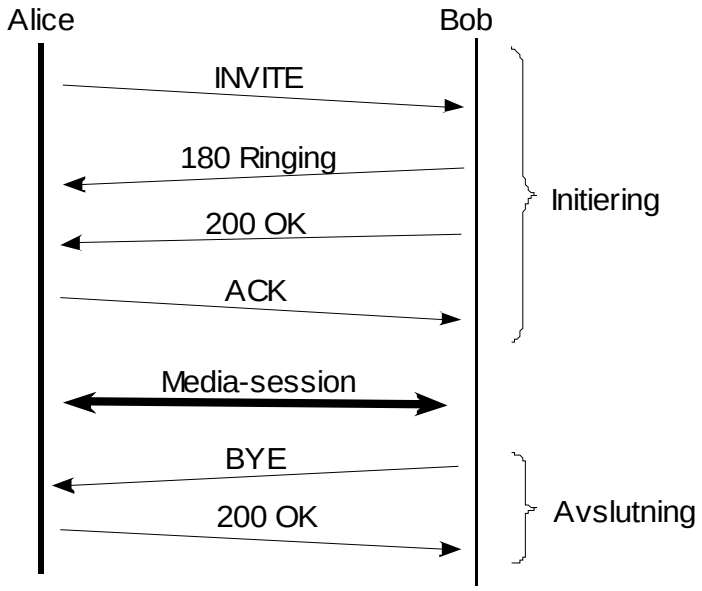
Figur 2.5 ger en förenklad bild av den multimediaprotokollstack som presenteras i [1], och innehåller bara de protokoll som är relevant för det aktuella problemområdet. Den ger en överblick i hur SIP och andra protokoll som diskuteras i de senare sektionerna relaterar till varandra.



Figur 2.5 - Ett urval ur Internets protokollsamling för multimedia

En SIP-klient (ett användarprogram) kan således koppla upp sig mot en SIP-server (ett annat användarprogram) och skapa och hantera en session genom att utbyta förfrågningar och svar. I Figur 2.6 ges ett förenklat exempel (notera att förhandling av sessionsparametrar för att

upprätthålla mediasessionen inte framgår). Man kan dela in flödesdiagrammet i tre tydliga faser, initiering av mediasession, pågående mediasession, och avslut av mediasession.



Figur 2.6 - Ett enkelt SIP-exempel

Antag två användare, Alice och Bob, som sitter med varsin SIP-kapabel enhet. I figuren ovan skickar Alice en förfrågan till Bob där hon indikerar att hon vill påbörja en samtalssession. När SIP-telefonen hos Bob börjar ringa (därav “ringing”) skickas ett svar till Alice som indikerar detta. När Bob sedan lyfter luren på telefonen signaleras det att han har svarat. Alice i sin tur svarar med en slutgiltig bekräftelse när hon mottar svaret. Mediasessionen kan sedan påbörjas, i enlighet med information i header-fält i de förfrågningar och svar som skickades under initieringen (förhandling om vilken codec som används, portnummer mm). För att skicka mediatrafiken används ett annat protokoll, exempelvis RTP. Bob lägger på luren, vilket genererar en BYE-förfrågan till Alice, som sedan besvarar denna. Därmed är samtalet avslutat.

Som synes kan båda sidor skicka förfrågningar. Detta betyder att vem som är klient och vem som är server varierar beroende på hur utbytandet av meddelanden sker. I exemplet ovan agerar Bob server under initieringen, men eftersom han kopplar ner samtalet genom att skicka en BYE-förfrågan blir rollerna ombytta. Både SIP-förfrågningar och -svar är textbaserade. De har samma uppbyggnad, och innehåller ett antal header-fält och en eventuell meddelandekropp. Härnäst ses innehållet i det exemplifierade INVITE-meddelande som Alice skickar till Bob:

```
INVITE sip:bob@bobsdomain.com SIP/2.0
Via: SIP/2.0/UDP sipdevice@alicedomain.com
Max-Forwards: 70
To: Bob <sip:bob@bobsdomain.com>
From: Alice <sip:alice@alicedomain.com>
Call-ID: a84b4c76e66710@sipdevice.alicedomain.com
CSeq: 314159 INVITE
Content-Type: application/sdp
Content-Length: 142

(SDP-last som anger mediasessionsparametrar)
```

Den första raden kallas “start line”, och listar metoden (i detta fall INVITE), den efterfrågade SIP-adressen (“Request-URI”), och versionen av SIP som används - alla separerade med mellanslag. Raderna avslutas med carriage-return line-feed. Nästkommande header är Via. Varje SIP-enhet som skickar eller vidarebefordrar ett meddelande stämplar sin egen adress här. Även SIP-versionen och transportprotokoll (i detta fall UDP) tas med. To och From visar ursprung och mål för meddelandet. SIP-URL inom större-än-mindre-än-tecken används för att routa meddelandet. Call-ID ser ut som en mail-adress, men är i själva verket en identifierare för att hålla reda på en specifik SIP-session, och är globalt unikt. Detta används av båda parter för att identifiera ett specifikt samtal då de i praktiken kan ha flera samtal mellan sig. CSeq, för “command sequence”, innehåller ett nummer följt av aktuell metod. Detta nummer inkrementeras för varje ny förfrågan som skickas. Via, To, From, Call-ID, CSeq och Max-Forwards är obligatoriska i alla SIP-meddelanden. Utöver dessa finns andra headrar som kan inkluderas som valfri extra information, eller information som behövs vid en specifik typ av förfrågan. I exemplet kan Contact användas för att routa direkt till Alice, och slutligen indikerar Conent-Type och Content-Length att meddelandekroppen är SDP av längd 142 bytes.

I RFC 3261 [3] förklaras det att detaljerna kring mediasessionen, såsom typen av media, codec eller samplingshastighet inte beskrivs med SIP. Istället innehåller SIP-meddelandets kropp information som beskriver detta, enligt något annat protokollformat, exempelvis SDP (”SDP-last” i exemplet). En liknelse görs med hur en webbsida bärs av ett HTTP-meddelande.

Här näst ges korta beskrivningar av de SIP-förfrågningar som är väsentliga för uppgiften, baserat på informationen som ges i [1].

- INVITE

Denna metod används för att upprätthålla en mediasession mellan två användare. Den har vanligtvis en meddelandekropp med mediainformation om avsändaren. En användarklient som skickar ett INVITE-meddelande skapar ett globalt unikt Call-ID som används under samtalets pågående.

- ACK

ACK-metoden används för att bekräfta (“acknowledge”) slutliga svar på INVITE-förfrågningar. Slutliga svar på alla andra förfrågningar (se nedan) ACK:as inte. En ACK bör bara innehålla en application/sdp-meddelandekropp såvida en sådan inte var med i INVITE-metoden.

- PRACK

Används för att bekräfta mottagning av så kallade provisoriska svar (se svarsklass 1xx i tabell 2.1). Den används då provisoriska svar är kritiska för att bestämma samtalets tillstånd. Ett sådant fall är sammankoppling med PSTN. PRACK används där för att bekräfta mottagandet av ett speciellt svar som indikerar framstegen vid initiering av en mediasession, och är egentligen en extension till SIP-protokollet. Kan innehålla en meddelandekropp.

- OPTIONS

OPTIONS-metoden används för att fråga ett användarprogram eller server om dess kompetens och för att upptäcka dess nuvarande tillgänglighet. Exempelvis vilka metoder som stöds.

- INFO

Används av ett användarprogram för att skicka samtalssignaleringsinformation till ett annat användarprogam som det redan har etablerat en mediasession med. Innehåller vanligtvis en meddelandekropp. Innehållet kan vara signaleringsinformation, en händelse mitt i samtalet, eller någon form av “stimulans”. INFO-metoden är intressant

då även den är en extension som tagits fram med PSTN-samverkan i åtanke, se kapitlet om att sammankoppla SIP och ISUP för mer information.

- BYE

Används för att avbryta en etablerad mediasession. Denna metod har ingen meddelandekropp

- CANCEL

Används för att avbryta ett uppringningsförsök. Inte heller denna metod har någon kropp.

En server som tar emot någon av ovanstående förfrågningar skickar ett SIP-svarsmeddelande till klienten. Ett SIP-svar även det innehålla olika header-fält. Tabell 2.1 är inspirerad från en motsvarighet i [1], och visar att det finns 6 klasser för SIP-svar. De fem första lånades från HTTP; medan den sjätte skapades specifikt för SIP [1].

Klass	***Beskrivning***
1xx	Informational/Provisional: Indikerar initieringens status innan mediasessionen är upprätthållen. Exempel: 180 Ringing (det ”ringer” hos servern)
2xx	Success: förfrågan har lyckats. Exempel: 200 OK vid uppkoppling som svar på INVITE (servern har svarat)

Klass	*Beskrivning*
3xx	Redirection: servern har returnerat adressen till en annan server där användaren kan hittas. Exempel: 302 Moved temporarily
4xx	Client error: förfrågan misslyckades på grund av ett fel orsakat av klienten. Exempel: 404 Not Found, adressen leder inte till någon användare
5xx	Server failure: förfrågan misslyckades på grund av ett fel orsakat av servern. Exempel: 502 Bad Gateway
6xx	Global failure: förfrågan har misslyckats. Exempel: 600 Busy Everywhere

Tabell 2.1 - Klasser av SIP-svar

Denna korta introduktion till SIP har varit väldigt avskalad. I själva verket finns det fler viktiga egenskaper som gör protokollet intressant. Nämnas kan t.ex. olika typer av servrar (inte att förknippa med servrar i sessionssammanhang) som erbjuder tjänster för att ändsystemen ska kunna hitta varandra. Proxyservrar och omdiringeringssevrar är viktiga i de

fall då klienten som vill ringa bara känner till mottagarens SIP-adress, och inte dess faktiska IP-nummer. Dessutom finns särskilda registreringssevrar där användare kan registrera sig, så att denna mappning mellan SIP-adresser och IP-nummer fungerar. SIP-servrar har vanliga domännamn, och hittas som sig bör genom DNS-namnuppslagning.

2.6.3 Session Description Protocol (SDP)

SIP-protokollet beskriver som sagt inte själva mediasessionerna, utan används bara för att initiera och koppla ner sådana. För att kunna utbyta mediatrafik behöver vi bland annat kunna specificera hur den ser ut och hur den ska överföras. Som framgick i beskrivningen av SIP kan Session Description Protocol (SDP) användas till detta. SDP-beskrivningar skickas helt enkelt som last i SIP-meddelandekroppar. [2] skriver att:

Session Description Protocol (SDP) [RFC 2327] specificerar hur informationen som krävs för att beskriva en session ska kodas. SDP inkluderar inte någon form av transportmekanism eller någon form av förhandling av parametrar. En SDP-beskrivning är helt enkelt en samling information som ett system kan använda för att ansluta sig till en mediasession. Den inkluderar, exempelvis, IP-adresser, portnummer, och tider och datum när sessionen är aktiv.

Exempelvis skulle SDP-lasten i det INVITE-meddelande som Alice skickar till Bob i sektionen om SIP kunna se ut så här:

```
v=0
o=Alice 1234567890 1234567890 IN IP4 sipdevice.alicedomain.com
s=Phone Call
c=IN IP4 100.110.120.130
t=0 0
m=audio 62331 RTP/AVP 0
a=rtpmap:0 PCMA/8000
```

Som synes är SDP i likhet med SIP textbaserat, och en beskrivning består av ett antal rader på formen:

```
Typ=värde
```

Utan att gå in alltför mycket på detaljer och beskriva varje, till synes kryptisk rad fokuserar vi istället på det mest relevanta Bob kan utläsa av ovanstående, som att Alice IP-adress är 100.110.120.130, att mediaformatet är ljud (audio), att portnumret som Bob uppges skicka media till är 62331, att mediatransportprotokollet är RTP, att kodningen som Bob ska använda för att koda ljudet är PCM a-law vid samplingshastigheten 8 kHz. Bob kan sedan ange sin motsvarighet genom att inkludera SDP-beskrivningar i sin 200 OK-bekräftelse. Om Alice därefter inte accepterar Bobs förslag så skickar hon en ACK följt av en BYE, så att sessionen avslutas [1].

2.6.4 Real-Time Transfer Protocol (RTP) och RTP Control Protocol (RTCP)

För att kunna förmedla rösttrafik kom prototypen att använda sig av RTP- och RTCP-protokollen [12], (Real-time Transport Protocol respektive RTP Control Protocol). Ett problem som uppstår i samband med överföring av realtidsmedia över ett paketväxlat nätverk är att man inte kan garantera kvaliteten på överföringen. Realtidsmedia ställer höga krav på överföringshastighet, responstid och stabilitet för att fungera tillfredsställande. Om något av dessa behov inte uppfylldes så skulle det märkas för de parter som var involverade i samtalet, och i värsta fall skulle samtalet bli omöjligt att genomföra.

RTP är ett protokoll för att överföra realtidsmultimedia men är på inget vis begränsat till endast detta. RTP stöder unicasting och har möjlighet för multicasting, men multicasting stöds bara om underliggande lager stöder det. RTP är designat för att fungera oavsett hur underliggande transport- och nätverkslager är konfigurerade, men används oftast ovanpå UDP (User Datagram Protocol), dock inte i prototypen. RTP är inte i sig självt väl lämpat för mediaöverföring i realtid då protokollet inte kan garantera att meddelanden kommer fram eller att de kommer fram i rätt sekvens. För att administrera RTP och säkerställa en viss kvalitetsnivå så används RTCP.

RTCP används för att förmedla information om kvalitet på sändningen, deltagarinformation (namn, e-mailadresser etc.) och statistik för RTP-sessionen. Denna dialog deltagare emellan sker kontinuerligt under en RTP-session och genom dessa meddelanden justeras nödvändiga parametrar för överföring av multimedia. Dessa meddelanden skickas av alla deltagare i en session för att säkerställa kvaliteten på sändningen genom att kontinuerligt

rapportera statistik och sändnings- och mottagningsmöjligheter för samtliga inblandade. Aktiva sändare skickar sändarrapporter med statistik över överföring och mottagning, mottagare skickar mottagarrapporter med mottagningsinformation.

Det finns olika typer av RTCP-rapporter som skickas över nätverket i under en RTCP-session [13]:

- Sender Report (SR), for transmission and reception statistics from participants who are not active senders
- Receiver Report (RR), for reception statistics from participants who are not active senders
- Source Description (SDES) items, including CNAME
- BYE, for indicating end of participation
- APP, for application specific functions.

2.7 VoIP/PSTN-gateway

Nu när mekanismerna för signalering och mediaöverföring presenterats för PSTN och Internet är det dags att undersöka deras möjlighet att samverka.

2.7.1 Existerande system

Det primära området för gateway-prototypen var som bekant att konvertera mellan SIP och ISUP-protokollen, och kontrollera mediaöverföringen. Detta medför dock inte att den prototyp som denna uppsats handlar om kan jämföras med andra tjänster och produkter som tillhandahåller tjänster för kommunikation mellan VoIP och PSTN, exempelvis Skype [10] och Google-Talk [11], eftersom dessa visserligen har stöd för ISUP/SIP-konvertering men har avancerade grafiska gränssnitt och stor ytterligare funktionalitet utöver signalkonvertering. Prototypen skulle endast konvertera signaler mellan SIP och ISUP.

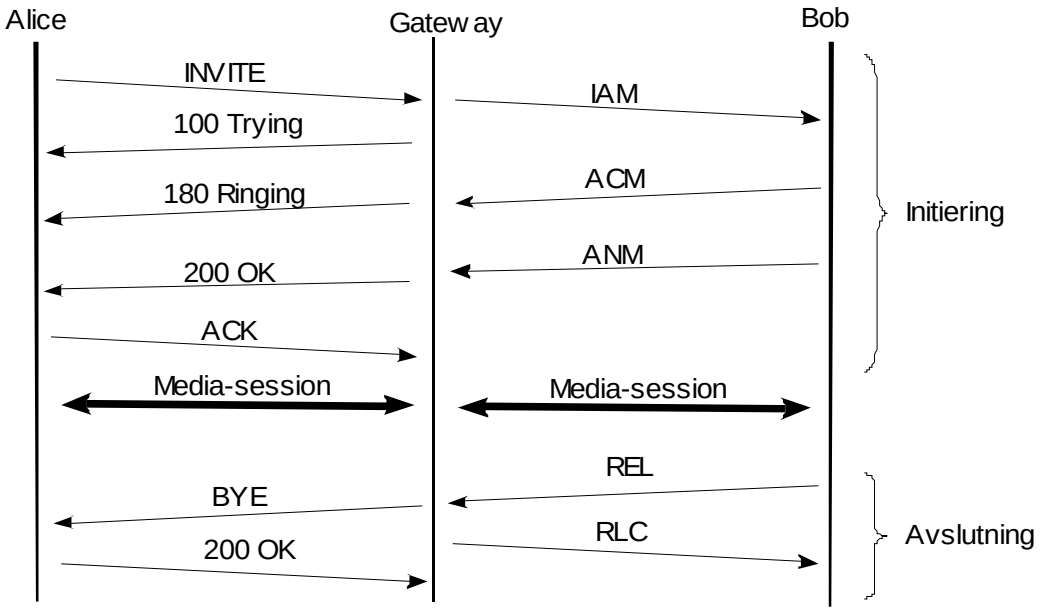
Det har tidigare gjorts prototyper för konvertering av signaler mellan SIP och ISUP [9]. Denna prototyp implementerade dock aldrig uppkoppling från SIP-sidan till ISUP-sidan, och har inga möjligheter att överföra ljud från den ena sidan till den andra. Målsättningen för

prototypens funktionalitet var att kunna genomföra samtal från båda sidor såväl som att kunna överföra rösttrafik mellan båda sidor.

2.7.2 Att sammankalla SIP och ISUP

När de två berörda signaleringsprotokollen SIP (medelst SDP) och ISUP har presenterats är det dags att beskriva deras samverkan, samt hur media kan flöda mellan nätverken de ingår i. Givet att båda protokollen baseras på förfrågningar och svar kan man kanske intuitivt ana sig till en del. En ISUP-förfrågan översätts till motsvarande SIP-förfrågan av gateway-systemet, och dess svar översätts tillbaka från SIP till ISUP.

I [2] beskrivs gateway-systemets roll i de båda nätverken. Den agerar ändpunkt i SIP-nätverket, och nätverksnod i PSTN-nätet. Med denna mekanism påverkas inte SIP-arkitekturen på något sätt, menar man, då gateway-systemet helt enkelt ser ut som ett vanligt användarprogram. Man beskriver dessutom andra fördelar som att gateway-system låter SIP bibehålla alla sina goda egenskaper vid ihopkopplingen med PSTN. “En generell regel för SIP-samverkan med andra system är att se till att SIP inte behöver ändra sitt beteende för att göra så. Det är alltid bättre att bygga ett komplext gateway-system vid kanten för protokollkonversation än att klämma in mer komplexitet i protokollen själva”, avslutar man [2]. [1] förklarar också en viktig skillnad mellan vanliga SIP-användarprogram och gateway-system, nämligen antalet tillåtna användare: “medan ett användarprogram vanligtvis stöder en enda användare, så kan ett gateway-system stödja hundratusentals användare”.



Figur 2.7 - Förenklad bild av mappningen mellan SIP och ISUP

Figur 2.7 visar ett förenklat exempel på hur mappningen kan se ut (se bilaga A för ett mer detaljerat exempel). RFC 3398 [4] fokuserar i detalj på översättningen av ISUP-meddelanden till SIP-meddelanden, och mappningen av ISUP-parametrar till SIP-headrar, och ger bara en SIP-mappning för de ISUP-parametrar som kan användas av mellanhänder i routningen av SIP-förfrågningar. Dessutom ges en lista på de SIP-mekanismer som behövs för att mappningen ska fungera, inklusive några som inte ingår i basspecifikationen för protokollet, så kallade “extensioner”. Om SIP-enheten involverad i samtalet inte stödjer dem är det fortfarande möjligt att genomföra ett samtal, även om beteendet vid etableringen av samtalet kan vara ett annat än det som förväntas av användaren. Som exempel ges att någon lyfter luren innan ringsignalen hörs, eller att använder inte informeras att samtalet vidarebefordras. De SIP-extensioner som tagits fram i syfte att möjliggöra SIP-ISUP-mappning beskrivs i RFC 3398 [4], och några1 av dem är relevanta för uppgiften i fråga och tas därför upp här. De tidigare nämnda SIP-metoderna INFO och PRACK (se avsnitt 2.6.2) är två exempel. INFO-metoden används då det finns ISUP-meddelanden som inte mappar till något SIP-meddelande. INFO-meddelanden har helt enkelt motsvarande information i sina meddelandekroppar. PRACK används för att få till stånd pålitlig överföring av provisoriska svar, så att gateway-systemet kan hålla reda på vilket tillstånd samtalet befinner sig i [1].

Hur mappningen av meddelanden sker mellan SIP och ISUP är ganska rättfram, och beskrivs ingående i RFC-dokumentet [4]. Där finns flödesdiagram liknande det ovan som

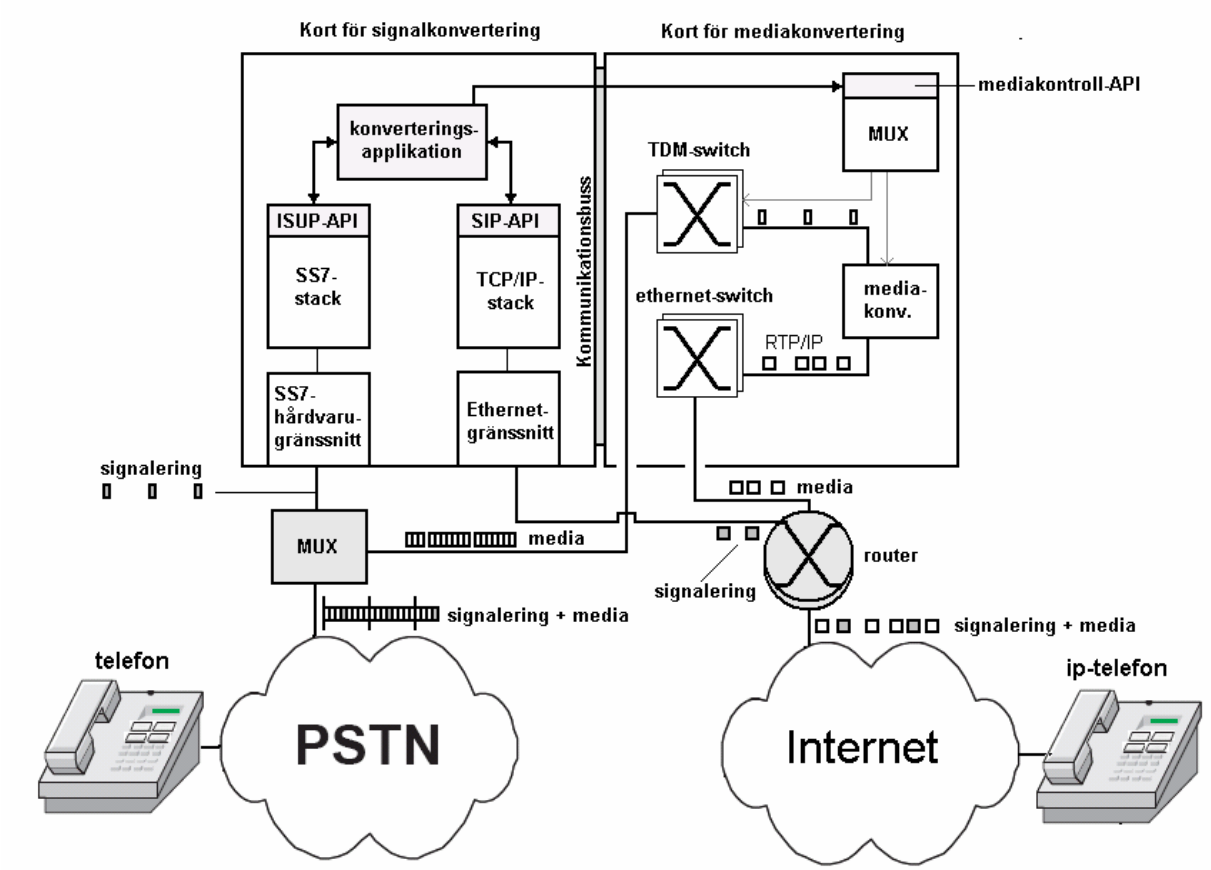
visualiserar förloppet i olika fall. Dessutom specificeras en mappning mellan statuskoder för de båda protokollen, exempelvis “1 Alerting” på ISUP-sidan till SIP-motsvarigheten “180 Ringing”.

2.7.3 Konvertering av media

Då röst som överförs på PSTN-nätet är PCM kodad är det enklaste förfarandet att låta gateway-systemet ta informationen i en specifik tidslucka, och skicka den som last i ett RTP-paket (där det framgår att en PCM-codec använts) till rätt avsändare i enlighet med vad som bestämdes under signaleringen vid upprätthållandet av mediasessioen. På motsvarande sätt tas informationen i ett inkommande RTP-paket och skickas ut på rätt tidslucka när röst överförs från Internet till PSTN. Som nämndes i avsnittet om VoIP finns det risk för paketfördröjningar på Internet, och dessa fördröjningar kan variera (så kallat ”jitter”). Dessutom finns förstås problemet med paketförluster. Detta kan lösas på olika sätt, som redogörs i [5]. Genom att buffra mottagna paket innan de skickas ut på rätt tidsluckor kan man slippa avklippta ord, men på bekostnad av fördröjning. Denna ”uppspelningsfördröjning” kan lämpligen anpassas efter de rådande förhållandena på nätverket, så att man får en bra avvägning. Paketförluster kan hanteras genom att exempelvis spela upp senast mottagna paket ännu en gång.

Vår prototyp av en VoIP/PSTN-gateway var tänkt att innehålla hårdvara som skulle sköta denna hantering av multimediaöverföring åt oss, varför vi inte går djupare in i ämnet.

2.8 Beskrivning av gateway-system



Figur 2.8 - Överblick av gateway-systemet

2.8.1 Överföring av samtalsinformation från PSTN- till SIP-telefon

2.8.1.1 Signalinformationens väg

Signalen kommer in från PSTN-sidan (vänstra molnet i figuren), och in i den MUX-enhet (multiplexer) som separerar kontrollinformationen och röstinformationen. De tidsluckor som innehåller kontrollinformation går till SS7-hårdvarugränssnittet för vidare behandling. Denna information går sedan upp genom SS7-stacken och slutligen till det ISUP-API som finns. Detta API interagerar sedan gateway-applikationen mot, och beroende på vad som tas emot från ISUP- eller SIP-API:et så genomgår gateway-applikationen en tillståndstransition för att kunna hantera samtalet på ett korrekt sätt. ISUP-informationen som gateway-applikationen

mottar ger ibland (beroende på vilken ISUP-primitiv som mottogs) även upphov till interaktion med SIP-API:et (till exempel ett mottaget IAM-meddelande ger upphov till ett skickat INVITE-meddelande). I dessa fall så kommer gateway-applikationen att arbeta mot SIP-API:et, som i sin tur genererar de primitiver som behövs och skickar dessa via den TCP/IP-stack som API:et har. I samband med detta så är alltså signaleringsinformationen omformaterad till ip-paket. Gateway-applikationen kommer också att styra den MUX som finns på mediakonverteringsdelen för att ge rätt tidslucka till rätt samtal. TCP/IP-stacken är sedan kopplad till ett ethernet-gränssnitt som skickar signaleringsinformationen till en router i Internet som sedan förmedlar dessa ip-paket till ip-telefonen i andra änden.

2.8.1.2 Röstinformationens väg

Röstinformationen kommer precis som signaleringsinformationen in i den MUX som då dirigerar om röstinformationens tidsluckor till mediakonverteringsdelens TDM-switch. Denna växel separerar individuella tidsluckor och dirigerar dessa till mediakonverteraren. Vilka tidsluckor som TDM-switchen bryter ut och skickar vidare beror på den inställning som den av gateway-applikationen styrda MUX:en har. Dessa tidsluckor går sedan vidare in till mediakonverteraren, som omvandlar dessa tidsluckor till RTP-information formaterade som ip-paket. Dessa ip-paket skickas sedan vidare till en router som förmedlar dessa paket till ip-telefonen i andra änden.

2.8.2 Överföring av samtalsinformation från SIP- till PSTN-telefon

2.8.2.1 Kontrollinformationens väg

Den information som kommer ifrån ip-telefonen separeras i en router och signaleringsinformationen går via ethernet-gränssnittet till den TCP/IP-stack som SIP-API:et arbetar mot. Detta API kommer sedan att informera gateway-applikationen och möjliggöra för denna att vidtaga nödvändiga åtgärder. Gateway-applikationen kan (som nämnts ovan) på grund av denna information övergå från ett tillstånd till ett annat, vilket är tvunget för att kunna hantera ett samtals olika faser. Gateway-applikationen kan som resultat av denna information också anropa funktioner i ISUP-API:et som i sin tur genererar primitiver som

sedan skickas ned via hårdvarugränssnittet till den MUX som sammanför denna information med röstinformationen, i form av tidsluckor. Dessa tidsluckor kommer sedan att nå PSTN-telefonen.

2.8.2.2 Röstinformationens väg

Röstinformationen (i form av ip-paket) kommer från ip-telefonen (via Internet) till den router som sedan dirigerar om dessa ip-paket. Denna router dirigerar sedan röstinformationen till den ethernet-switch som finns på mediakonverteringskortet. Denna switch kommer sedan att skicka ip-paketen till mediakonverteraren, som i sin tur kommer att översätta dessa ip-paket till tidsluckor. De tidsluckor som genereras av mediakonverteraren kommer sedan att gå vidare till TDM-switchen som kommer att integrera dessa tidsluckor i rätt sekvens, som i sin tur går vidare till den MUX som sammanför signaleringsinformationen med röstinformationen. Till sist når denna information PSTN-telefonen.

2.9 Sammanfattning

I detta kapitel har de tekniker som behövs för att implementera en VoIP/PSTN-gateway introducerats. Efter en inledande förklaring av PSTN-nätet, med fokus på signalering genom SS7 och i synnerhet ISUP-protokollet, så presenteras viktiga protokoll i multimediaprotokollstacken för Internet, och hur dessa tillsammans samverkar för att möjliggöra multimediatjänster till användare. En sådan tjänst är internettelefoni, eller VoIP, som för en slutanvändare påminner om den funktionalitet det kretskopplade telefonsystemet erbjuder. Med detta menas möjligheten att ringa till en annan användare, att föra ett samtal och slutligen koppla ned – över Internet. Utöver det finns andra tjänster som revolutionerar den sedvanliga telefonin.

SIP-protokollet är ett lättviktigt signaleringsprotokoll för Internet, som lämpar sig väl för just internetsamtal. Det kan jämföras med ISUP. För att specificera detaljer kring mediasessionen används SDP-meddelanden, som kan skickas som last i SIP-meddelanden. SIP sköter inte alls mediaöverföringen, utan förlitar sig på att andra protokoll, såsom RTP och RTCP utför detta.

Att skicka relaltidsinteraktiv media över Internet är inte helt trivialt. Fördröjningar i det paketförmedlande nätverket sänker kvalitén på tjänsten vilket förstås även påverkar användarna, och kan i dagsläget inte undvikas även om det kan hanteras till viss del genom bufferthantering och interpolering (eng. interpolation) [5].

Att kunna mappa SIP till ISUP (signaleringsöversättning), och skicka media mellan PSTN och Internet (mediaöversättning), möjliggör sammankoppling av de båda telefonisystemen genom ett gateway-system. Det får som följd att användare av de två näten kan ringa till varandra, med fördelar såsom lägre kostnader som följd. När vi nu har sett teknikerna och förstått problemområdet är det dags att beskriva hur vi försökte implementera en prototyp av ett sådant gateway-system.

3 Implementering

3.1 Introduktion

I det här kapitlet beskrivs implementeringen av konverteringsapplikationen som mappar protokollmeddelanden mellan ISUP och SIP, samt mediakonverteraren som möjliggör att rösttrafiken kan överföras mellan tidsluckor och RTP-paket. Syftet med kapitlet är dels att ge läsaren en djupare förståelse för problemet att implementera de två modulerna, och samtidigt förklara vilka lösningar som finns för att utföra uppgiften.

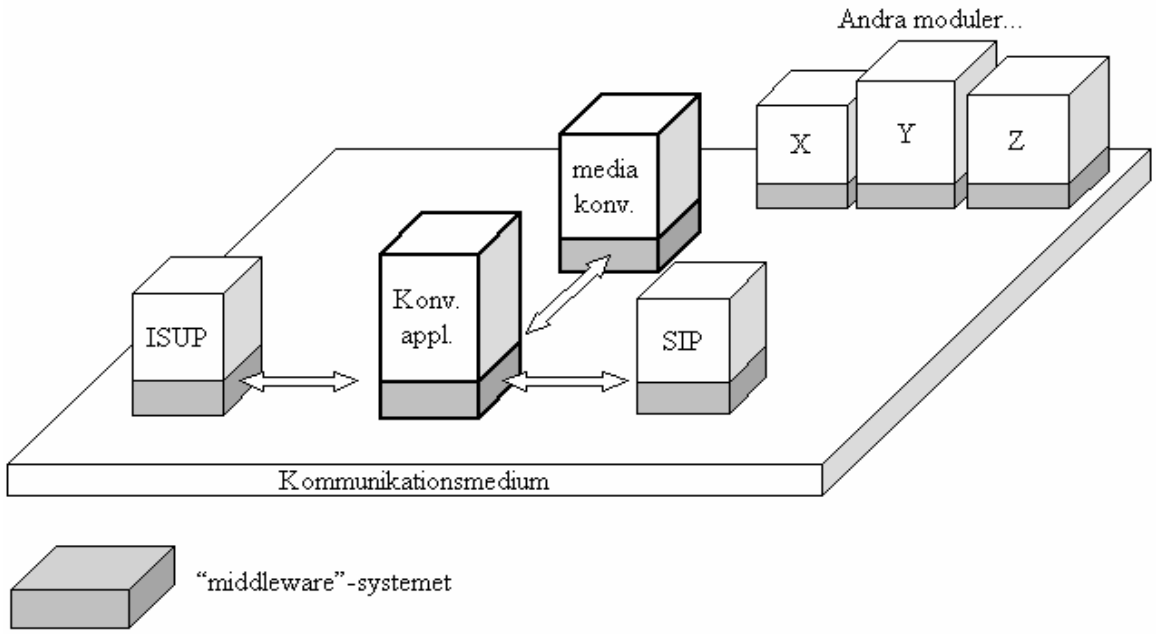
Först presenteras gateway-mjukvarans modulbaserade uppbyggnad, och hur denna influerar konstruktionen av konverteringsapplikationen och mediakonverteraren. I synnerhet beskrivs det “middleware”-system, mjukvaran som möjliggör meddelandeorienterad kommunikation mellan olika moduler, så att exempelvis konverteringsapplikationen kan interagera med ISUP-, SIP- och mediakonverteringsmodulerna. Därefter introduceras de designmetoder som använts som konceptuell grund för utvecklingen, nämligen tillstånds- och flödesdiagram. Förklaringar ges på hur man med dessa metoder relativt enkelt kan beskriva protokollmappningen mellan ISUP och SIP.

När designkoncepten påvisats förklaras först konverteringsapplikationens interna funktionalitet, med fokus på protokollmappningen mellan ISUP och SIP. Vi ska särskilt se användandet av funktioner som hanterar inkommanden av protokollmeddelanden i olika tillstånd, och som ser till att exempelvis mappning sker till motsvarande protokoll eller att fel hanteras. Ett scenario beskrivs där konverteringsapplikationen utför protokollmappningen från ISUP till SIP för ett inkommande protokollmeddelande. Därefter presenteras implementeringen av mediakonverteraren, och hur denna kontrolleras av konverteringsapplikationen för att koppla upp mediavägen.

Slutligen redogörs för den utvecklingsmetodik som använts, och hur vi under arbetet har försökt att eftersträva idéerna bakom “extreme programming” (XP), och hur vi har utvecklat konverteringsapplikationen och mediakonverteraren iterativt med testning och versionshantering.

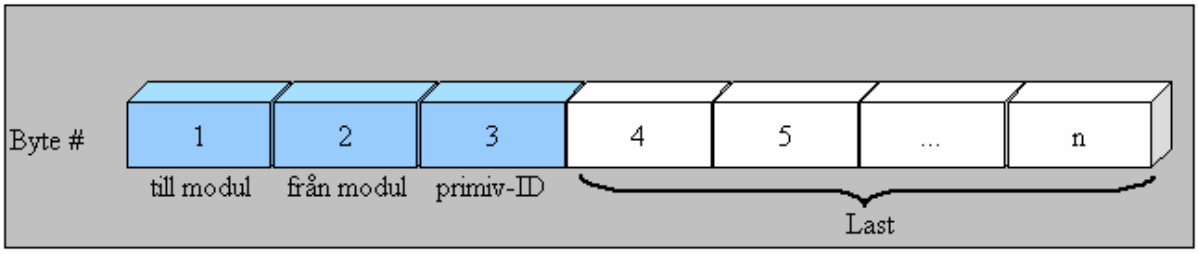
3.2 Gateway-systemets modulbaserade uppbyggnad

Gateway-systemets mjukvara är uppdelad i moduler som kommunicerar med varandra genom att skicka meddelanden (ej att förväxla med protokollmeddelanden) mellan sig, via ett gemensamt “middleware”-system som framgår av Figur 3.1. ”Middleware”-systemet finns medlänkat i varje modul, och det kan ses som det undre lager i modulerna vilket ansvarar för att skicka och ta emot meddelanden (en liknelse kan göras med applikationslagret kontra undre lager i TCP/IP-stacken). Konverteringsapplikationen, SIP-stacken, ISUP-stacken samt mediakonverteraren är exempel på moduler, och kan ses som fristående processer som körs i systemet. Konverteringsapplikationen använder sig av de andra modulerna för att utföra sin uppgift. ISUP- och SIP-modulerna behövs för att skicka och ta emot respektive protokollmeddelanden, och mediakonverteraren används för att upprätthålla mediaöverföringen. Men för att konverteringsapplikationen ska kunna använda de andra modulerna måste den först “binda” sig till dem, vilket görs genom att skicka en bindningsbegäran (ett särskilt meddelande) till var och en av modulerna. Internt i konverteringsapplikationen finns information om vilka moduler den behöver binda sig till vid uppstart. Först när bindningen är upprätthållen kan andra meddelanden börja utbytas med de nu bundna modulerna.

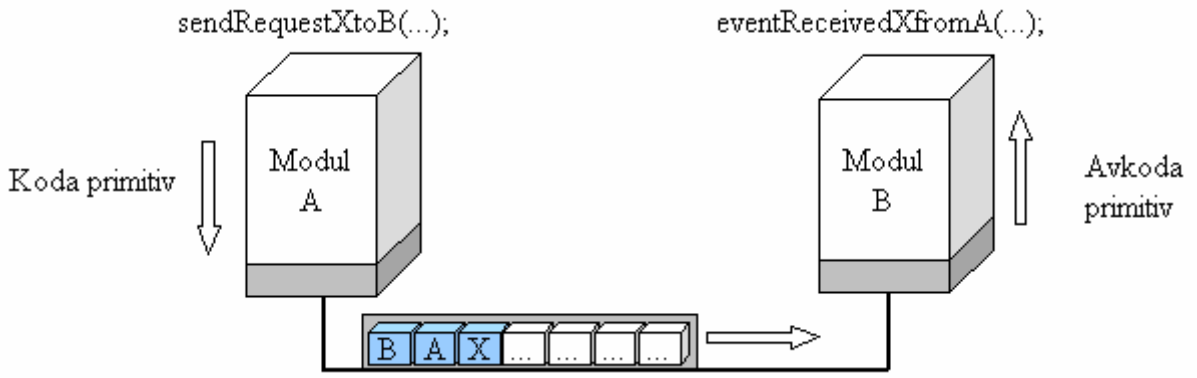


Figur 3.1 - Gateway-mjukvarumodulernas samverkan

Istället för att konverteringsapplikationen länkas ihop med ISUP-modulen och anropar en funktion i den senare (som exempelvis skickar ett IAM-meddelande), så skickar applikationen istället ett meddelande till ISUP-modulen. Meddelandet, eller meddelandeprimitiven som är den korrekta termen, innehåller information som skulle ha funnits i motsvarande funktionsanrop, såsom värden på parametrar. Utöver denna information finns även en header med primitiv-id (för att hitta rätt funktion), mottagarmodul-id och sändarmodul-id kodade som en sekvens av bytes. Principen beskrivs i Figur 3.2. Kodningen och avkodningen av meddelandeprimitiver måste göras av varje modul. "Middleware"-systemet ser till att skickade meddelandeprimitiver kommer till rätt mottagare, vilket åskådliggörs i Figur 3.3. Meddelandeprimitiver kan användas för att skicka förfrågningar och svar på förfrågningar, vilket används flitigt i kommunikationen mellan modulerna (en liknelse kan göras med parametrar och returvärden vid funktionsanrop).



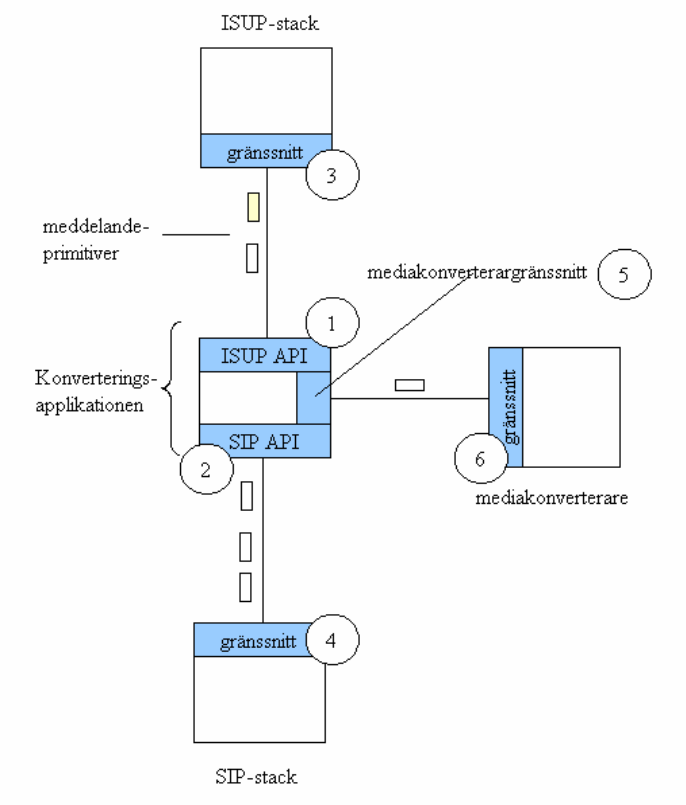
Figur 3.2 - Principen för hur en meddelandeprimitv är uppbyggd



Figur 3.3 - Hur en meddelandeprimitiv kodas, skickas, tas emot och avkodas

Genom detta förfarande ser man till att modulerna inte anropar varandra direkt. Det innebär förstås en viss overhead jämfört med om man länkat ihop modulerna direkt. Dock finns det fördelar, såsom ökad modularitet och lösare koppling. Om en modul behöver ändras kan detta förhoppningsvis göras utan att andra moduler behöver anpassas. Dessutom behöver modulerna inte vara skrivna i samma språk. Som framgår i avsnitt 3.7 underlättar det även vid testning, då man kan ersätta riktiga moduler med en “testmodul”. Genom att i denna testmodul inspektera (byte för byte) från den testade modulen skickade meddelandeprimitiver, kan man verifiera att den testade modulen fungerar som den ska. Testmodulen har nämligen information om hur de mottagna meddelandeprimitiverna (givet meddelandeprimitiver som den själv skickat) ska se ut.

Figur 3.4 kan betraktas som Figur 3.1 ovanifrån. Både ISUP- och SIP-API:et arbetar internt mot “middleware”-systemet (1 och 2). När man anropar en API-funktion för att skicka information till motsvarande modul, tar funktionen i fråga och packar parametrarna och annan information enligt ovan till en meddelandeprimitiv, och skickar den via “middleware”-systemet till rätt modul. Inkommande meddelandeprimitiver packas upp av motsvarande API och korrekt hanteringsfunktion anropas med motsvarande information. På så vis sker detta mer eller mindre transparent för programmeraren. ISUP- och SIP-modulen har färdiga gränssnitt mot ”middleware”-systemet för att kunna hantera, skicka och ta emot meddelandeprimitiver (3 och 4). Vi kommer själva att få utveckla ett gränssnitt så att konverteringsapplikationen kan arbeta mot mediakonverteraren (5), samt ett motsvarande gränssnitt för mediakonverteringsmodulen (6).



Figur 3.4 - Kommunikation via "middleware"

3.3 Designmodell

3.3.1 Befintligt system

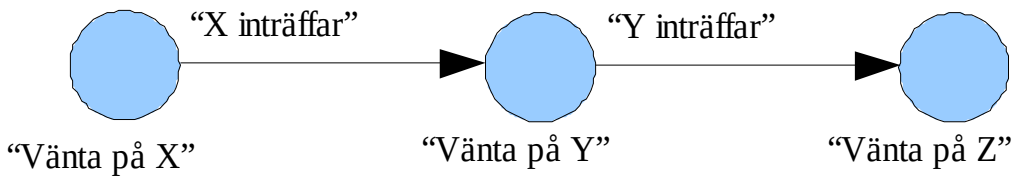
Vi börjar med att beskriva designen av konverteringsapplikationen. Den övergripande arkitekturen hos den befintliga applikationen för konvertering mellan ISUP och ISDN som legat till grund för arbetet, har inte behövt ändras nämnvärt om man ser till konverteringsapplikationens logiska uppdelning. Förvisso har ISDN ersatts av SIP, men förfarandet är det samma. Fokus har därför istället lagts på designen av funktionaliteten för själva mappningen mellan ISUP- och SIP-protokollen. Vi har valt att återanvända den designmodell som fanns för den tidigare applikationen, och beskriver i det här avsnittet de grundläggande idéerna.

3.3.2 Händelser och tillstånd

Ankomsten av en meddelandeprimitiv till konverteringsapplikationen, vare sig det rör sig om en förfrågning eller ett svar, kan ses som en ”händelse”. Detta hanteras av respektive API (för SIP och ISUP) som anropar en ändamålsenlig hanteringsfunktion när ett meddelande av viss typ anländer. Det är upp till användaren av API:erna (alltså programmeraren) att själv implementera sådana funktioner. Under kompilering av systemet länkas dessa till interna funktionsanrop i API-biblioteket. Exempelvis kan en funktion som hanterar inkommande förfrågningar om sessionsinitiering på ISUP-sidan innehålla kod som, direkt eller indirekt, skickar motsvarande meddelande till SIP-sidan, och vice versa.

Vidare kan konverteringsapplikationen befinna sig i olika ”tillstånd”. Innan en förfrågning kommit in från SIP- eller ISUP-sidan för att initiera ett samtal är applikationen i ett väntetillstånd. Därefter, beroende på vilka händelser som inträffar (alltså vilka meddelandeprimitiver som tas emot) kommer applikationen att gå från tillstånd till tillstånd. Ponera exempelvis att konverteringsapplikationen tagit emot en förfrågan från ISUP via ISUP-modulen, mappat den till motsvarande SIP-förfrågan, X, som den skickat iväg den till SIP-modulen. Antag vidare att konverteringsapplikationen därefter förväntar sig ett svar från SIP-sidan. Man kan betrakta denna väntan som applikationen befinnandes i “vänta på svar på SIP-förfrågan X”-tillståndet. När applikationen sedan får det väntade svaret kan tillståndet bytas. Om istället en oväntad händelse inträffar (det vill säga att applikationen får en primitiv som inte hör hemma i tillståndet) måste det hanteras. Antingen kan man skriva en funktion som förvisso hanterar den ogiltiga händelsen i tillståndet, men som inte gör något konkret (vare sig protokollmappning eller tillståndsbyte) förutom att eventuellt logga det inträffade i en loggfil för senare analys, eller så innehåller funktionen felhantering som exempelvis kopplar ner samtalet.

Modellen med händelser och tillstånd visualiseras ofta genom så kallade tillståndsdiagram. Ett tillståndsdiagram består av en mängd noder och en mängd kanter, som bildar en riktad graf. Noderna symboliserar tillstånd, medan kanterna representerar övergången från ett tillstånd till ett annat när en händelse inträffar.



Figur 3.5 - Exempel på tillståndsdiagram

Konverteringsapplikationens design har gjorts till stor del med ovanstående koncept. Ankomsten av ISUP- eller SIP-primitiver ses som händelser, och beroende på när och hur dessa anländer går applikationen från tillstånd till tillstånd. När ett samtal kopplats ned och resurser frigjorts befinner sig applikationen åter i väntetillståndet, och kan hantera nästa samtal. Tillståndsdiagram är passande representation då de ger en god överblick av hur applikationen internt hanterar samtal, både initiering, modifiering och nedkoppling. Dessutom användes samma koncept för att beskriva det befintliga systemet.

3.3.3 Flödesdiagram

Förutom hanteringen av inkommande primitiver måste skickandet av utgående primitiver beskrivas. Dessa framgår inte i tillståndsdiagrammet, men måste likväl hanteras när vissa händelser inträffar. För att visa detta används flödesdiagram som påvisar interageringen mellan ISUP och SIP genom gateway-systemet. Ett antal sådana sekvensdiagram finns redan beskrivna i RFC 3398 [1] och [23], och ett exempel ges i bilaga A. Vi har valt att betrakta dessa flödesdiagram som typiska användarfall, och har även baserat testningen av applikationen på dessa (se avsnitt 3.7).

3.4 Applikationens uppbyggnad

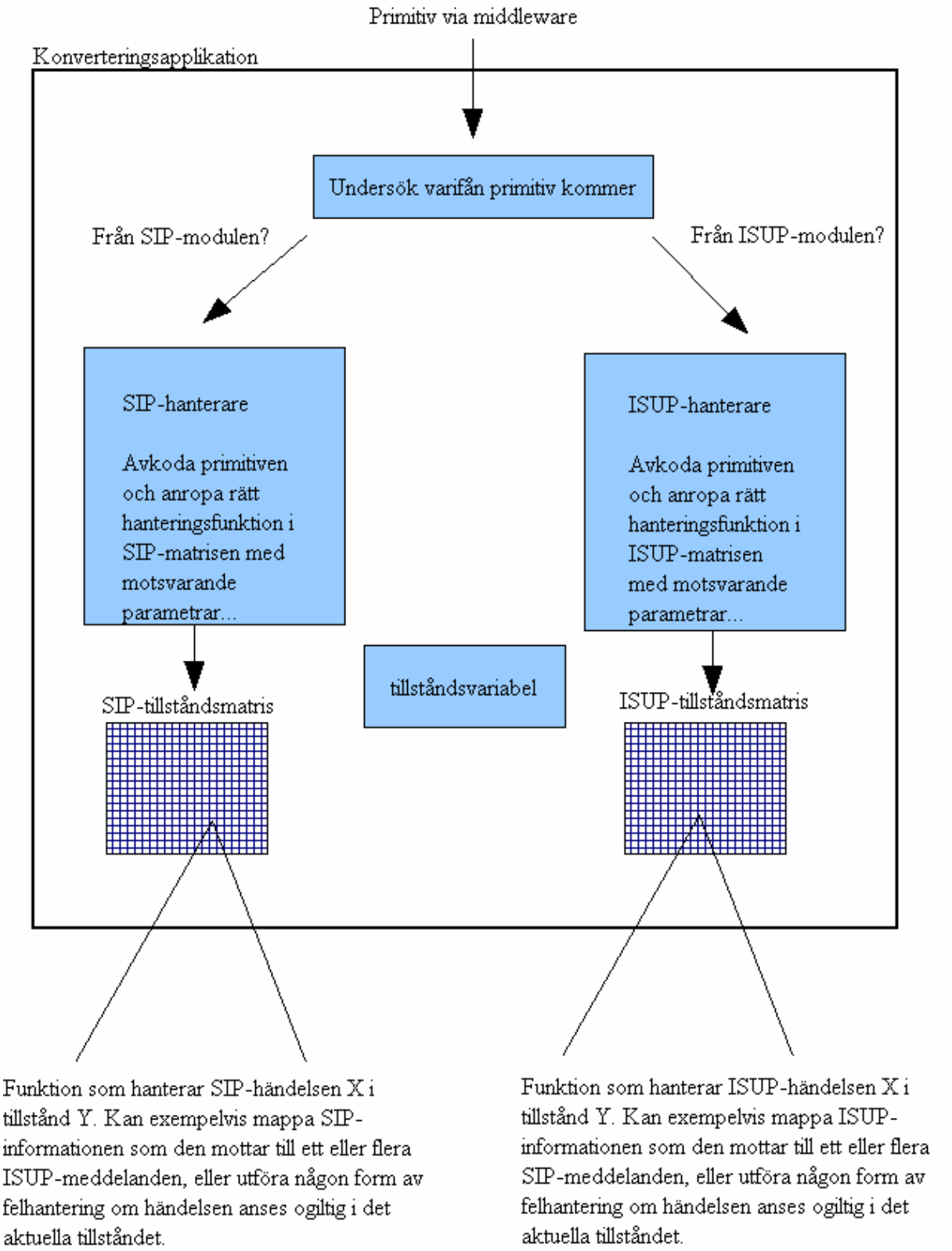
Nu när de bakomliggande idéerna med händelser och tillstånd introducerats, samt modulernas samverkan via “middleware”-systemet förklarats är det dags att beskriva hur konverteringsapplikationen har implementerats. Precis som var fallet med designen, har implementationsvalen också till stor del gjorts i enlighet med den befintliga applikationen.

Detta då vi på uppdragsgivarens begäran stegvis har ersatt den ursprungliga ISDN-funktionaliteten med motsvarande för SIP. En logisk överblick av systemet ges i Figur 3.6.

Via “middleware”-systemet kommer meddelandeprimitiver från andra moduler till applikationen. Meddelandeprimitiverna kan vara skickade från ISUP-, SIP- eller mediakonverteringsmodulen. När konverteringsapplikationen tagit reda på från vilken avsändare primitiven kommer ifrån, skickar den primitiven till avkodning, vilket applikationen antingen kan göra på egen hand eller överlåta till ett ändamålsenligt API (SIP eller ISUP). När detta är gjort kontrollerar applikationen vilken “händelse” som den mottagna primitiven motsvarar, och använder den informationen tillsammans med aktuellt “tillstånd” för att avgöra vilken funktion i tillståndsmatrisen som ska “hantera händelsen i det givna tillståndet”. Denna funktion kan i sin tur exempelvis innehålla funktionalitet som skickar iväg en eller flera nya primitiver och byter tillstånd, eller felhantering om händelsen inte förväntades inträffa i det aktuella tillståndet.

Värt att notera är alltså att det finns två stycken matriser, för att hantera ISUP- respektive SIP-relaterade händelser, men bara *en* gemensam tillståndsvariabel. Funktionerna som nås genom matriserna sköter mappningen mellan protokollen, och en funktion i ISUP-matrisen för att hantera inkommandet av ett IAM-meddelande (i det tillstånd ett IAM-meddelande accepteras) kan exempelvis innehålla kod för att skicka motsvarande INVITE-meddelande till SIP-sidan. Arbetet har till stor del bestått i att skriva om matrisfunktionera för ISUP-händelser så att de skickar SIP-meddelanden istället för ISDN-meddelanden, samt att ersätta ISDN-matrisen med motsvarande SIP-matris.

Antag att ett ISUP-meddelande anländer till gateway-systemet. ISUP-modulen kommer att hantera detta och skicka motsvarande information till konverteringsapplikationen, i form av en meddelandeprimitiv. Konverteringsapplikationen i sin tur tar emot meddelandeprimitiven, upptäcker att den är skickad från ISUP-modulen, packar upp meddelandeprimitiven och analyserar vilken funktion i ISUP-matrisen som ska hantera ankomsten av den, givet det aktuella tillståndet. Matrisfunktionen i sin tur mappar ISUP-informationen i meddelandeprimitiven till SIP-information, vilken i sin tur kodas som en ny meddelandeprimitiv som slutligen skickas från konverteringsapplikationen till SIP-modulen. SIP-modulen tar därefter och skickar motsvarande information i ett SIP-meddelande ut ur gateway-systemet.



Figur 3.6 - Konverteringsapplikationens övergripande funktion

Applikationen lagrar internt information om pågående samtal, som används vid upprätthållandet av mediaöverföringen. Genom att koda en meddelandeprimitiv med information om hur mediaöverföringen ska gå till, såsom val av codec, vilken tidlucka som ska mappas till vilken port med mera (se kapitel 2), och skicka den till mediakonverteringsmodulen kommer den sistnämnda att utifrån informationen i meddelandeprimitiven möjliggöra mediaöverföring.

3.5 Mediakonverterarens uppbyggnad

Likt konverteringsapplikationen kan mediakonverteraren ta emot primiver och befinna sig i olika tillstånd, och har därför en egen intern tillståndsmatris. I matrisen finns funktioner för att hantera inkommandet av primitiver. Dessa funktioner utför ingen protokollmappning, utan konfigurerar hårdvaran som sköter mediakonverteringen. När konverteringsapplikationen efter lyckad initial protokollmappning väljer att etablera mediasessionen skickar den en begäran i form av en meddelandeprimitiv till mediakonverteraren om att koppla upp mediavägen. Antaget att mediakonverteraren då befinner sig i rätt tillstånd (”vänta på uppkopplingsförfrågan”) kommer den att avkoda den inkommande primitiven och använda informationen till att instruera hårdvaran hur mediaöverföringen ska ske. Detta gör mediakonverteraren via ett ändamålsenligt API. Informationen som anges i meddelandeprimitiven är exempelvis nätverkskabel och tidlucka för PSTN-sidan, och för IP-sidan lokalt portnummer, den andra SIP-nodens IP-adress samt portnummer, och val av codec. Givet att inga problem uppstår vid konfigurationen av hårdvaran byter mediakonverteraren därefter tillstånd till ”uppkopplad”. Konverteringsapplikationen kan på motsvarande sätt vid nedkoppling skicka en primitiv för att koppla ner mediaöverföringen. Meddelandeprimitiver som anländer i fel tillstånd hanteras på liknande sätt som i konverteringsapplikationen.

Som nämndes i avsnitt 3.2 måste först en inledande bindning ske mellan konverteringsapplikationen och mediakonverteraren för att de ska kunna utbyta andra meddelandeprimitiver. Detta görs vid systemets uppstart och behöver inte upprepas för varje nytt samtal. Konverteringsapplikationen skickar en bindningsbegäran, och mediakonverteraren svarar huruvida bindningen accepterades eller inte. Vidare har mediakonverteraren en intern rutin för initial konfiguration av hårdvaran, vilken också bara

anropas i startskedet. Där anges lågnivådetaljer som exempelvis klockning av komponenterna på mediakonverteringskortet.

3.6 Programutvecklingsmetodik

Vi använde oss av en del idéer från XP-modellen [20] för att utveckla mjukvaran till gateway-applikationen, däribland parprogrammering, 10-minute builds, informative workspace samt iterativ och testdriven utveckling. Vi använde dock en mer dynamisk iterationstid som mest beror på komplexiteten för det arbetsmoment som skulle utföras.

I och med att vi alltid hade en fungerande version av gateway-systemet sparat i versionshanteringssystemet så uppfyllde vi XP-principen om "10-minute-build", det vill säga att man på maximalt 10 minuter skulle kunna presentera en fungerande och (så långt det var möjligt) aktuell version av programmet.

Vår utveckling av gateway-systemet baserades på iterationer, där vi i början av en iteration bestämde oss för vilken funktionalitet som skulle implementeras och skrev testfall som krävde denna funktionalitet för att inte misslyckas. Iterationens slut var således när vi hade implementerat denna funktionalitet och alla testfallen accepterades (gick igenom utan fel). När en iteration var över laddades den nya versionen upp till versionshanteringssystemet.

En annan XP-princip som vi har anammat är principen om testdriven utveckling, där testfallen skrivs först, och dessa testfall fallerar om programmet inte har den funktionalitet som krävs för att testfallen skall gå igenom felfritt. Det är sedan upp till programmerarna att förse programmet med sådan funktionalitet att testfallen går igenom, och detta blev således ett bevis för programmets funktionalitet.

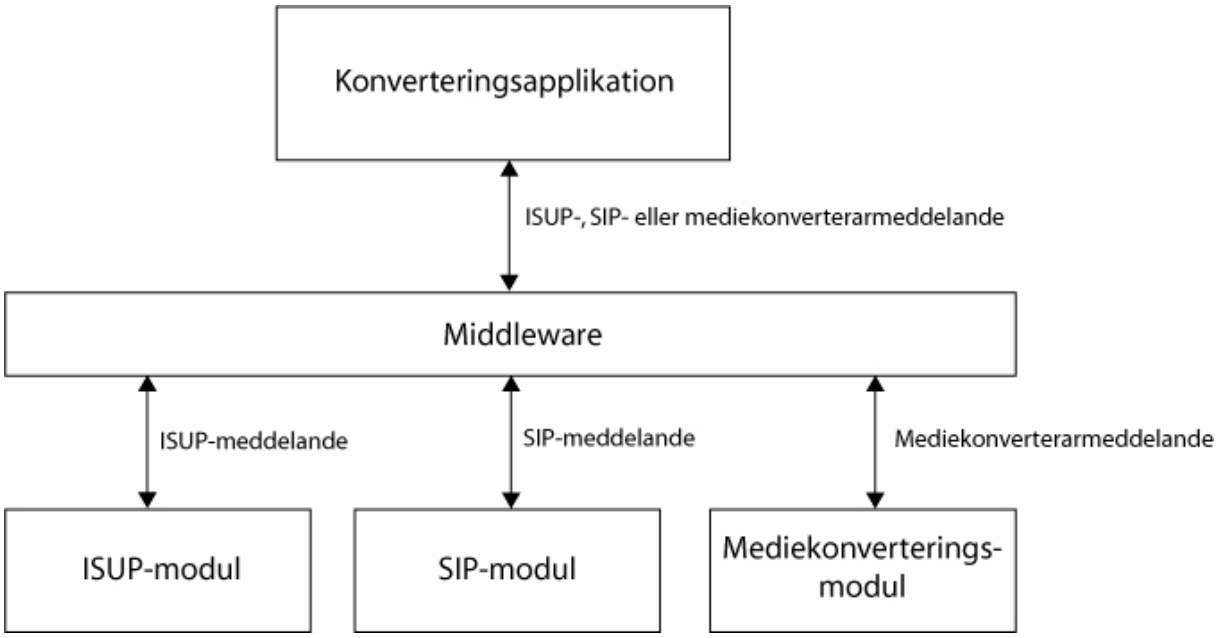
3.7 Testmiljö

Testmiljön för vår gateway-applikation kom att simulera kommunikationen med SIP-stacken och SS7-stacken såväl som mediakonverteraren med hjälp av det ”middleware” som

alla moduler kommunicerade med. Gateway-applikationen var helt ovetandes om detta när den skickade primitiver som var ämnade åt någon annan modul, och upplevde det som att den skickade primitiver till SIP- och ISUP-API:et men i realiteten fångades dessa primitiver upp av testmiljön som eventuellt skickade tillbaka nya primitiver till applikationen.

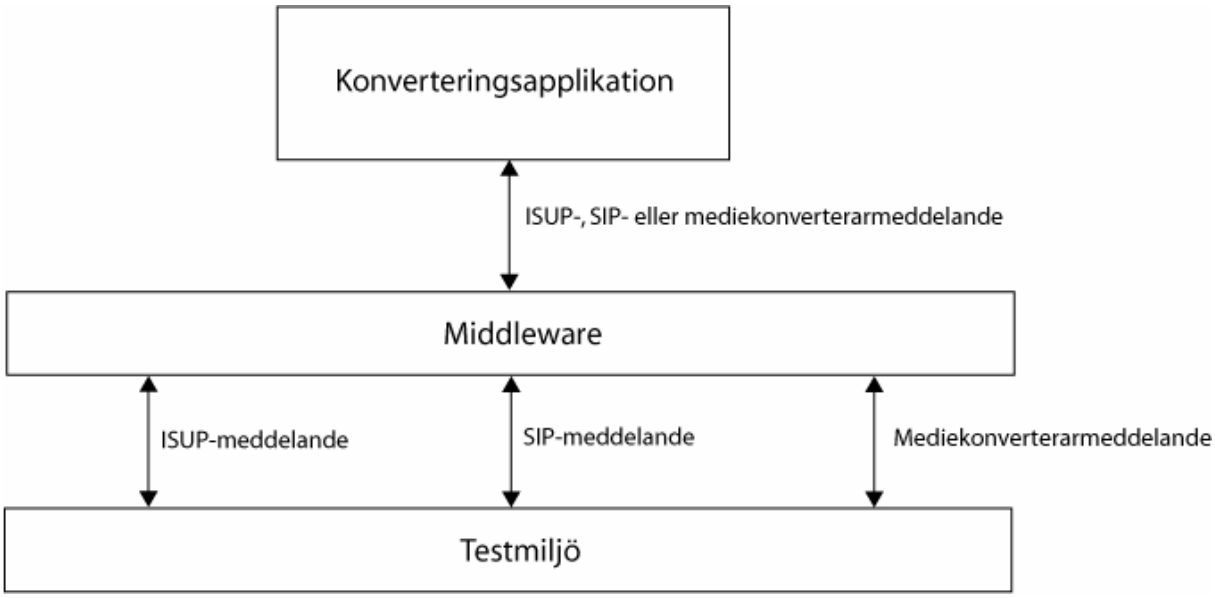
Våra tester för gateway-systemet var av så kallad "black box"-art. Detta innebär att testerna skrivs utan vetskap om programmets interna funktionalitet, utan endast testar med vetskap om den yttre funktionaliteten. Testmiljön ger en viss inmatning och säkerställer att programmet ger förväntad utmatning. Ett alternativ till "black box"-testning är så kallad "white box"-testning där man med vetskap om programmets interna struktur ger en viss inmatning som förhoppningsvis kan påvisa misstag och felaktigheter i koden. För vår prototyp kom vi dock endast att använda oss av ”black box”-tester och funktionstester på målhårdvaran.

Den testmiljö som vi kom att använda oss av var en version av CUnit [21]. Således kom våra testfall att bestå av C-kod som med hjälp av macros arbetade mot det ”middleware” som var gemensamt för samtliga moduler i systemet. För att testfallen skulle gå igenom behövde vi specificera de primitiver som detta kommunikationslager skickade och tog emot. Det fanns 2 olika typer av tester som kunde skrivas med hjälp av CUnit, interaktiva och icke-interaktiva, och de testfall som vi skrev för vårt program var av icke-interaktiv karaktär, även kallat "basic test". Detta innebar i praktiken att testfallen inte berodde på någon extern källa för inmatning, utan alla test kördes direkt och resultaten och statistiken för testfallen skrevs ut på skärmen samt till en loggfil. Testfallen delades in i logiska grupper för enklare översikt, där till exempel uppkoppling var en grupp, nedkoppling en annan.



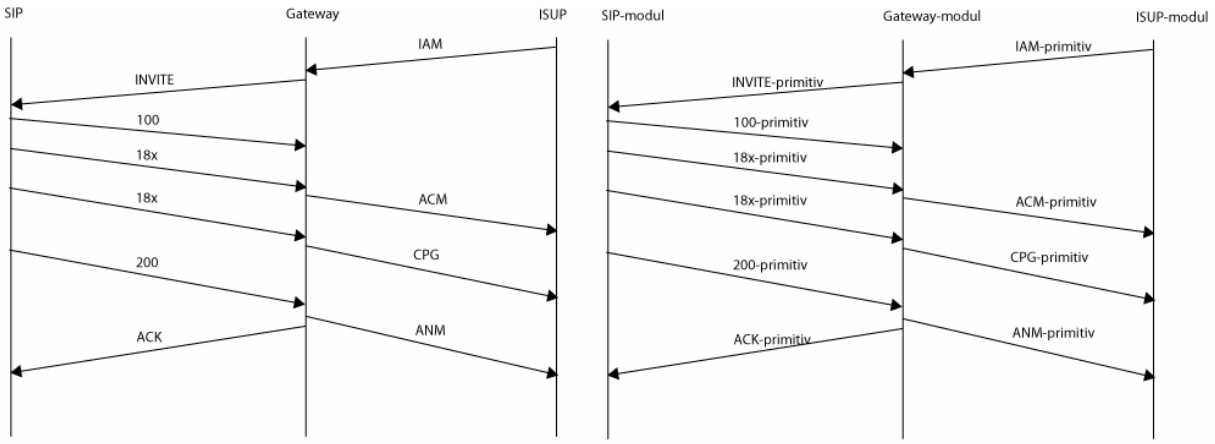
Figur 3.7 - Bild av hur gateway-systemets mjukvara interagerar med andra moduler genom ”middleware”-systemet

När en modul skickar ett meddelande ämnat åt någon annan modul så skickas detta meddelande genom ”middleware”-systemet, som vidarebefordrar meddelandet till rätt modul. Det finns möjlighet att konfigurera systemet för att specificera vilka moduler som får kommunicera med varandra. I vårt gateway-system så behövde gateway-applikationen kunna kommunicera med ISUP-, SIP- och mediakonverteringsmodulerna, men dessa i sin tur behövde endast kunna skicka och ta emot meddelanden till och från gateway-applikationen. I Figur 3.7 illustreras vilka typer av meddelanden som varje modul kom i kontakt med i gateway-systemet. Gateway-applikationen kunde ta emot och skicka ISUP-, SIP- och mediakonverterarmeddelanden, men till exempel ISUP-modulen kunde bara ta emot och skicka meddelanden som kom från eller var ämnade till gateway-applikationen.



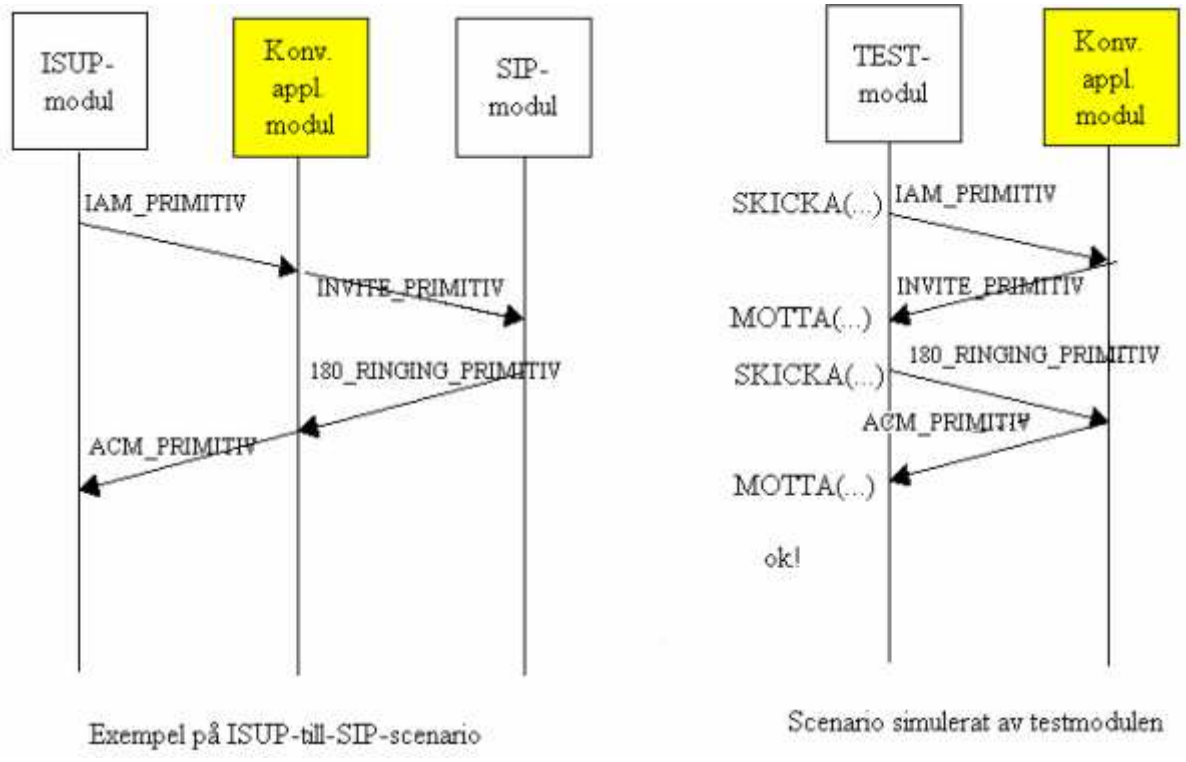
Figur 3.8 - Bild av hur testmiljön simulerar ISUP-, SIP- och mediakonverteringsmodulerna när konverteringsapplikationen testas

När gateway-applikationen skulle testas så konfigurerades systemet så att alla meddelanden som applikationen skickade till ISUP-, SIP och mediakonverteringsmodulerna dirigerades om till testmiljön. På så sätt hade testmiljön full kontroll över vilka meddelanden som gick till och från gateway-applikationen, och således kunde man genomföra ”black box”-tester på gateway-applikationen.



Figur 3.9 - Användningsfall 8.1.1 "En-bloc call setup (non auto-answer)" i RFC 3398 och dess implementation i gateway-systemet

När ett användningsfall skulle implementeras så behövdes de primitiver som förekom i användningsfallet översättas till motsvarande meddelandeprimitiver som ”middleware”-systemet sedan hanterade. I Figur 3.9 finns två sekvensdiagrammet för en typisk uppkoppling från ISUP-sidan till SIP-sidan. Det vänstra sekvensdiagrammet visar hur RFC-dokumentet specificerar vilka primitiver som skickas och tas emot av respektive part, och det högra diagrammet visar hur detta översattes i gateway-systemet som skickade motsvarande meddelandeprimitiver mellan ISUP- och SIP-modulerna.



Figur 3.10 - Exempel på en sekvens av meddelanden och hur denna sekvens ser ut när den har översatts till motsvarande test i testmiljön

När vi i början av en iteration bestämde vilket användningsfall i RFC-dokumentet [4] alternativt ITU-specifikationen [23] som borde implementeras så noterade vi sekvensen som framgick av användningsfallet och vilka tillstånd som vår applikation skulle komma att hamna i. Nästa steg var att modellera de primitiver som skulle skickas mellan gateway-applikationen och SIP- respektive ISUP-API:et via ”middleware”-systemet. Ett testfall skrevs som testade funktionaliteten i gateway-systemets mjukvara. I samband med att testerna skrevs så var man även tvungen att specificera innehållet i de meddelandeprimitiver som testmiljön skickade och tog emot. Dessa data fanns specificerade i systemdokumentationen som vi fick

av uppdragsgivaren. Det nya testfallet kom givetvis att misslyckas då programmet ännu inte hade den funktionalitet som krävdes för att testfallet skulle gå igenom. Det blev då det naturliga steget att uppdatera mjukvaran så att testfallet gick igenom. När detta skett så hade iterationen avslutats och den nya, fungerande versionen av mjukvaran laddats upp till versionshanteringssystemet.

Varje modul som ska kommunicera med någon annan modul måste skapa en koppling modulerna emellan innan kommunikationen kan fortskrida. I vårt fall var det SS7-stacken, SIP-stacken och mediakonverteraren som skulle kopplas till gateway-applikationen. Detta skedde genom att den modul som skulle kommunicera med gateway-applikationen skickade en bindningsbegäran och att denna applikation svarade med en bindningsbekräftelse, vilket bekräftade att kommunikationsvägen fungerade. Efter detta kunde modulerna skicka meddelanden till varandra. Detta förfarande gällde för SS7-stacken, SIP-stacken och för mediakonverteraren. Värt att notera är att de inledande bindningsprimitiverna placerades i ett separat testfall och att dessa testfall senare låg som förkrav för många andra tester då dessa tester utgick från att en giltig uppkoppling fanns mellan modulerna. Dessa inledande meddelandeprimitiver fanns dock inte med i användningsfallet då de var specifika för den miljö som vi arbetade i.

3.8 Versionshantering

I vårt projekt kom vi att använda oss av Subversion för versionshantering.

Subversion är ett projekt som har som målsättning att vara bättre än CVS [22], och Subversion-projektet har en helt öppen och fritt tillgänglig källkod, och är kostnadsfritt att använda. Subversion används som en databas med olika versioner av ett projekts källkod, en så kallad repository. Användare kan hämta ut och lämna in uppdateringar av koden med hjälp av denna repository. Detta innebär att man med lätthet kan spåra ändringar i källkoden, och vid eventuella fel kan man lätt gå tillbaka till en tidigare fungerande version för att lättare kunna hitta felet.

Subversion har många likheter gentemot CVS, men även vissa skillnader. Bland annat så är subversions uppdateringar i databasen atomiska, till skillnad från CVS som kan misslyckas

med att flytta över flera filer om till exempel en konflikt uppstår i en av dessa filer när den laddas upp till databasen. Subversion kan precis som CVS tillåta att flera användare arbetar mot databasen samtidigt.

Ett versionshanteringssystem var i det närmaste nödvändigt för oss i och med att vi utvecklade gateway-applikationen i enlighet med de XP-principer som vi valt att använda oss av, som bland annat tar upp versionshantering som ett krav [20].

3.9 Sammanfattning

I kapitel 3 har vi beskrivit koncepten som användes vid konverteringsapplikationens design, genom att presentera tillstånds- och flödesdiagram. Dessutom har en beskrivning getts av hur de moduler som gateway-systemet består av kommunicerar med varandra genom ”middleware”-systemet. Meddelandeprimitiver för kommunikation mellan moduler har också beskrivits, och det har framgått vilken funktionalitet respektive modul behöver för att möjliggöra hantering, packning och uppackning av meddelandeprimitiver. Vidare har konverteringsapplikationens logiska uppbyggnad presenterats, och ett scenario förklarats som beskriver hur mappningen kan gå till från ISUP till SIP.

Den programutvecklingsmetodik vi har valt att använda oss av när vi har utvecklat mjukvaran till gateway-systemet består till stora delar av XP-modellen, med bland annat parprogrammering och testdriven och iterationsbaserad utveckling. När gateway-applikationen har utökats med ny funktionalitet så har testfallen skrivits först, och uppgiften har sedan varit att lägga till kod i applikationen så att dessa testfall slutligen gått igenom. Efter det att testerna gått igenom och gateway-applikationen blev uppdaterad med ny funktionalitet så sparades denna nya version i en versionshanteringsdatabas, detta dels för att försäkra sig om att vi alltid hade en backup av den senaste versionen, och dels för att vi skulle kunna gå tillbaka till en tidigare fungerande version om vi av misstag råkat raderat kod eller filer. I vårt fall kom vi att använda oss av versionshanteringssystemet Subversion.

4 Resultat och utvärdering

4.1 Introduktion

I det här kapitlet presenteras resultaten av de tester som genomförts. På så vis framgår det vilka scenarion gateway-systemet klarar av att hantera. Därefter görs en utvärdering av testresultaten och arbetet i sin helhet, i synnerhet vilka problem som uppstått och hur de har lösts.

Testerna kan delas in i enhetstester och funktionstester. Enhetstesterna har utförts i testmiljön som beskrevs i kapitel 3 och används för att säkerställa att konverteringsapplikationen och mediakonverteraren skickar och tar emot meddelandeprimitiver korrekt. Resultaten av dessa testfall presenteras i form av sekvensdiagram med beskrivande text, och motsvarande pseudo-testkod finns i bilaga B.

Funktionstester å andra sidan görs på den faktiska hårdvaran, utan testmiljöns inblandning. Det är då som det framgår om systemet fungerar i verkligheten. Förutom att använda systemet för att ringa ett samtal från en ”vanlig” telefon till en VoIP-telefon som vi kom att göra, kan man även göra mer avancerade tester som att utsätta systemet för belastningar (exempelvis stora mängder trafik, felaktiga protokollmeddelanden, fördröjningar i nätverket) för att simulera sällsynta men tillika möjliga extremfall. Resultaten av funktionstester presenteras i avsnitt 4.2.

När testresultaten har analyserats utvärderas också arbetet i stort, i form av förstudier (inläsning av dokumentation) och tillvägagångssättet vid utvecklingen av gateway-systemet.

4.2 Resultat

När vi hade utvecklat gateway-applikationen såväl som mediakonverterarens mjukvara så hade vi skrivit testfall för att säkerställa funktionaliteten i dessa moduler. Dessa testfall skrevs dels för att testa den grundläggande funktionaliteten, när modulen mottog primitiver med korrekta data och i rätt ordning, men även för att testa att modulen kunde upptäcka felaktigheter och reagera på dessa. Felaktigheter kunde vara primitiver som skickades i fel ordning eller inte alls, eller primitiver som innehöll orimliga värden. Modulen var i dessa fall tvungen att reagera på detta avvikande beteende, och testmiljön garanterade att modulen reagerade på ett korrekt sätt genom att utvärdera vilket tillstånd som modulen befann sig i och/eller att mottaga primitiver med felmeddelanden som modulen skickade. I bilaga B redovisas testfallen med sekvensdiagram och kod.

Testfallen kan klassindelas i olika grupper beroende på vilket område som de testar. De klassindelningar av testfall som vi har gjort under arbetet är:

Uppkoppling	Tester för att säkerställa att uppkoppling av samtal fungerar
Nedkoppling	Tester för att säkerställa att nedkoppling av samtal fungerar
Interntest	Tester som säkerställer att gateway-applikationen befinner sig i rätt tillstånd, har rätt uppfattning om antalet aktiva samtal etc.
Felkontroll	Tester som skickar primitiver i fel format eller med felaktiga värden och säkerställer att gateway-applikationen hanterar avvikande beteende på ett korrekt sätt.

Tabell 4.1 - Tabell över de olika klassindelningarna för testfall

Det hände ofta att ett testfall kunde anses tillhöra fler än en kategori, exempelvis ett felkontrollstest under uppkopplingsfasen av ett samtal som skulle kunna tillhöra både kategorin uppkoppling såväl som felkontroll. Vi har dock valt att kategorisera testfallen efter deras syfte, så att om syftet med testfallet är att felkontrollera gateway-applikationen för en viss sekvens av primitiver under uppkopplingsfasen så är det i första hand en felkontroll. Om

ett test gick ut på att testa de grundläggande meddelandesekvenserna (sekvenser där allt går rätt till och inga problem uppstår) under uppkopplingsfasen ansågs testfallet följaktligen tillhöra uppkopplingskategorin.

När alla testfallen för gateway-applikationen och mediakonverteraren var gjorda hade vi 31 testfall för gateway-applikationen och 9 testfall för mediakonverteraren.

Modul	Antal testfall	Antal godkända testfall	Antal icke godkända testfall	Antal godkända testfall (%)
Gateway-applikationen	27	24	3	89
Mediakonver-teraren	9	9	0	100

Tabell 4.2 - Tabell med statistik över antalet testfall för gateway-applikationen och mediakonverteraren.

	Gateway-applikationen	Mediakonverteraren
Uppkoppling	15	1
Nedkoppling	4	1
Interntest	8	5
Felkontroll	0	2

Tabell 4.3 - Tabell med fördelningen av antalet testfall baserat på deras klassindelning för respektive modul

När vi mot slutet av experimentet blev klara med mjukvaran och de nödvändiga testfallen hade skrivits så fick vi möjligheten att testa gateway-systemet i verkligheten. I uppdragsgivarens laborationssalar fick vi möjligheten att funktionstesta systemet i en sluten men ändå realistisk miljö. I ett funktionstest så testas systemet i en så verklighetsnära miljö

som möjligt, vilket kan avslöja brister och felaktigheter som inte syns i ”black box”-tester, där man själv skriver testfallen. I funktionstestet fanns inga testfall, utan det som var relevant var huruvida gateway-systemet kunde kommunicera korrekt med andra moduler och enheter i en verklighetstrogen miljö.

Första steget i detta funktionstest bestod av att helt enkelt montera kretskorten och överföra mjukvaran till dessa kort, sedan att starta systemet och se att alla komponenter aktiverades utan problem. Speciell utrustning för att generera ISUP-meddelanden kopplades in som gick via gateway-systemet vidare till en SIP-telefon. Det slutgiltiga testet bestod av att lyfta luren, slå ett nummer och att detta samtal sedan kopplades via gateway-systemet till den andra telefonen.

I laborationssalarna fanns möjligheter att konfigurera stora delar av miljön som gateway-systemet testas i, bland annat hur telefonnumret skulle överföras (från ISUP-sidan så finns möjligheten att hela telefonnumret skickas i en ACM-primitiv, eller att delar av numret skickas i en eller flera SAM-primitiver) och vilken ISUP-standard som skulle användas. Eftersom det var en prototyp som vi hade konstruerat så fanns det inte funktionalitet för att hantera alla möjliga alternativ, utan var konstruerad för att visa principen för en PSTN/VoIP-gateway, således var miljön konfigurerad så att alla siffror i telefonnumret fanns i en enda ACM-primitiv. Vi hade även konfigurerat de kringliggande systemen så att vi på förhand visste vilken typ av ljudkomprimering som skulle användas, och behövde således inte hantera alla alternativ som fanns utan kunde inrikta oss på en enda standard.

Bindningen mellan modulerna lyckades på första försöket. Alla modulerna kunde kommunicera med varandra och inga felmeddelanden rapporterades. Problem uppstod dock när ett samtal skulle genomföras. För att lösa problemet kunde vi med hjälp av Ethereal se informationen som transporterades till och från gateway-systemet. Modulerna kunde konfigureras till att skriva ut fel-loggar och loggfiler som visade aktiviteten i SS7-stacken fanns också till hjälp. Vi fick även assistans av personal från uppdragsgivaren. Ur dessa fel-loggar kunde vi utläsa att SIP-stacken skickade ut meddelanden på nätverket men att dessa meddelanden direkt returnerades till samma modul som den emanerade ifrån. Det visade sig att på grund av ett felaktigt satt parameter i ett funktionsanrop till SIP-API:et så fick alla meddelanden som skickades från gateway-applikationen via SIP-stacken samma destinationsadress som ursprungsadress, vilket gav oväntade resultat. En snabb modifiering av

gateway-applikationen gjordes och laddades på nytt ner på hårdvaran. När detta problem var ur världen så skickades meddelanden ut på nätverket, men SIP-telefonen gav ingen märkbar reaktion. Problemet var en felkonfigurering i själva SIP-telefonen (den aktuella SIP-telefonen hade ett stort antal konfigurationsmöjligheter, och man kunde med hjälp av den inbyggda HTTP-servern ställa in dessa parametrar, men också via det grafiska gränssnittet på telefonen). När detta problem hade upptäckts och lösts så gjordes ett nytt försök att ringa till SIP-telefonen. Telefonen gav signal. Vi kunde efter detta funktionstest konstatera att gateway-applikationen kunde binda till de andra modulerna, och att det var möjligt att genomföra den signalmässiga delen av ett samtal mellan en ”vanlig” telefon och en SIP-telefon via gateway-systemet.

På grund av externa leveransproblem fick vi tyvärr inte tillgång till det hårdvarukort för konvertering av media som var nödvändigt för att testa om mediakonverteraren kunde få rösttrafiken att skickas mellan telefonerna efter uppkopplingsfasen.

4.3 Utvärdering av arbetet

4.3.1 Planering

Vid arbetets början gjordes en grov tidsplanering. Vi försökte identifiera lämpliga deluppgifter och tilldelade dem två veckor vardera. Men efter att arbetet fortgått under ett par veckor stod det dock klart att planeringen inte skulle hålla. Dels hade vi underskattat den mängd tid som behövdes för att läsa in oss i ämnesområdet, dels hade vi från början förbisett en rad uppgifter som vi upptäckte först senare i och med att implementeringen påbörjats. Några inledande tekniska problem med Make-filer och konfigurationsfiler tog också tid att lösa, uppskattningsvis två till tre veckor. Planeringen var således alltför optimistisk och vi kom senare att arbeta mer och mer ”fritt”, och istället rikta in oss på långsiktiga mål. Vi hade förmodligen haft nytta av att följa planeringen för att lättare bedöma framsteg och status, men då vi fann det svårt att planera med tanke på oförutsägbara problem var planeringen något vi mest såg som en lista av arbetsmoment utan tidsaspekt.

4.3.2 Inläsning och förberedelser

Innan konkret utveckling av konverteringsapplikationen och mediakonverteraren kunde ske behövde vi i inlärningssyfte inhämta information om berörda protokoll, programbibliotek och system. Till att börja med sökte vi reda på litteratur och annan dokumentation (mestadels webbsidor) om SIP- och ISUP-protokollen, och vi fann det lättare att hitta information om det förstnämnda. Vidare undersökte vi hur uppdragsgivarens proprietära API-bibliotek för protokollen i fråga kunde användas. Detta genom att studera tillhörande dokumentation med exempel, och genom att försöka överblicka hur biblioteken var uppbyggda av för ändamålet avsedda funktioner och datastrukturer. Vi anser att sekvensdiagram och kodexempel var till störst nytta, och hade gärna sett fler sådana.

Vi har även fått erfara att SIP och ISUP är väldigt olika protokoll. Medan SIP har varit relativt enkelt att lära sig och överblicka (till stor del beroende på de välskrivna och lättförståeliga böcker som skrivits om protokollet) så har ISUP varit betydligt mer svårgreppbart. ISUP är ett mycket komplext protokoll, och det finns många parametrar som måste sättas till rätt värden för att åstadkomma önskade resultat. ISUP-meddelanden är heller inte lika lättlästa som SIP-motsvarigheterna eftersom ISUP använder binära protokollmeddelanden. Vi tror även att vår tidigare oerfarenhet av PSTN försvårade det hela då vi inte från början var bekanta med den underliggande nätverksarkitekturen och SS7-stacken. Användandet av SIP var däremot lättare då vi redan hade grundläggande kunskaper i datakommunikation, och förstod begrepp som Internet, TCP/IP och routing. Vi saknade en enkel introduktion till de delar av ISUP som var relevanta för oss, och en kurs i ämnet hade förmodligen varit till stor hjälp.

Samtidigt försökte vi få oss en översikt av strukturen hos den befintliga applikationen för konvertering mellan ISDN och ISUP. Vi läste programmets källkod och försökte först identifiera dess huvuddelar och hur dessa hängde ihop. Utöver själva koden studerades även ett tillhörande tillståndsdiagram som visualiserade mappningen, och vi skapade motsvarande diagram för konvertering mellan ISUP och SIP.

Det befintliga systemet hade tillhörande dokumentation som förutsatte förkunskaper i signalering och gateway-utveckling hos läsaren, varför vi fann den svår att förstå. Vi är övertygade om att arbetet hade underlättats avsevärt om applikationen och systemet i stort dessutom hade beskrivits på en högre abstraktionsnivå i ett separat designdokument, med

nybörjare som oss i åtanke. Då hade förmodligen den faktiska implementeringen kunnat påbörjas tidigare.

Handledaren hos uppdragsgivaren har efter att vi kommenterat ovanstående gjort gällande att den ursprungliga applikationen utvecklades med antagandet att efterföljande utvecklare hade goda kunskaper i SIP- och ISUP-protokollen, och därmed motiverat den avancerade dokumentationen. Vi anser emellertid att även om vi besuttit dessa kunskaper skulle systemet ändå kunnat beskrivas – utöver protokollmappningen – för att underlätta för nykomna utvecklare som oss själva.

Utöver den ursprungliga applikationen var vi även tvungna att bekanta oss med ”middleware”-systemet och användandet av detta för att koppla samman mjukvarumoduler, samt inte minst förstå oss på testmiljön och förfarandet vid skrivandet av tester. Här var dock inlärningskurvan inte lika brant då handledaren beskrev ”middleware”-systemet och testmiljön.

4.3.3 Utveckling

Konverteringsapplikationen utvecklades genom att steg för steg byta ut ISDN-funktionaliteten i den ursprungliga applikationen mot motsvarande för SIP. Med utgångspunkt från testfallen för originalapplikationen skrevs nya testfall, och vi hade på så vis ganska enkelt att se vad som skulle göras. Dock var det från början inte uppenbart hur vi skulle börja modifiera den ursprungliga applikationen. Det första steget var att få applikationen att länkas samman med SIP API:et. När väl det var gjort skrevs kod för att möjliggöra bindning till SIP-modulen vid uppstart av applikationen. Därefter var det dags för den faktiska protokollmappningen, vilket ledde till att fler och fler API-funktioner kom att användas.

Många förenklingar har fått göras under utvecklingens gång. Bara ett begränsat antal scenarier stöds (se testresultaten), och många detaljer har fått ”hårdkodas” både i konverteringsapplikationen och i mediakonverteraren för att vi skulle hinna med arbetet i tid. Det har hela tiden varit fråga om att utveckla en prototyp av en VoIP/PSTN-gateway, och därför är systemet av förklarliga skäl inte lika dynamiskt och användbart som en färdig

produkt hade förväntats vara (vilket inte heller var tanken). Likväl kan prototypen ses som ett bevis på att det faktiskt går att mappa ISUP till SIP givet de tillgängliga API:erna.

Vi upplevde det vid några tillfällen som att det var oklart vad som skulle göras dynamiskt och inte. Det resulterade tyvärr i att vi ibland ägnade tid åt att sätta oss in i funktionalitet som egentligen inte behövdes. Förlorad tid är förstås förargligt, och vi inser att vi borde ha ställt fler frågor till uppdragsgivaren angående vad denne förväntade sig. Till viss del tror vi också att det beror på den föränderliga kravspecifikation som vi utgått ifrån, där vissa ursprungliga krav förenklades eller helt och hållet togs bort under arbetets gång.

4.3.4 Testning

Vi har haft stor nytta av den testdrivna utveckling som vi försökt att anamma i så stor utsträckning som möjligt. Dels har vi på ett ganska enkelt sätt kunnat se vad som gjorts och vad som återstått. Dessutom har det varit ovärderligt att kunna köra sviten av tester efter varje modifikation av koden för att säkerställa programmets korrekthet. Tester i testmiljön kunde dock inte garantera att de testade programmen fungerade i ”verkligheten”, så funktionstester behövdes också för att även avgöra om så var fallet. En annan nackdel med testerna i testmiljön var att de var besvärliga att skriva då testmiljön inte var anpassad för SIP-protokollet. En lättanvänd testmiljö hade kanske bättre motiverat skrivandet av tester.

4.4 Sammanfattning

I det här kapitlet har vi utvärderat resultatet av arbetet med att utveckla prototypen. Vi har valt att presentera resultatet i form av lyckade testfall utförda i testmiljön och en redogörelse för ett genomfört funktionstest, vilket visar på prototypens funktionalitet. Detta ger en fingervisning om vad prototypen är kapabel till. Statistiken för testerna har redovisats och en beskrivning av förfarandet vid funktionstestet.

Vi har även redovisat vad som har påverkat arbetet i stort, med fokus på planering, inläsning, utveckling och testning. I utvärderingen har vi bland annat kommit fram till att inlärningskurvan var högre än väntat, och att utvecklingsprojektet därmed försenades. Andra aspekter som påverkat utfallet var avancerad dokumentation av design och arkitektur hos

framförallt det befintliga systemet för konvertering mellan ISUP och ISDN som krävde goda förkunskaper hos läsaren och låg på en relativt hög nivå, och därför kanske inte lämpade sig för nybörjare som oss själva.

5 Slutsatser

5.1 Sammanfattning av arbetet

Arbetet att utveckla en prototyp av en VoIP/PSTN-gateway kan sägas ha varit uppdelat i två tydliga huvudmoment, dels inläsning av bakgrundsmaterial samt design av prototypen, och dels faktisk implementering. Den teoretiska aspekten innebar förutom studerande av signalering - ISUP, SIP och tillhörande protokoll, samt mappningen dem emellan - också att förstå hur konverteringsapplikationen skulle kunna utvecklas. Det senare genom att granska den befintliga applikationen för konvertering mellan ISUP och ISDN, och dess interaktion med övriga moduler i gateway-systemet. Det praktiska arbetet utfördes därefter genom att steg för steg i den ursprungliga applikationen ersätta ISDN- med SIP-funktionalitet, och att på så vis implementera mappningen mellan ISUP och SIP i enlighet med vad som specificeras i ITU-rekommendationen [23] för ändamålet. Dessutom skrevs en separat modul för mediakonvertering, också den i grund och botten en modifikation av en tidigare modul för konvertering mellan ISDN-buren röst och motsvarande för PSTN. Konverteringsapplikationen anropar mediakonverteraren när mediaöverföringen ska upprätthållas. På så vis är det meningen att mediakonverteraren ska se till att rätt tidlucka på en given lina kopplas till rätt RTP-port, så att användaren av den vanliga telefonen kan tala med användaren av ip-telefonen.

När man i efterhand blickar tillbaka på projekttiden är det lätt att dra flera slutsatser, både vad gäller arbetsmomenten och resultatet i form av den färdiga prototypen. Det faller sig naturligt att börja från början och först kommentera arbetet i dess startskede. Då hade vi mycket bakgrundsmaterial att sätta oss in i, och inläsningen tog längre tid än väntat. Innan vi fick en översikt av problemet, och innan vi förstod de ingående fackspråken, teknikerna och koncepten, var det svårt att avgränsa materialet – det vill säga att avgöra vad som var relevant för uppgiften. Det hade varit önskvärt med mer lättillförståelig dokumentation, framförallt om ISUP som är ett så pass komplext protokoll. Det befintliga systemet hade också kunnat förstås enklare om dess design och arkitektur dokumenterats också med nybörjare i åtanke.

Dock råder det inga tvivel om att tillgången till ett befintligt system, om än med närmast obefintlig dokumentation, var ett ovärderligt hjälpmedel. Genom att läsa ISDN-till-ISUP-applikationens kod och därefter gradvis ersätta den med vår egen kunde vi ta del av och själva nyttja väl beprövade och lämpliga designlösningar. Detaljerade specifikationer för protokollmappningen var också till stor hjälp, framförallt ITU-rekommendationen [23]. Vi hade således tillgång till en färdig mall för implementering och en detaljerad guide för protokollmappningen, varför vi egentligen bara behövde implementera funktionaliteten för att realisera den sistnämnda i den föregående. Vi anser att det var en bra utgångspunkt för utveckling av prototypen av gateway-systemet, och rekommenderar andra utvecklare i liknande situation att följa exemplet.

Testdriven utveckling har varit en viktig hörnsten i arbetet. Att köra serien av testsviter efter varje ändring gav snabbt indikationer på om konverteringsapplikationen respektive mediakonverteraren fungerade eller inte. Testning på hårdvaran behövde därmed heller inte göras i varje steg. Att i testmiljön simulera de omkringliggande modulerna gjorde det enkelt att isolera konverteringsapplikationen och mediakonverteraren, och eventuella fel kunde därför relativt snabbt lokaliseras. Dessutom var det enkelt att se vilka scenarier som kunde hanteras av prototypen genom att helt enkelt få listningen av genomförda testfall.

5.2 Problem

Under inläsningsperioden och utvecklingen var det största problemet att dokumentationen av det befintliga systemet var svår för oss att sätta sig in då den var skriven på en relativt avancerad nivå. Det rekommenderas därför att utvecklare av liknande system ser till att dokumentera design och arkitektur också med oerfarna i åtanke, kanske i ett separat dokument.

Vi fann även att dokumentationen (i form av användarmanualer och API-specifikationer) till hårdvarukortet som mediakonverteraren skulle köras på inte var särskilt hjälpsam för oinvigda. Många tekniska detaljer angavs bara vid namn, utan vidare förklaring. Vi fick dock hjälp av vår handledare hos uppdragsgivaren med tolka delar av materialet.

Tekniska besvär som följd felaktigt inställda konfigurationsfiler har också förekommit, men har alla kunnat lösas efter felsökning. Det har i de flesta fall berott på att vi från början inte förstått hur inställningarna i filerna skulle skrivas, eller varför. Konfiguration rörde främst middleware-systemet, och problemen var att modulerna inte kunde kommunicera korrekt. Det hade varit önskvärt med en ingående genomgång av dessa filer, så att vi kunnat identifiera problemen tidigare.

Nämnas kan också att oklarheter i vad som egentligen ingick i kravspecifikationen resulterade i att arbete lades ner på att förstå, och i viss mån implementera, funktionalitet som skulle visa sig inte behövas. I efterhand inser vi att vi borde ha talat mer i detalj med uppdragsgivaren om vad denne förväntade sig.

När vi funktionstestade prototypen hade vi problem att få kontakt med SIP-telefonen vid uppringning från den vanliga telefonen. Vi misstänkte först att det var översättningen av signaleringen som var orsaken, då loggarna från systemet var minst sagt märkliga. Meddelandeprimitiver skickades och togs emot i fel tillstånd. Vidare syntes inga SIP-meddelanden till på nätverket. Det visade sig bero på dels felaktigt angivna IP-nummer och portar i konverteringsapplikationen, vilket vi märkte när vi av en händelse analyserade trafiken på loopback-gränssnittet hos gateway-systemets nätverkskort. Systemet försökte nämligen ansluta till sig själv! Utöver detta beklagliga misstag fanns default-inställningar i ip-telefonen som förhindrade direktuppkoppling (utan att gå via någon mellanliggande SIP-server). När problemen ordnats upp fungerade emellertid signaleringen som väntat.

Slutligen kan nämnas problemet med att hårdvarukortet som skulle möjliggöra översättning av rösttrafiken inte anlände i tid, varför prototypen inte fick stöd för annat än grundläggande signalering. Mediakonverteraren kom därför inte att användas.

5.3 Den färdiga prototypen

Rent logiskt kan resultatet av arbetet med att utveckla prototypen delas upp i konverteringsapplikationen och mediakonverteraren, som två separata men samverkande mjukvarumoduler. Prototypen klarar av att hantera grundläggande översättning av signalering

vid uppkoppling och nedkoppling av ett samtal från en vanlig telefon och en ip-telefon. Detta innebär elementär mappning mellan ISUP och SIP (inklusive SDP) i konverteringsapplikationen. Telefonnumren mappas till SIP-adresser, och protokollmeddelanden med tillhörande parametrar översätts i enlighet med ITU-rekommendationen [23]. Konverteringsapplikationen instruerar mediakonverteraren via middleware-systemet om hur överföringen ska ske, med information erhållen vid den inledande signaleringen. Dock finns inte den nödvändiga hårdvara som mediakonverteraren behöver arbeta mot för att möjliggöra översättningen av rösttrafiken.

Förutom att prototypen inte klarar av alla tänkbara signaleringsscenarier (uppskattningsvis 30-40 %, med tanke på frånvaron av stöd för SIP-till-ISUP-uppringning och att vi inte täcker in alla fall från ISUP till SIP), kan heller inte röstöverföring göras. En annan påtaglig svaghet att vissa parametrar har hårdkodats. Exempel på sådana parametrar är vilken tidslucka som media kommer in på från PSTN-sidan. Parametrarna har placerats i särskilda källkodsfiler för enkel åtkomst. Dessa kan modifieras efter behov, och en omkompilering av mjukvaran är allt som krävs för att ändringarna ska träda i kraft. Dock kan det inte ändras under körning. Detta förfarande är en konsekvens av uppdragsgivarens rekommendation om att börja med den enklast möjliga lösningen, och göra systemet mer mångsidigt efter hand i mån av tid. Vi hann inte med att göra applikationen mer dynamisk.

Rent praktiskt kan prototypen användas för ringa ett enkelt, men ändock komplett samtal (utan mediaöverföring) från en vanlig telefon (kopplad till en ISUP-kompatibel telefonväxel) till en SIP-telefon. Den kan utan vidare modifikation användas exempelvis i demonstrationssyfte.

5.4 Vidareutveckling av prototypen

Som framgår av föregående avsnitt finns det stor motivering till vidareutveckling. Förutom att använda ett hårdvarukort för mediaöverföring, ses kanske främst behovet av att implementera stöd också för uppringning från en ip-telefon till en vanlig telefon, med andra ord scenarier där signalering påbörjas från SIP till ISUP. Detta ställer krav på nya funktioner för att hantera händelser, exempelvis hanteringen av ett inkommande INVITE-meddelande.

Med tanke på att denna översättning i mångt och mycket är motsatsen till den som har implementerats (fast ”åt andra hållet”), samt det faktum att ITU-rekommendationen [23] även här i detalj specificerar hur mappningen ska gå till, borde inte uppgiften vara alltför svår.

En annan viktig utökning vore att göra prototypen mer avancerad och dynamisk. Exempelvis skulle en komplett mappning mellan ISUP och SIP (inte bara de mest grundläggande protokollmeddelandena) vara användbar. Därtill skulle stöd för fler, speciella scenarier läggas till, som oväntade händelser (meddelanden tas emot i fel tillstånd) och kollisioner (SIP och ISUP försöker båda att göra samma sak). I prototypen hanteras detta bara för de grundläggande användarfallen. Systemet har heller inget stöd för att hantera belastningar. Dessutom borde gränssnittet mellan konverteringsapplikationen och mediakonverteraren byggas ut så att mediakonverteraren kan meddela problem vid resursallokering, samt att konverteringsapplikationen kan hantera sådana problem.

Att göra prototypen mer dynamisk innebär exempelvis att ersätta användandet av hårdkodade parametrar med funktionalitet för att beräkna motsvarande värden, så att en potentiell användare kan använda systemet utan att behöva konfigurera parametrarna på egen hand. Ett annat nödvändigt tillägg för att prototypen ska kunna användas i ”verkligheten” (som en kommersiell VoIP/PSTN-gateway) är förstås att den ska kunna hantera multipla pågående samtal. Detta ställer visserligen krav på implementering av datastrukturer som kan hålla reda på tillstånd för varje aktivt samtal, men borde inte heller vara en särskild arbetskrävande utökning (i synnerhet inte med avseende på att motsvarande lösning redan finns i den ursprungliga applikationen för ISUP-till-ISDN-konvertering).

5.5 Avslutande kommentarer

Arbetet har varit intressant och lärorikt. Det har inneburit att praktiskt få använda tidigare erhållna kunskaper i datakommunikation, och inte minst att inhämta nya. Internettelefoni känns som ett i allra högsta grad aktuellt ämne, och vi har fått god förståelse för hur tekniken fungerar. Intressant har det också varit att stifta bekantskap med de underliggande multimediaprotokollen, och då i synnerhet SIP – som vi tycker är ett designmässigt mycket snyggt protokoll i all sin enkelhet.

Vi har även fått studera och arbeta med tekniker från telekommunikation, vilket har varit en helt ny erfarenhet. Och även om arbetet i sig inte ställt krav på att förstå PSTN i större utsträckning än de berörda protokollen och överföringsmetoderna av media, har det automatiskt resulterat i att vi lärt oss mer om hur det världsomspännande telefonnätet fungerar i stort. En fascinerande lärdom som är minst lika intressant som den om Internets innandöme. Vidare har vi kunnat ställa de två nätverken mot varandra, och jämfört skillnader och likheter, fördelar och nackdelar, och gamla beprövade metoder mot nya revolutionerande IT-trender.

Implementeringen av prototypen satte våra tidigare kunskaper i mjukvaruutveckling på prov, och vi har lärt oss många nya idéer, designlösningar och tillvägagångssätt. Att för första gången få arbeta med ett riktigt projekt har varit oerhört givande. Det känns som att vi då först på allvar förstod vikten av design, testdriven utveckling, parprogrammering, dokumentation och de andra stora grundstommarna i ”software engineering”-läran. Att därefter i praktiken testa och felsöka mjukvaran på den riktiga hårdvaran var väldigt spännande, och utgjorde ett bra komplement till kodskrivandet.

Det slutgiltiga resultatet är en förenklad VoIP/PSTN-gateway som stödjer grundläggande användarfall vid uppringning från en vanlig telefon till en SIP-kapabel ip-telefon. Den har förvisso brister, men är likväl fullt möjlig att använda för att demonstrera enkel protokollmappning. Tyvärr kunde vi inte använda systemet för att ringa till varandra och prata, vilket var ett mål under hela projekttiden.

Det är vår förhoppning att den framtagna prototypen på ett eller annat sätt kommer att komma till användning hos uppdragsgivaren, exempelvis för att demonstrera hur en VoIP/PSTN-gateway kan se ut och fungera för potentiella intressenter. Det vore förstås roligt om prototypen därtill kom att vidareutvecklas till ett fullt funktionsdugligt system att använda i verkligheten.

Viktigast av allt är kanske ändå den inblick vi har fått av i hur det är att arbeta i branschen, och hur den fungerar. Inte minst hur det är att jobba i projekt, med många olika uppfattningar och viljor. Ett stort tack till TietoEnator som gjorde arbetet möjligt.

Referenser

[1] Alan B. Johnston, ”SIP: Understanding the Session Initiation Protocol”, Artech House Publishers, 2001

[2] Gonzalo Camarillo, ”SIP Demystified”, McGraw-Hill TELECOM, 2002

[3] Network Working Group, ”RFC 3261 - SIP: Session Initiation Protocol” [www] <http://www.faqs.org/rfcs/rfc3261.html> (07-02-22 08:30)

[4] Network Working Group, ”RFC 3398 - Integrated Services Digital Network (ISDN) User Part (ISUP) to Session Initiation Protocol (SIP) Mapping” [www] <http://www.faqs.org/rfcs/rfc3398.html> (07-02-22 08:30)

[5] James F. Kurose, Keith W. Ross, ”Computer Networking: A top-down approach featuring the Internet”, 3rd edition, Addison Wesley, 2005

[6] James E. Goldman, Philip T. Rawles, ”Applied Data Communications: a business-oriented approach”, 3rd edition, Wiley, 2001

[7] Dryburgh, L. & Hewett, J. 2004, Signaling System No. 7 (SS7/C7): Protocol, Architecture, and Services, .

[8] Russell, T. 2002, Signaling system #7, 4.th edn, McGraw-Hill, New York.

[9] Örtlund, J. & Bolin, N. 2001, Implementation av en ISUP-SIP gateway, Universitetet, Karlstad.

[10] Skype, http://www.skype.com

[11] Google Talk, http://www.google.com/talk/

[12] RFC 1889 - RTP

[13] Minoli, D. & Minoli, E. 2002, Delivering voice over IP networks, 2.th edn, Wiley, Indianapolis, IN.

[14] Hersent, O., Petit, J. & Gurle, D. 2005, IP telephony : deploying voice-over-IP protocols, Wiley, Chichester.

[15] ITU-T, http://www.itu.int

[16] ETSI, http://www.etsi.org/

[17] ANSI, http://www.ansi.org/

[18] Kessler, G.C., Southwick, P.V. & ebrary, I. 1998, Isdn, Signature edn, McGraw-Hill, New York ; London.

[19] Towards the Next Generation Network: The Softswitch Solution, Karl-Johan Grinnemo and Anna Brunström, Dep of CS, Karlstad University, 2006

[20] Beck, K., Andres, C. 2005, Extreme Programming Explained, Addison Wesley

[21] CUnit, http://cunit.sourceforge.net

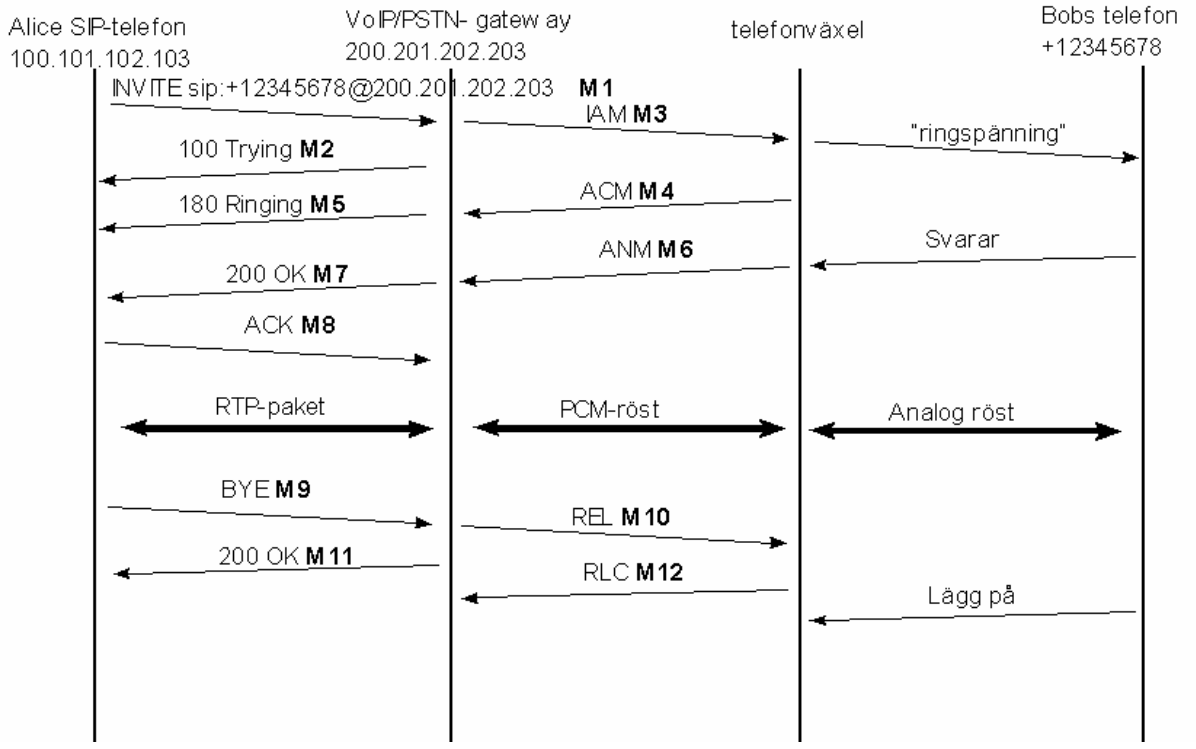
[22] CVS, www.nongnu.org/cvs/

[23] ITU-T Q.1912-5, Interworking between Session Initiation Protocol (SIP) and Bearer Independent Call Control Protocol or ISDN User Part

A Bilaga – Detaljerade exempel av samtal

Innehållet i den här bilagan baseras löst på avsnitt 9.3, *SIP to PSTN Call Through Gateway*, och 9.4, *PSTN to SIP Call Through Gateway* i [1], och använder snarlika exempel med påföljande beskrivningar. Det är bra exempel som tydligt visar hur de SIP-meddelanden som beskrevs i kapitel 2 används, och hur de mappar mot olika ISUP-meddelanden.

A.1 Internet-till-PSTN-samtal genom VoIP/PSTN-gateway



Figur A.1 - Internet-till-PSTN-samtal genom VoIP/PSTN-gateway

Alice SIP-telefon placerar det uppringda numret i en SIP URL som används i Request-URI och To-headern. Antag för enkelhets skull att SIP-telefonen har förkonfigurerats med gateway-systemets IP-adress, så att den kan skicka ett INVITE-meddelande direkt till

gateway-systemet (200.201.202.203) utan att gå via någon extra SIP-proxyserver. Gateway-systemet initierar samtalet in i PSTN genom att välja en SS7-ISUP-telefonlina till nästa telefonswitch. Informationen i SIP-headern mappas till ett IAM-meddelande. Ett ACM-meddelande (Address Complete Message) skickas tillbaka till gateway-systemet för att påvisa att linan har allokerats. Gateway-systemet översätter ACM-meddelandet till ett 180 Trying-svar.

Uppringningen är slutförd när Bob svarar, vilket gör att telefonswitchen skickar ett ANM-meddelande (Answer Message) till gateway-systemet. Ett 200 OK-svar skickas till Alice SIP-telefon, vilket innehåller SDP-information som anger på vilken port för RTP gateway-systemet kommer att skicka ljudet från PSTN. Alice SIP-telefon skickar ett ACK-meddelande för att slutföra initieringssignaleringen. Gateway-systemet gör att PSTN-ljudet kan strömma i båda riktningar.

Samtalet avslutas slutligen när Alice lägger på och hennes SIP-telefon skickar ett BYE-meddelande till gateway-systemet. BYE-meddelandet översätts till ett REL-meddelande (Release Message). Gateway-systemet skickar ett 200 OK-svar till SIP-telefonen, och mottager ett RLC-meddelande (Release Confirm Message) från PSTN.

Här nedan visas innehållet i de SIP- och ISUP-meddelanden som skickas. Se tillhörande kommentarer.

M1:

```
INVITE sip:+12345678@200.201.202.203;user=phone SIP/2.0
Via: SIP/2.0/UDP 100.101.102.103:5060
From: <+11223344@100.101.102.103;user=phone>
To: <sip:+12345678@200.201.202.203;user=phone>
Call-ID: 1234567890@100.101.102.103
CSeq: 1 INVITE
Contact: +11223344@100.101.102.103;user=phone
Content-Type: application/sdp
Content-Length: 150

v=0
o=Alice 1234567890 1234567890 IN IP4 100.101.102.103
```

```
s=Phone Call
c=IN IP4 100.101.102.103
t=0 0
m=aduio 64331 RTP/AVP 8
b=AS:64
a=rtpmap:8 PCMA/8000
```

(Kommentarer: “user=phone” anges för att indikera att SIP-URL:en innehåller ett telefonnummer)

M2:

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 100.101.102.103:5060
From: <sip:+11223344@100.101.102.103;user=phone>
To: <sip:+12345678@200.201.202.203;user=phone>
Call-ID: 1234567890@100.101.102.103
CSeq: 1 INVITE
```

M3:

```
IAM (Initial Address Message)
Called Party Number=11223344, NPI=E.164, Nature of Address = International
Calling Party Number=12345678, NPI=E.164, Nature of Address = International
USI=Speech
```

(Kommentarer: Telefonnumren i SIP-URL:erna för To och From-headern har mappats till motsvarande ISUP-parametrar. “Called Party Number” är det uppringda numret, och Calling Party Number är numret till den som ringer upp. NPI definierar internationella nummer. .)

M4:

```
ACM (Address Complete Message)
```

M5:

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 100.101.102.103:5060
```

```
From: <sip:+11223344@100.101.102.103;user=phone>
To: <sip:+12345678@200.201.202.203;user=phone>
Call-ID: 1234567890@100.101.102.103
CSeq: 1 INVITE
```

M6:

```
ANM (ANswer Message)
```

M7:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 100.101.102.103:5060
From: <sip:+11223344@100.101.102.103;user=phone>
To: <sip:+12345678@200.201.202.203;user=phone>
Call-ID: 1234567890@100.101.102.103
CSeq: 1 INVITE
Contact:  sip:+12345678@200.201.202.203;user=phone
Content-Type: application/sdp
Content-Length: 162

v=0
o=- 1234567895 1234567895 IN IP4 200.201.202.203
s=Phone Call
c=IN IP4 200.201.202.203
t=0 0
m=aduio 54321 RTP/AVP 8
b=AS:64
a=rtpmap:8 PCMA/8000
```

(Kommentar: Gateway-systemet accepterar valet av codec.)

M8

```
ACK sip:+12345678@200.201.202.203;user=phone SIP/2.0
Via: SIP/2.0/UDP 100.101.102.103:5060
From: <sip:+11223344@100.101.102.103;user=phone>
To: <sip:+12345678@200.201.202.203;user=phone>
Call-ID: 1234567890@100.101.102.103
```

```
CSeq: 1 INVITE
```

M9:

```
BYE sip:+12345678@200.201.202.203;user=phone SIP/2.0
Via: SIP/2.0/UDP 100.101.102.103:5060
From: <sip:+11223344@100.101.102.103;user=phone>
To: <sip:+12345678@200.201.202.203;user=phone>
Call-ID: 1234567890@100.101.102.103
CSeq: 2 BYE
```

(Kommentarer: Cseq har inkrementerats.)

M10:

```
REL (RELease message)
```

`CauseValue=16` (Kommentar: ”normal call clearing”.)

M11:

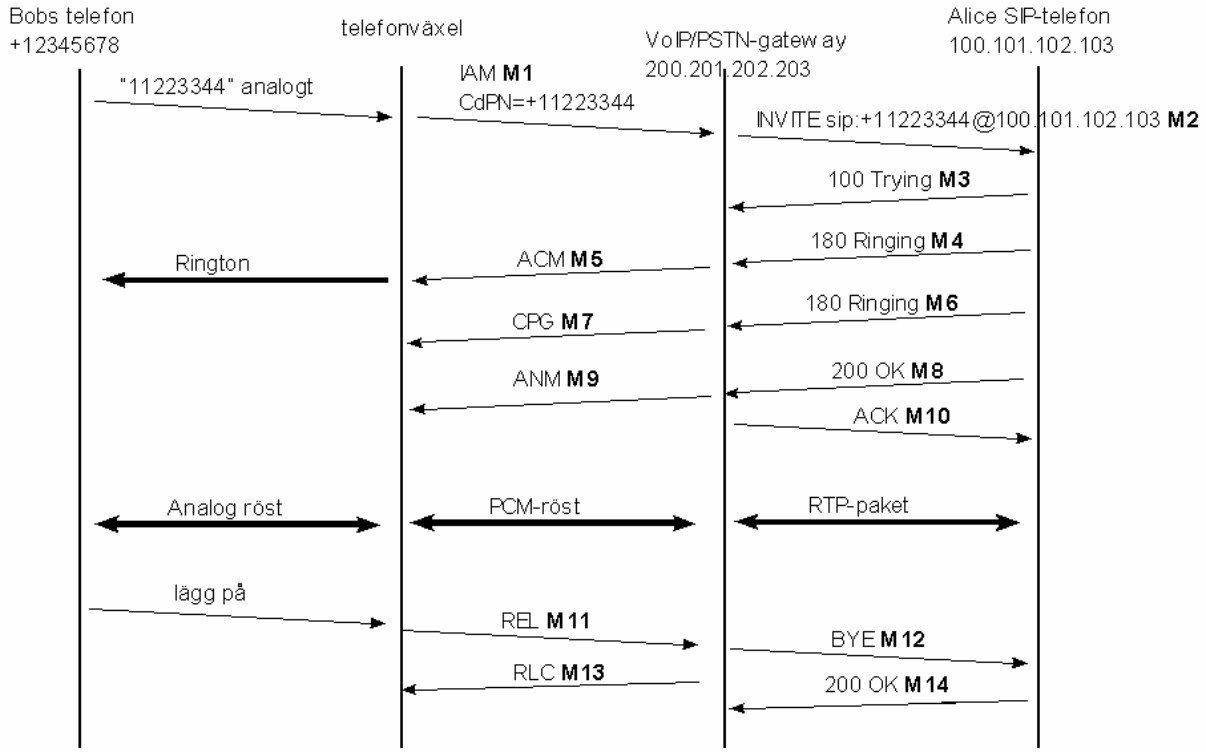
```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 100.101.102.103:5060
From: <sip:+11223344@100.101.102.103;user=phone>
To: <sip:+12345678@200.201.202.203;user=phone>
Call-ID: 1234567890@100.101.102.103
CSeq: 2 BYE
```

M12:

```
RLC (ReLease Complete message)
```

A.2 PSTN-till-Internet-samtal genom VoIP/PSTN-gateway

Här visas motsvarande scenario för uppringning från PSTN till Internet. Antag att Bobs telefonnummer är +12345678 och Alice har numret +11223344. Bob Ringer upp Alice, och efter att samtalet är uppkopplat väljer han att lägga på luren, varför samtalet kopplas ner.



Figur A.2 - PSTN-till-Internet-samtal genom VoIP/PSTN-gateway

M1:

```
IAM (Initial Address Message)
Called Party Number=12345678, NPI=E.164, Nature of Address =
International
Calling Party Number=11223344, NPI=E.164, Nature of Address =
International
USI=Speech
```

(Kommentarer: “Called Party Number” är det uppringda numret, och Calling Party Number är numret till den som ringer upp. NPI definierar internationella nummer.)

M2:

```
INVITE sip:+11223344@100.101.102.103;user=phone SIP/2.0
Via: SIP/2.0/UDP 200.201.202.203:5060
From: <sip:+12345678@200.201.202.203;user=phone>
To: <sip:+11223344@100.101.102.103;user=phone>
Call-ID: 1234567890@200.201.202.203
CSeq: 1 INVITE
Contact:  sip:+12345678@200.201.202.203;user=phone
Content-Type: application/sdp
Content-Length: 162

v=0
o=- 1234567890 1234567890 IN IP4 200.201.202.203
s=Phone Call
c=IN IP4 200.201.202.203
t=0 0
m=aduio 62331 RTP/AVP 8
b=AS:64
a=rtpmap:8 PCMA/8000
```

(Kommentarer: Det uppringda numret har mappats till en SIP-URL. “user=phone” anges för att indikera att SIP-URL:en innehåller ett telefonnummer. SDP-lasten indikerar att den media som förväntas är PCM A-law-kodad, och tas emot som RTP-trafik på port 62331.)

M3:

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 200.201.202.203:5060
From: <sip:+12345678@200.201.202.203;user=phone>
To: <sip:+11223344@100.101.102.103;user=phone>
Call-ID: 1234567890@200.201.202.203
CSeq: 1 INVITE
```

M4:

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 200.201.202.203:5060
From: <sip:+12345678@200.201.202.203;user=phone>
To: <sip:+11223344@100.101.102.103;user=phone>
Call-ID: 1234567890@200.201.202.203
CSeq: 1 INVITE
```

M5:

```
ACM (Address Complete Message)
Interworking Indicator=”interworking encountered”
ISUP Indicator=”ISUP not used all the way”
ISDN Access Indicator=”terminating access non-ISDN”
```

M6:

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 200.201.202.203:5060
From: <sip:+12345678@200.201.202.203;user=phone>
To: <sip:+11223344@100.101.102.103;user=phone>
Call-ID: 1234567890@200.201.202.203
CSeq: 1 INVITE
```

M7:

```
CPG (Call ProGress message)
Event indicator=”alerting”
```

(Kommentarer: Om ett ACM-meddelande redan har skickats skickas CPG-meddelanden.)

M8:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 200.201.202.203:5060
From: <sip:+12345678@200.201.202.203;user=phone>
To: <sip:+11223344@100.101.102.103;user=phone>
Call-ID: 1234567890@200.201.202.203
CSeq: 1 INVITE
Contact:  sip:+11223344@100.101.102.103;user=phone
```

```
Content-Type: application/sdp
Content-Length: 162

v=0
o=Alice 1234567895 1234567895 IN IP4 100.101.102.103
s=Phone Call
c=IN IP4 100.101.102.103
t=0 0
m=aduio 54321 RTP/AVP 8
b=AS:64
a=rtpmap:8 PCMA/8000
```

(Kommentar: Alice accepterar valet av codec.)

M9:

```
ANM (ANswer Message)
```

M10:

```
ACK sip:+11223344@100.101.102.103;user=phone SIP/2.0
Via: SIP/2.0/UDP 200.201.202.203:5060
From: <sip:+12345678@200.201.202.203;user=phone>
To: <sip:+11223344@100.101.102.103;user=phone>
Call-ID: 1234567890@200.201.202.203
CSeq: 1 INVITE
```

M11:

```
REL (RELease message)
CauseValue=16
```
(Kommentar: ”normal call clearing”.)

M12:

```
BYE sip:+11223344@100.101.102.103;user=phone SIP/2.0
Via: SIP/2.0/UDP 200.201.202.203:5060
From: <sip:+12345678@200.201.202.203;user=phone>
To: <sip:+11223344@100.101.102.103;user=phone>
Call-ID: 1234567890@200.201.202.203
CSeq: 2 BYE
```

(Kommentarer: Cseq har inkrementerats.)

M13:

```
RLC (ReLease Complete message)
```

M14:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 200.201.202.203:5060
From: <sip:+12345678@200.201.202.203;user=phone>
To: <sip:+11223344@100.101.102.103;user=phone>
Call-ID: 1234567890@200.201.202.203
CSeq: 2 BYE
```

B Bilaga – Testfall

Så som tidigare har beskrivits (avsnitt 3.7) så omsluter testmiljön gateway-applikationen och kontrollerar således flödet av primitiver till och från gateway-applikationen. Syntaxen för ett kommando i testmiljön är <SKICKA/MOTTA>(<MODUL>, <PRIMITIV>) där MODUL är den modul som primitiven PRIMITIV härstammar från eller är ämnat åt beroende på om kommandot var SKICKA eller MOTTA. Exempelvis kommandot *SKICKA(ISUP, IAM)* som alltså utläses ”Skicka (till gateway-applikationen) en IAM-primitiv som härstammar från ISUP-modulen”, och *MOTTA(SIP, INVITE)* utläses ”Motta en INVITE-primitiv (från gateway-applikationen) ämnat åt SIP-modulen”. Nedan följer kodbeskrivningar av de testfall som skrevs för gateway-applikationen, och även de testfall som gjordes för mediakonverteraren. Om ett eller flera testfall ligger som grund för ytterligare testfall kan man göra dessa testfall till förkrav för att testa det nya testfallet. Syntaxen för detta kommando är FÖRTEST(<LISTA>) där LISTA är en kommaseparerad lista med namn på de testfall som måste gå igenom för att det aktuella testfallet skall köras. De testfall som ges i denna lista körs alltså innan det aktuella testet körs, och kan ses som förvillkor för testfallet.

De moduler som förekommer i testfallen är ISUP-modulen (ISUP), SIP-modulen (SIP), mediakonverteraren (MC) och gateway-applikationen (GATEWAY).

Namn	Kategori	Resultat
xts01t1	Interntest	Godkänt

Tabell B.1 - Testfallet xts01t1

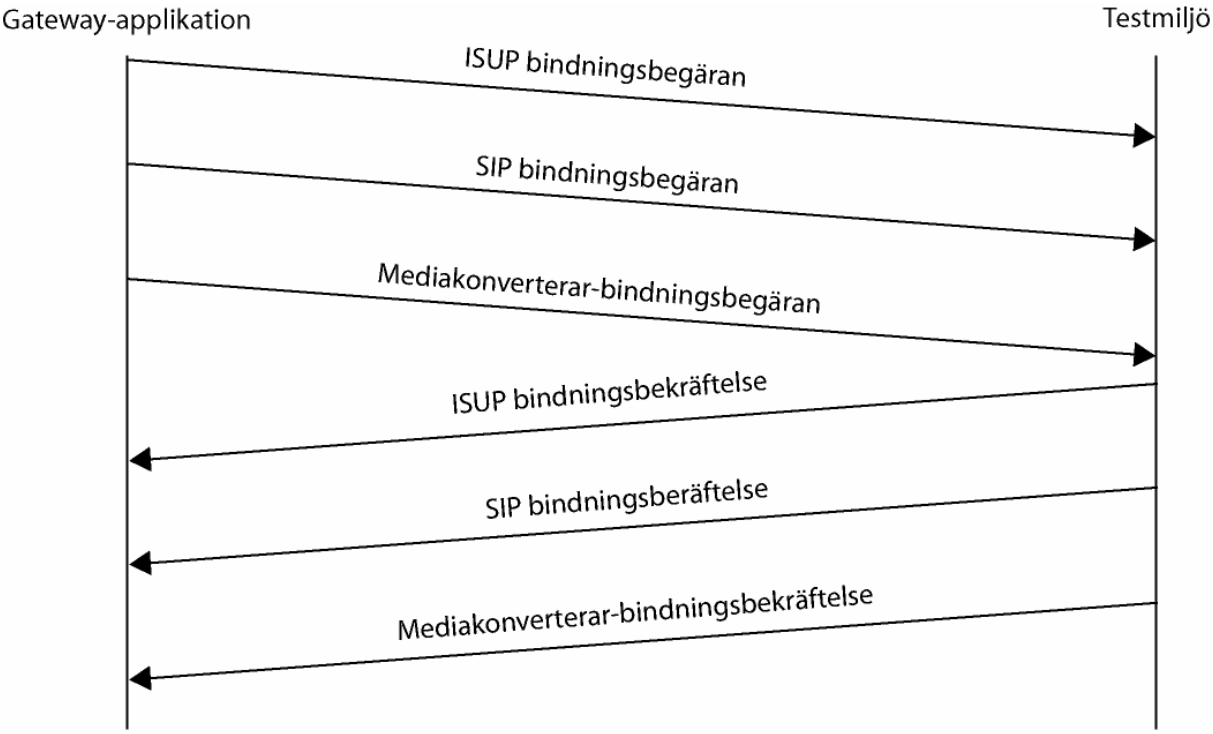
`Syfte:`

Testar det typiska bindningsförfarandet mellan gateway-applikationen, SIP-, ISUP-modulen och mediakonverteraren.

`Kod:`

```
MOTTA(ISUP, ISUP bindningsbegäran)
MOTTA(SIP, SIP bindningsbegäran)
MOTTA(MC, Mediakonverterar-bindningsbegäran)
```

```
SKICKA(ISUP, ISUP bindningsbekräftelse)
SKICKA(SIP, SIP bindningsbekräftelse)
SKICKA(MC, Mediakonverterar-bindningsbekräftelse)
```



Figur B.1 - Testfallet xts01t1

Namn	Kategori	Resultat
xts01t2	Interntest	Godkänt

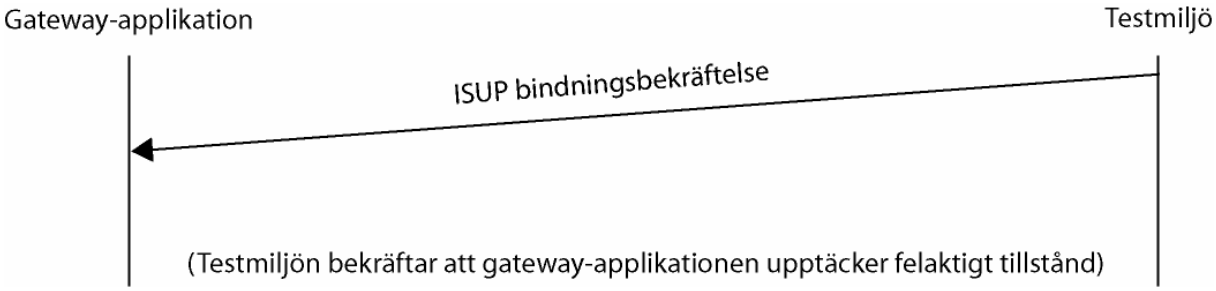
Tabell B.2 - Testfallet xts01t2

`Syfte:`

Testmiljön prövar att skicka en (för gateway-applikationen) oväntad primitiv inväntar att gateway-applikationen ska reagera på detta.

`Kod:`

```
SKICKA(ISUP, ISUP bindningsbekräftelse)
```



Figur B.2 - Testfallet xts01t2

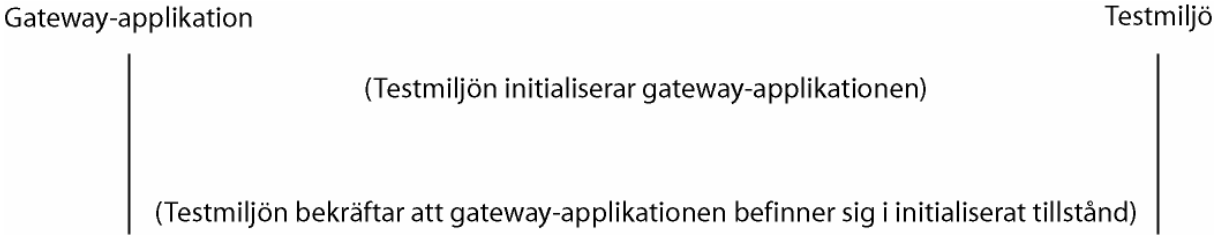
Namn	Kategori	Resultat
xts01_go_to_operational_state_Initiated	Interntest	Godkänt

Tabell B.3 - Testfallet xts01_go_to_operational_state_Initiated

`Syfte:`
Testmiljön aktiverar gateway-applikationen och bekräftar att den befinner sig i initierat tillstånd.

`Kod:`

`(Inga primitiver skickas)`



Figur B.3 - Testfallet xts01_go_to_operational_state_Initiated

Namn	Kategori	Resultat
xts01_go_to_operational_state_Started	Interntest	Godkänt

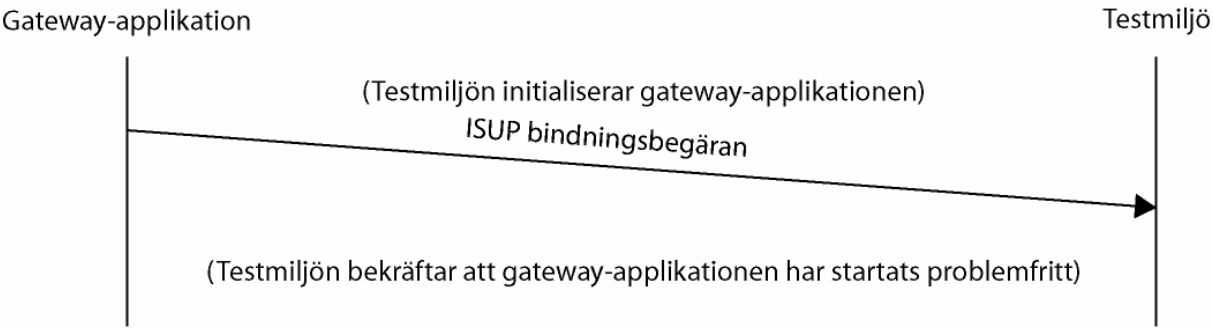
Tabell B.4 - Testfallet xts01_go_to_operational_state_Started

Syfte:

Testmiljön initialiserar gateway-applikationen, skickar en startbegäran (och försätter således gateway-applikationen i startat tillstånd) och bekräftar att gateway-applikationen börjar skicka en ISUP-bindningsbegäran och befinner sig i startat tillstånd.

Kod:

```
MOTTA(ISUP, ISUP bindningsbegäran)
```



Figur B.4 - Testfallet xts01_go_to_operational_state_Started

Namn	Kategori	Resultat
xts01_go_to_operational_state_Operational	Interntest	Godkänt

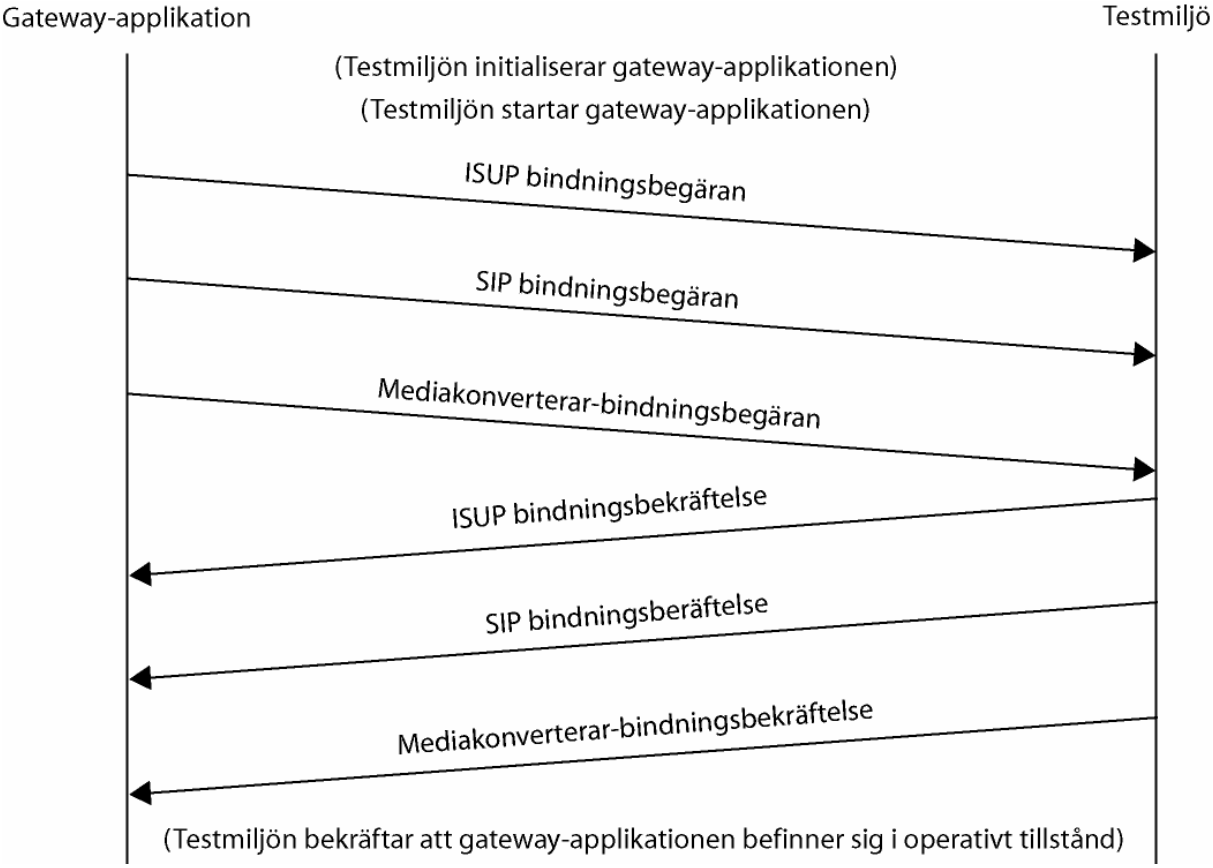
Tabell B.5 - Testfallet xts01_go_to_operational_state_Operational

Syfte:

Testmiljön binder alla moduler och bekräftar att gateway-applikationen befinner sig i operativt tillstånd.

Kod:

```
MOTTA(ISUP, ISUP bindningsbegäran)
MOTTA(SIP, SIP bindningsbegäran)
MOTTA(MC, Mediakonverterar-bindningsbegäran)
SKICKA(ISUP, ISUP bindningsbekräftelse)
SKICKA(SIP, SIP bindningsbekräftelse)
SKICKA(MC, Mediakonverterar-bindningsbekräftelse)
```



Figur B.5 - Testfallet xts01_go_to_operational_state_Operational

Namn	Kategori	Resultat
xts01_go_to_operational_state_Operational_large_config	Interntest	Godkänt

Tabell B.6 - Testfallet xts01_go_to_operational_state_Operational_large_config

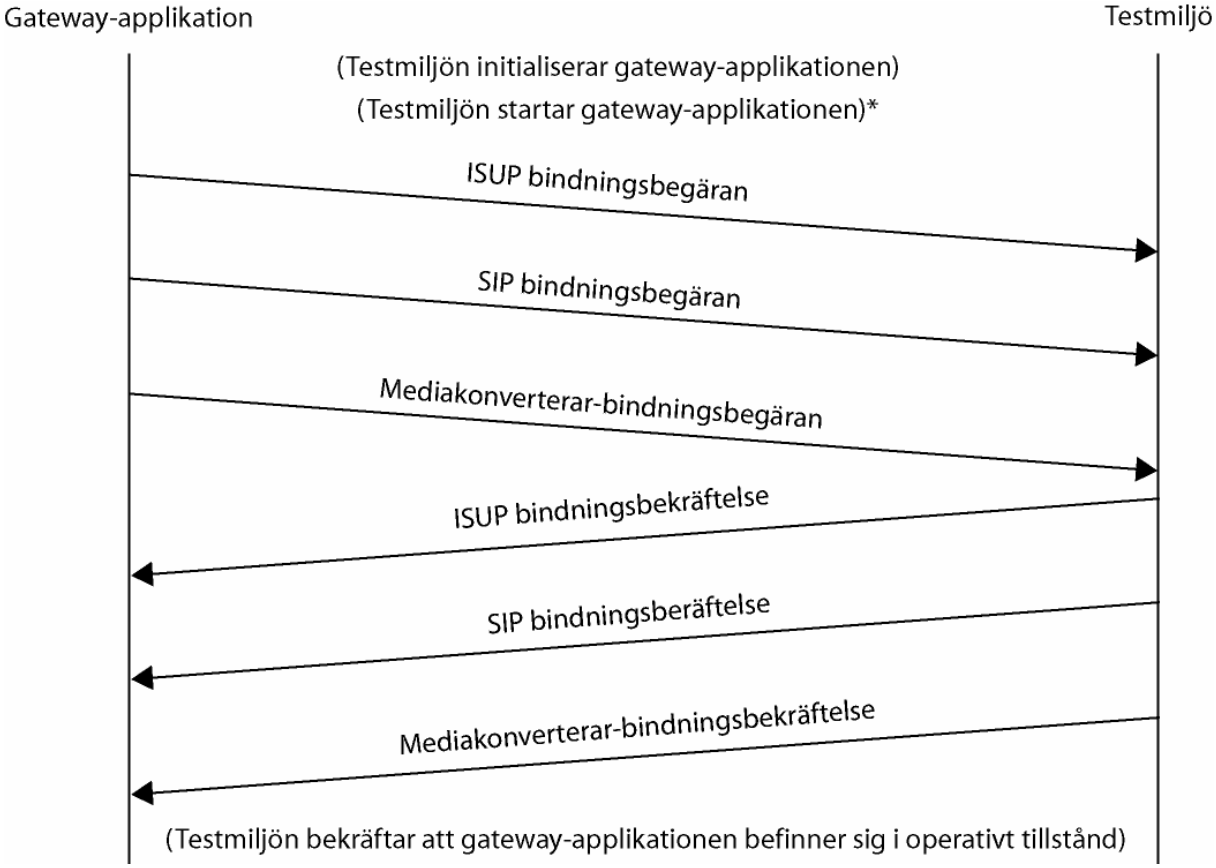
`Syfte:`

Testmiljön instruerar gateway-applikationen att läsa in en konfiguration som allokerar maximala resurser till gateway-applikationen. Testmiljön testar sedan så att gateway-applikationen kan hantera detta.

`Kod:`

```
MOTTA(ISUP, ISUP bindningsbegäran)
MOTTA(SIP, SIP bindningsbegäran)
```

```
MOTTA(MC, Mediakonverterar-bindningsbegäran)

SKICKA(ISUP, ISUP bindningsbekräftelse)

SKICKA(SIP, SIP bindningsbekräftelse)

SKICKA(MC, Mediakonverterar-bindningsbekräftelse)
*Testmiljön startar gateway-applikationen och instruerar den att läsa
in en konfigurationsfil där de maximala resurserna för applikationen
allokeras.
```



Figur B.6 - Testfallet xts01_go_to_operational_state_Operational_large_config

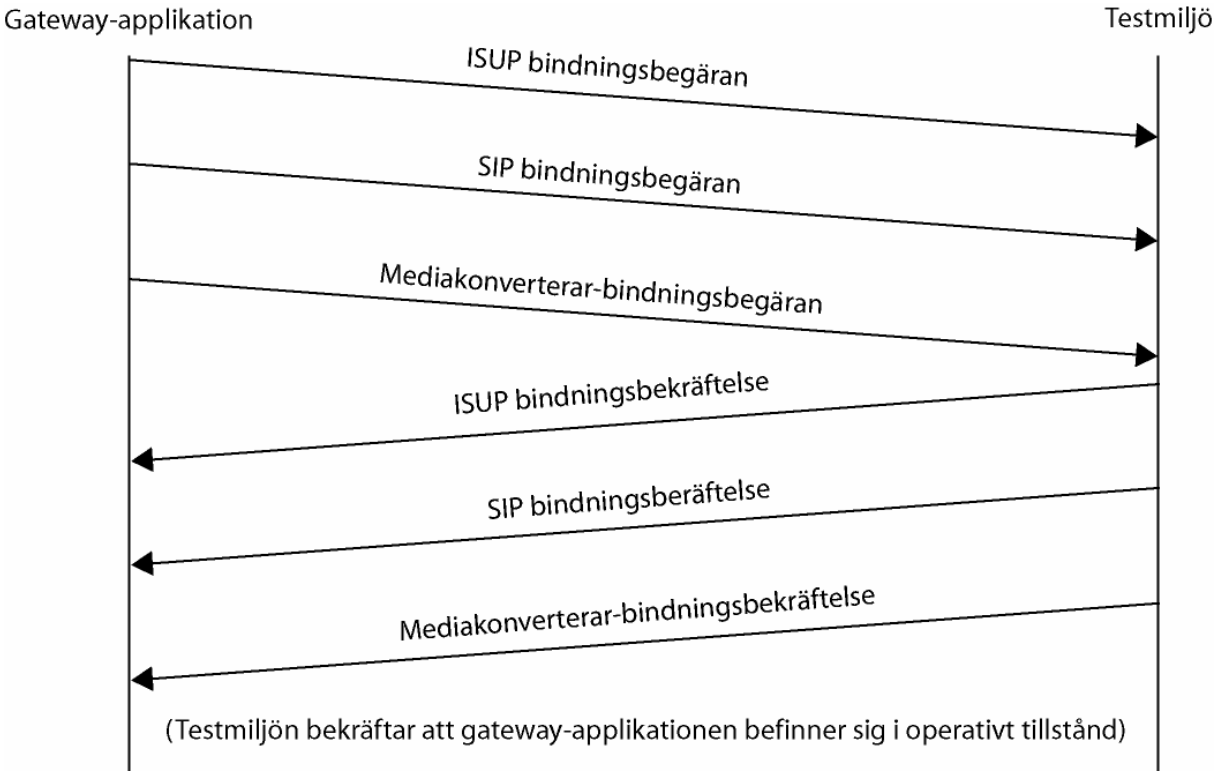
Namn	Kategori	Resultat
xts01t4	Interntest	Godkänt

Tabell B.7 - Testfallet xts01t4

```
Syfte:
```

Bekräftar att gateway-applikationen befinner sig i operativt tillstånd efter att bundits till SIP- och ISUP-modulerna samt mediakonverteraren.

```
Kod:
    MOTTA(ISUP, ISUP bindningsbegäran)
    MOTTA(SIP, SIP bindningsbegäran)
    MOTTA(MC, Mediakonverterar-bindningsbegäran)
    SKICKA(ISUP, ISUP bindningsbekräftelse)
    SKICKA(SIP, SIP bindningsbekräftelse)
    SKICKA(MC, Mediakonverterar-bindningsbekräftelse)
```



Figur B.7 - Testfallet xts01t4

Namn	Kategori	Resultat
xts02t1	Uppkoppling	Godkänt

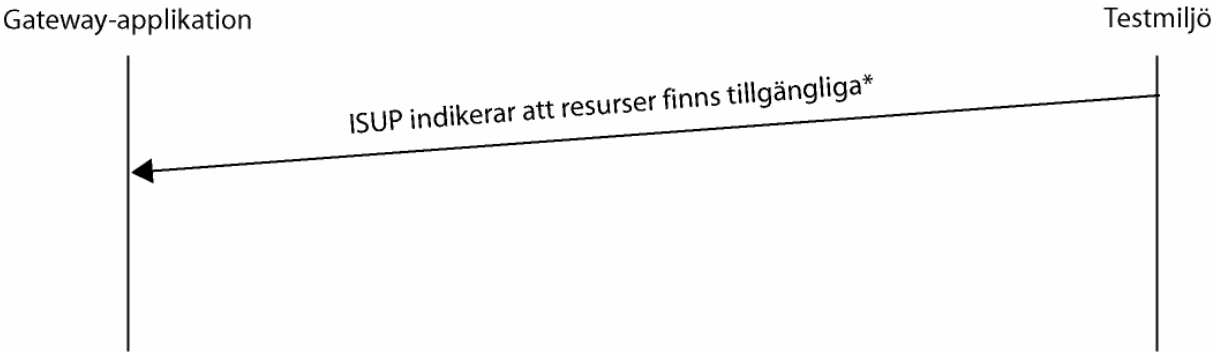
Tabell B.8 - Testfallet xts02t1

```
Syfte:
```

Simulerar att en resursindikering skickas från ISUP-modulen till gateway-applikationen (görs efter att en bindningsbekräftelse skickats av ISUP-modulen).

Kod:

```
FÖRTEST(xts01_go_to_operational_state_Operational)

SKICKA(ISUP, ISUP resursindikering)
```



Figur B.8 - Testfallet xts02t1

Namn	Kategori	Resultat
xts02_go_to_call_state_Idle	Interntest	Godkänt

Tabell B.9 - Testfallet xts02_go_to_call_state_Idle

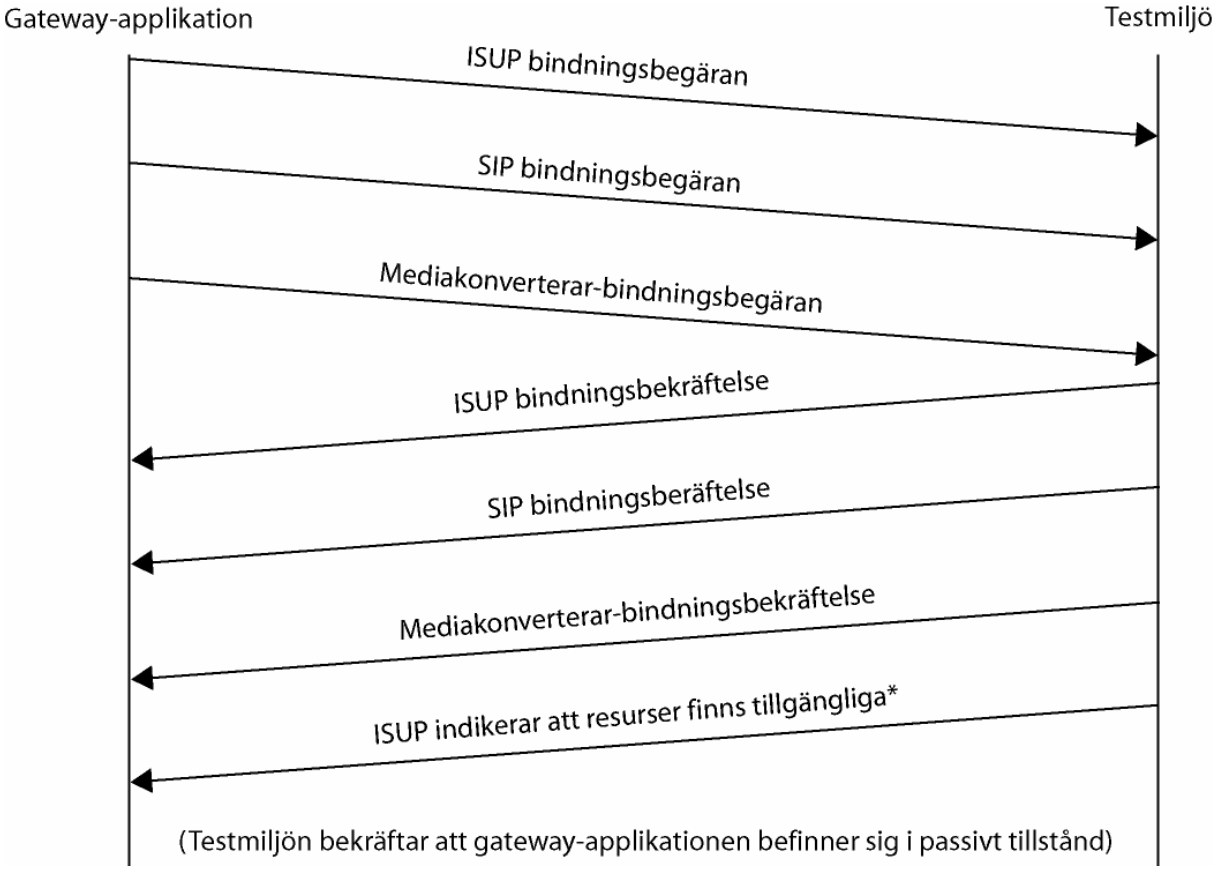
Syfte:

Binder som vanligt, skickar en resursindikering och bekräftar att gateway-applikationen befinner sig i passivt tillstånd.

Kod:

```
MOTTA(ISUP, ISUP bindningsbegäran)

MOTTA(SIP, SIP bindningsbegäran)

MOTTA(MC, Mediakonverterar-bindningsbegäran)

SKICKA(ISUP, ISUP bindningsbekräftelse)

SKICKA(SIP, SIP bindningsbekräftelse)

SKICKA(MC, Mediakonverterar-bindningsbekräftelse)
* När ISUP-modulen har bundits till gateway-applikationen så skickas en
speciell primitiv som indikerar att ISUP-modulen har allokerat
```

```
nödvändiga resurser.
SKICKA(ISUP, ISUP resursindikering)
```



Figur B.9 - Testfallet xts02_go_to_call_state_Idle

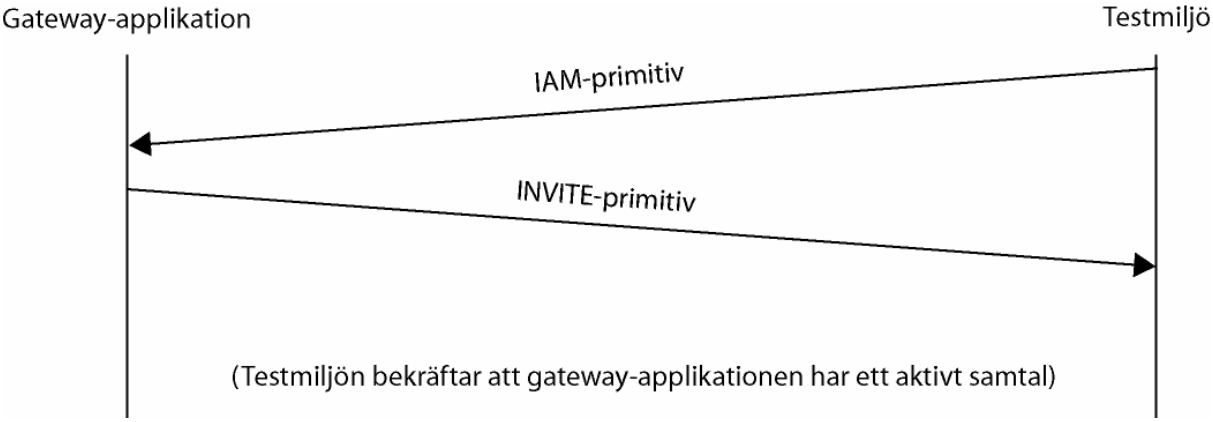
Namn	Kategori	Resultat
xts03t1	Uppkoppling	Godkänt

Tabell B.10 - Testfallet xts03t1

```
Syfte:
```
Testmiljön skickar en IAM från ISUP-sidan och bekräftar att en INVITE skickas till SIP-sidan med rätt parametrar.

```
Kod:
    FÖRTEST(xts01t1, xts02t1)
    SKICKA(ISUP, IAM-primitiv)
```

```
MOTTA(SIP, INVITE-primitiv)
```



Figur B.10 - Testfallet xts03t1

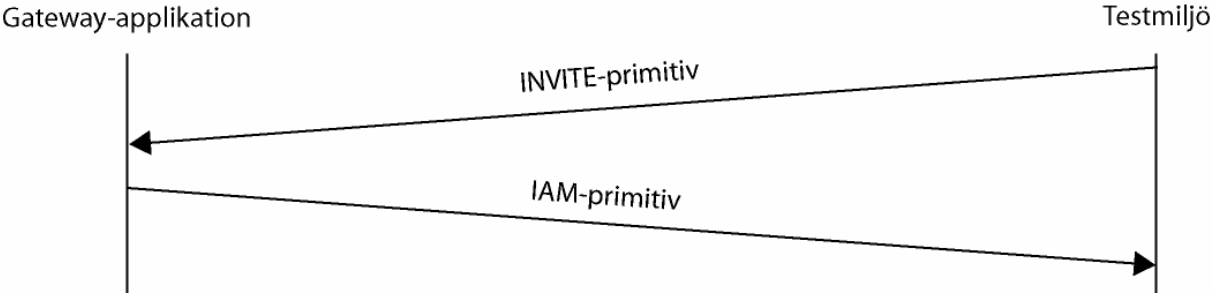
Namn	Kategori	Resultat
xts05t1	Uppkoppling	Ej godkänt

Tabell B.11 – Testfallet xts05t1

Syfte:
Testmiljön skickar en INVITE från SIP-sidan och tar emot en IAM ämnat åt ISUP-sidan

Kod:

```
FÖRTEST(xts01t1, xts02t1)

SKICKA(SIP, INVITE-primitiv)

MOTTA(ISUP, IAM-primitiv)
```



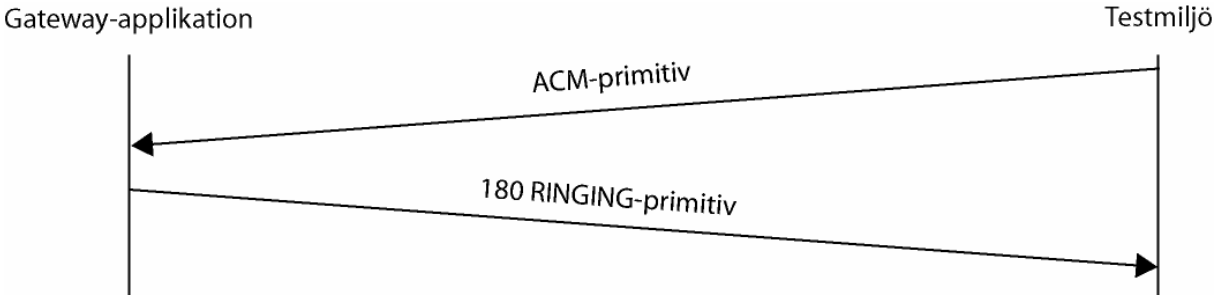
Figur B.11 – Testfallet xts05t1

Namn	Kategori	Resultat
xts05t2	Uppkoppling	Ej godkänt

Tabell B.12 – Testfallet xts05t2

Syfte:
Testmiljön skickar en ACM från ISUP-sidan och tar emot en 180 RINGING ämnat åt SIP-sidan

Kod:
```
FÖRTEST(xts01t1, xts02t1, xts05t1)

SKICKA(ISUP, ACM-primitiv)

MOTTA(SIP, 180 RINGING-primitiv)
```



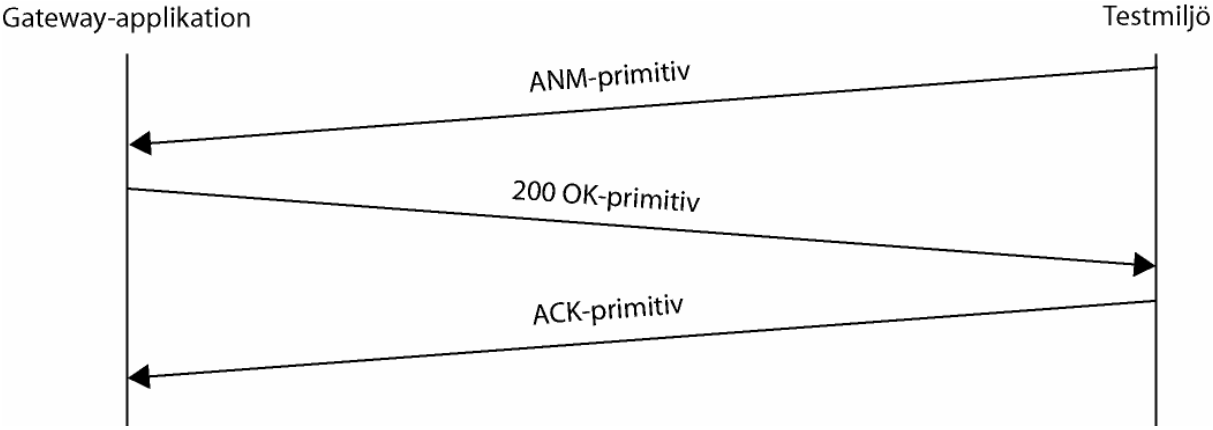
Figur B.12 – Testfallet xts05t2

Namn	Kategori	Resultat
xts05t3	Uppkoppling	Ej godkänt

Tabell B.13 – Testfallet xts05t3

Syfte:
Testmiljön skickar en INVITE från SIP-sidan och tar emot en IAM ämnat åt ISUP-sidan

Kod:
```
FÖRTEST(xts01t1, xts02t1, xts05t1, xts05t2)

SKICKA(ISUP, ANM-primitiv)

MOTTA(SIP, 200 OK-primitiv)

SKICKA(SIP, ACK-primitiv)
```



Figur B.13 – Testfallet xts05t3

Namn	Kategori	Resultat
xts05t5	Uppkoppling	Godkänt

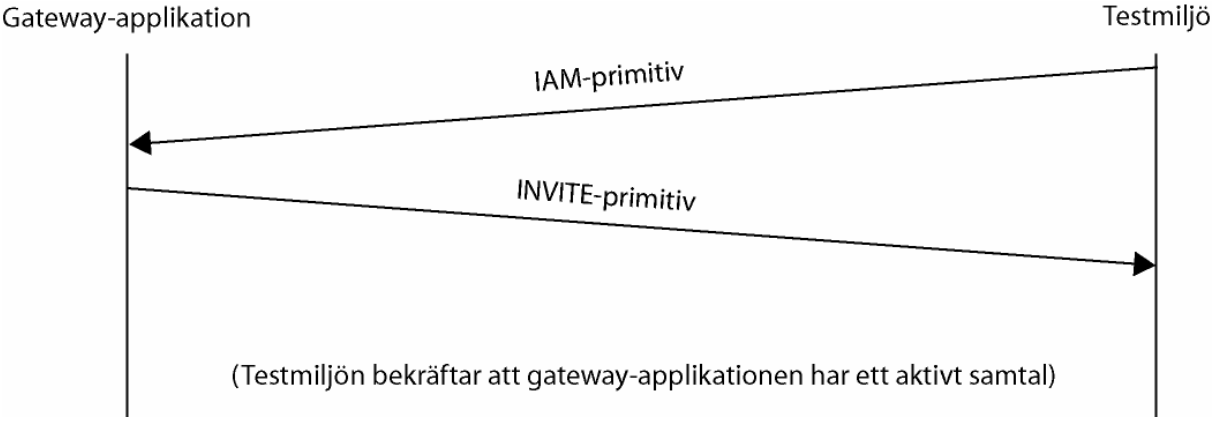
Tabell B.14 - Testfallet xts05t5

Syfte:

Testmiljön skickar en IAM från ISUP-sidan och bekräftar att en INVITE skickas till SIP-sidan.

Kod:

```
FÖRTEST(xts01t1, xts02t1)
SKICKA(ISUP, IAM-primitiv)
MOTTA(SIP, INVITE-primitiv)
```



Figur B.14 - Testfallet xts05t5

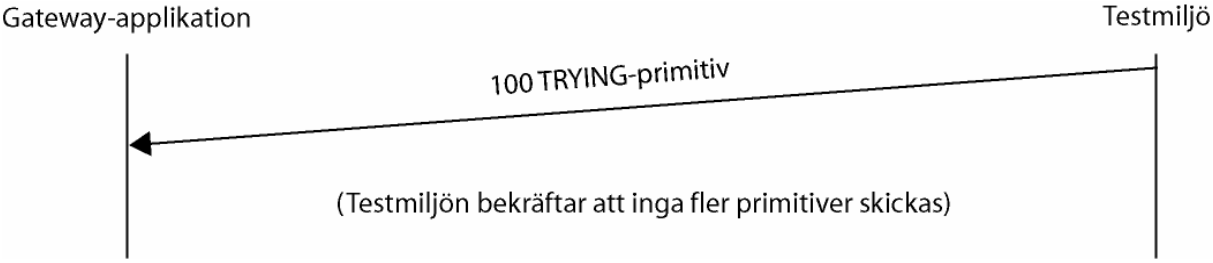
Namn	Kategori	Resultat
xts05t6	Uppkoppling	Godkänt

Tabell B.15 - Testfallet xts05t6

`Syfte:`
Testmiljön simulerar att en 100 trying anländer.

`Kod:`

```
FÖRTEST(xts01t1, xts02t1, xts05t5)

SKICKA(SIP, 100 TRYING-primitiv)
```



Figur B.15 - Testfallet xts05t6

Namn	Kategori	Resultat
xts05t7	Uppkoppling	Godkänt

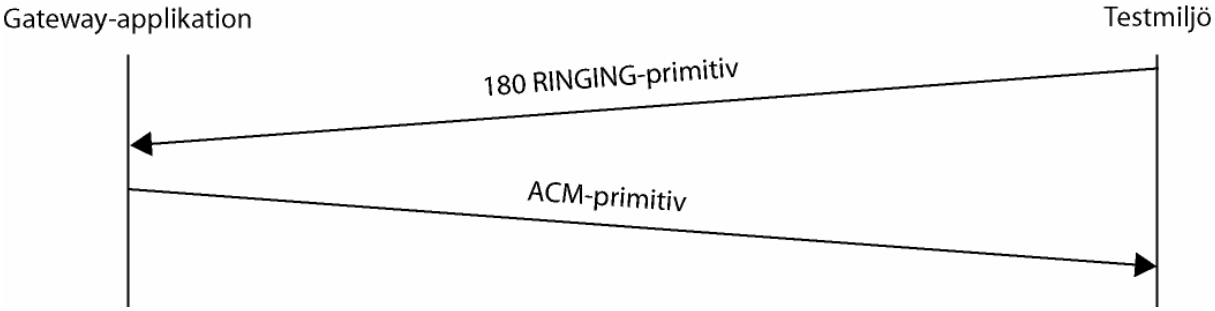
Tabell B.16 - Testfallet xts05t7

Syfte:

Testmiljön simulerar att en 180 RINGING-primitiv kommer, bekräftar att en ACM-primitiv skickas från ISUP-sidan.

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts05t5, xts05t6)
SKICKA(SIP, 180 RINGING-primitiv)
MOTTA(ISUP, ACM-primitiv)
```



Figur B.16 - Testfallet xts05t7

Namn	Kategori	Resultat
xts05t8	Uppkoppling	Godkänt

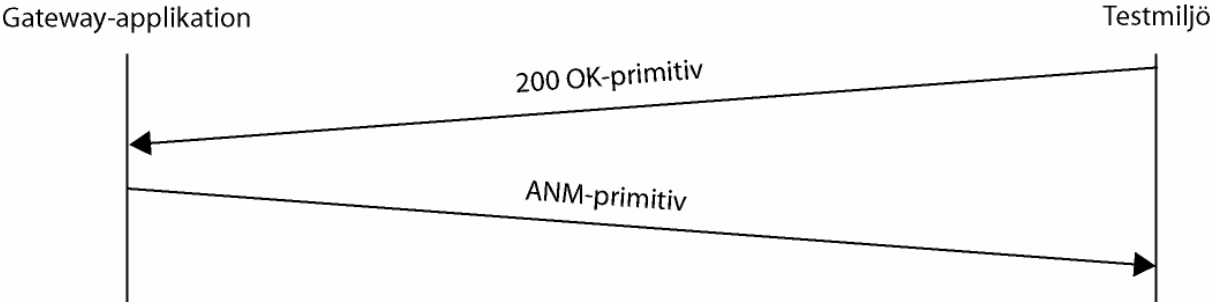
Tabell B.17 - Testfallet xts05t8

Syfte:

Testmiljön simulerar att en 200 OK kommer, bekräftar att en ANM skickas från ISUP-sidan.

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts05t5, xts05t6, xts05t7)
SKICKA(SIP, 200 OK-primitiv)
MOTTA(ISUP, ANM-primitiv)
```



Figur B.17 - Testfallet xts05t8

Namn	Kategori	Resultat
xts05t11	Nedkoppling	Godkänt

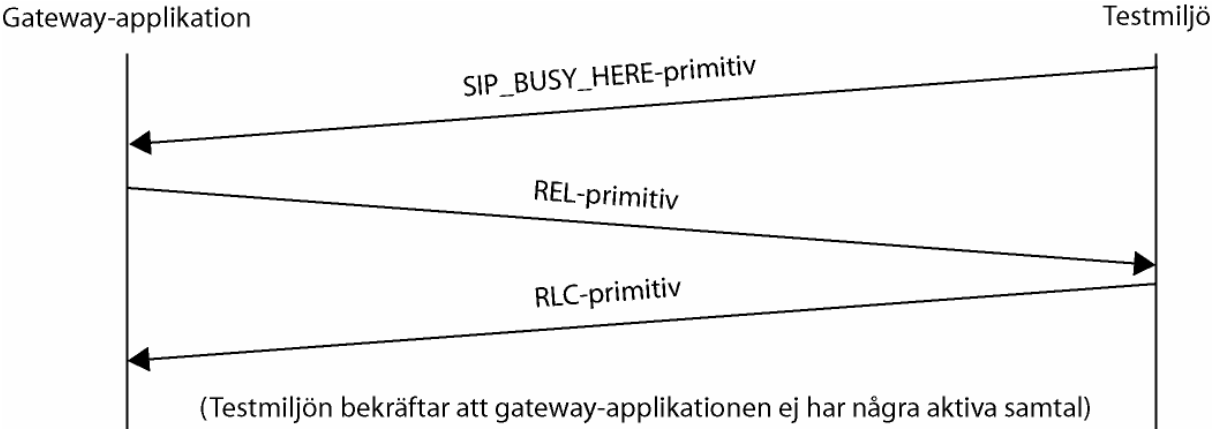
Tabell B.18 - Testfallet xts05t11

Syfte:

Testar gateway-applikationens reaktion på en SIP-initierad nedkoppling av ett samtal med anledning av att SIP-användaren är upptagen. Den förväntade reaktionen är att samtalet ej kan kopplas upp och gateway-applikationen skickar nedkopplingsbegäran till ISUP-modulen.

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts05t5, xts05t6)
SKICKA(SIP, SIP_BUSY_HERE-primitiv)
MOTTA(ISUP, REL-primitiv)
SKICKA(ISUP, RLC-primitiv)
```



Figur B.18 - Testfallet xts05t11

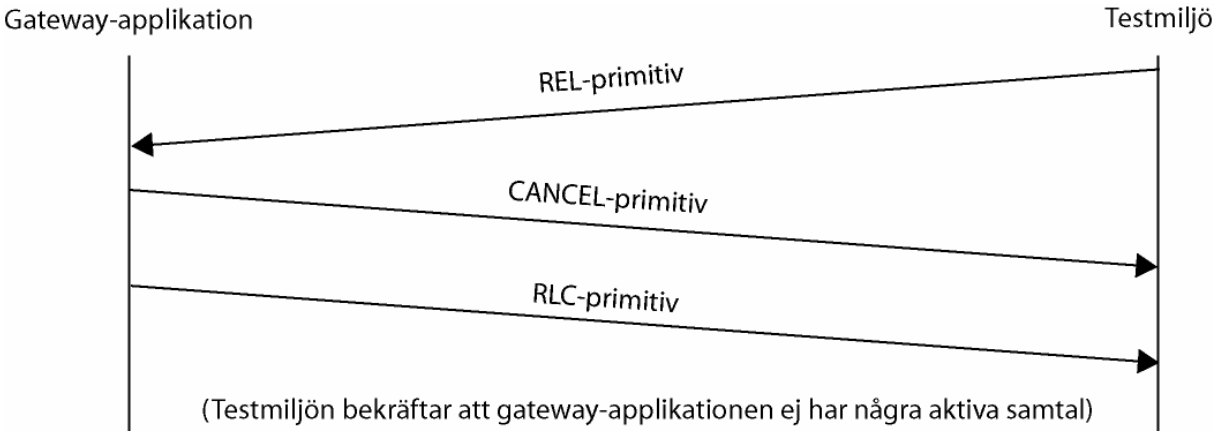
Namn	Kategori	Resultat
xts05t14	Nedkoppling	Godkänt

Tabell B.19 - Testfallet xts05t14

Syfte:
Simulerar hur ett ISUP-initierat samtal avbryts i uppkopplingsfasen (av ISUP).

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts05t5, xts05t6, xts05t7)
SKICKA(ISUP, REL-primitiv)
MOTTA(SIP, CANCEL-primitiv)
MOTTA(ISUP, RLC-primitiv)
```



Figur B.19 - Testfallet xts05t14

Namn	Kategori	Resultat
xts09t4	Uppkoppling	Godkänt

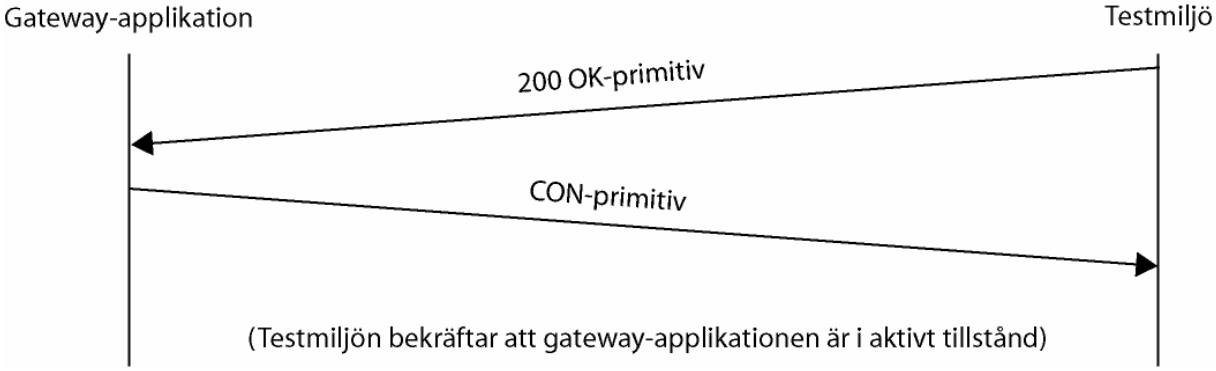
Tabell B.20 - Testfallet xts09t4

`Syfte:`

Simulerar snabb uppkoppling där en 200-primitiv skickas innan någon 180-primitiv, och testmiljön bekräftar sedan att en CON-primitiv skickas från ISUP-sidan.

`Kod:`

```
FÖRTEST(xts01t1, xts02t1, xts03t1, xts05t6)
SKICKA(SIP, 200 OK-primitiv)
MOTTA(ISUP, CON-primitiv)
```



Figur B.20 - Testfallet xts09t4

Namn	Kategori	Resultat
xts10t1	Uppkoppling	Godkänt

Tabell B.21 - Testfallet xts10t1

Syfte:

Testar meddelandesekvensen när SIP-sidan skickar en 100 TRYING för att informera ISUP-sidan att samtalet fortskrider.

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts03t1)
SKICKA(SIP, 180 RINGING-primitiv)
```

Figur B.21 - Testfallet xts10t1

Namn	Kategori	Resultat
xts10t2	Uppkoppling	Godkänt

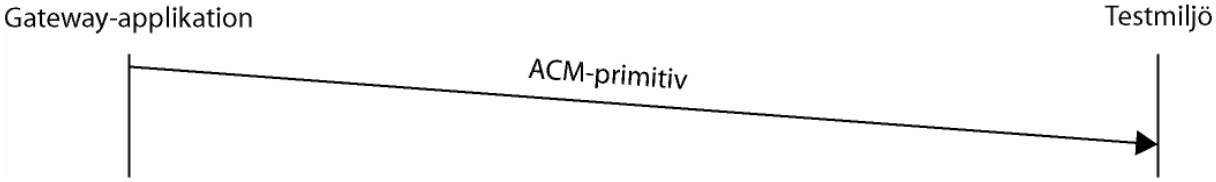
Tabell B.22 - Testfallet xts10t2

Syfte:

Bekräftar att en ISUP-primitiv skickas (ACM-primitiv) som svar på xts10t1.

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts03t1, xts10t1)
MOTTA(ISUP, ACM-primitiv)
```



Figur B.22 - Testfallet xts10t2

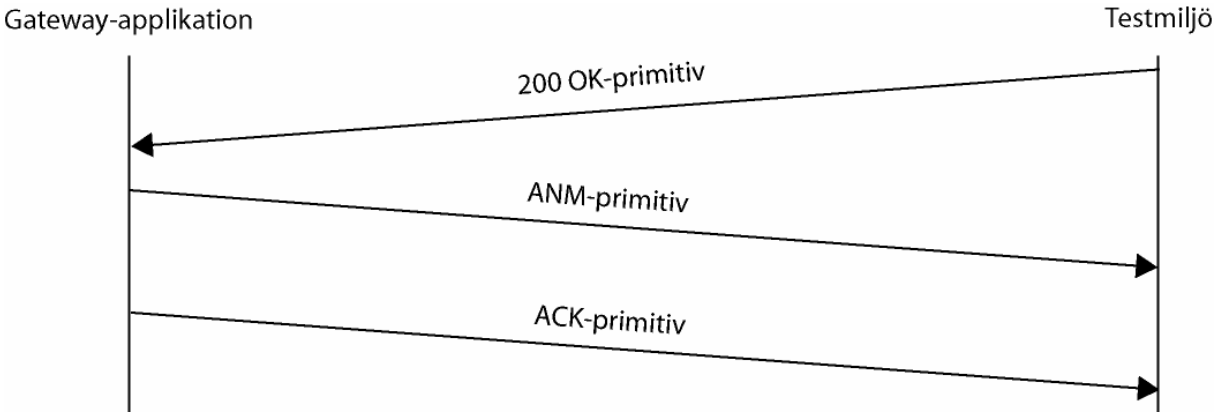
Namn	Kategori	Resultat
xts10t3	Uppkoppling	Godkänt

Tabell B.23 - Testfallet xts10t3

Syfte:
Simulerar den sista delen av uppkopplingsfasen mellan SIP och ISUP.

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts03t1, xts10t1, xts10t2)
SKICKA(SIP, 200 OK-primitiv)
MOTTA(ISUP, ANM-primitiv)
MOTTA(SIP, ACK-primitiv)
( Ej med i test: MOTTA(MC, Mediakonverterarkopplingsbegäran) )
```



Figur B.23 - Testfallet xts10t3

Namn	Kategori	Resultat
xts10t4	Nedkoppling	Godkänt

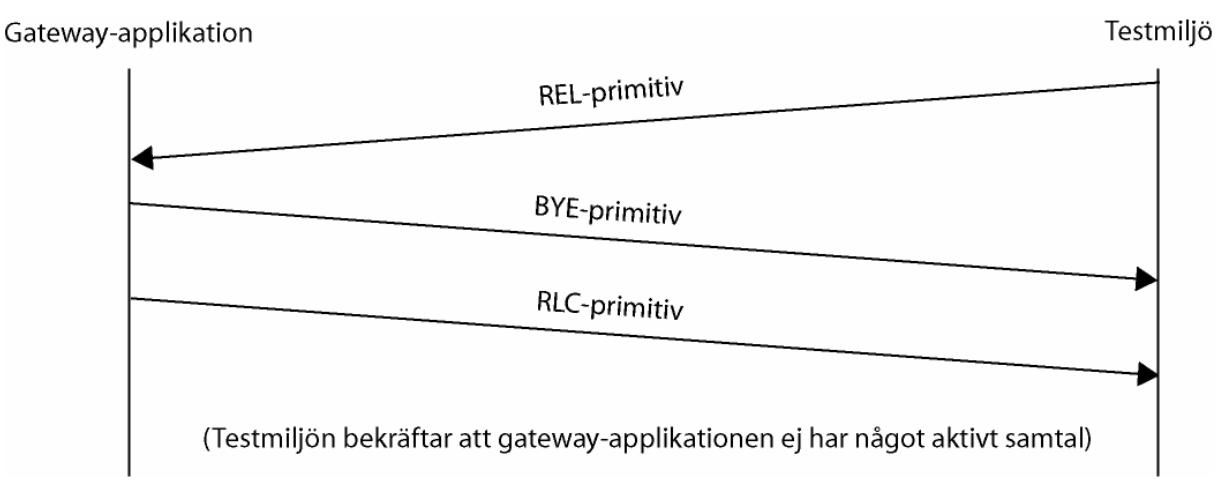
Tabell B.24 - Testfallet xts10t4

`Syfte:`

Simulerar hur ISUP-sidan initierar nedkoppling av ett samtal.

`Kod:`

```
FÖRTEST(xts01t1, xts02t1, xts03t1, xts10t1, xts10t2, xts10t3)
SKICKA(ISUP, REL-primitiv)
( Ej med i test: MOTTA(MC, Mediakonverterarnedkopplingsbegäran) )
MOTTA(SIP, BYE-primitiv)
MOTTA(ISUP, RLC-primitiv)
```



Figur B.24 - Testfallet xts10t4

Namn	Kategori	Resultat
xts10t5	Nedkoppling	Godkänt

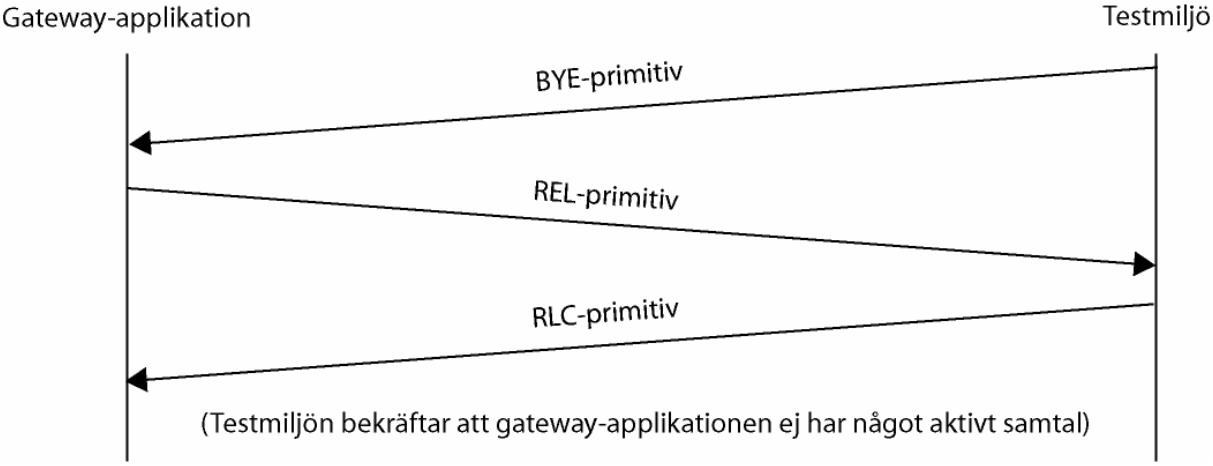
Tabell B.25 - Testfallet xts10t5

Syfte:

Simulerar hur SIP-sidan initierar nedkoppling av ett samtal.

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts03t1, xts10t1, xts10t2, xts10t3)
SKICKA(SIP, BYE-primitiv)
MOTTA(ISUP, REL-primitiv)
SKICKA(ISUP, RLC-primitiv)
( Ej med i test: MOTTA(MC, Mediakonverterarnedkopplingsbegäran) )
```



Figur B.25 - Testfallet xts10t5

Namn	Kategori	Resultat
xts10t6	Uppkoppling	Godkänt

Tabell B.26 - Testfallet xts10t6

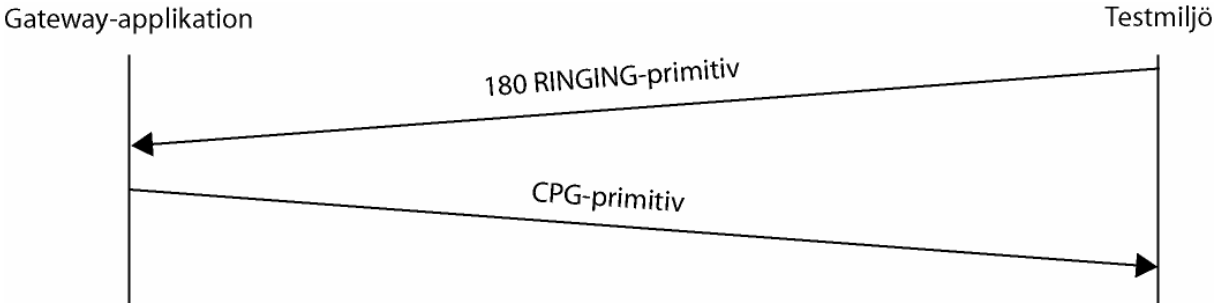
Syfte:

Testmiljön simulerar uppkoppling av ett samtal där fler än en 180 ringing når gateway-applikationen, detta är fallet om abonnenten på SIP-sidan inte lyfter luren direkt utan låter det gå några signaler.

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts03t1, xts10t1, xts10t2)
```

```
SKICKA(SIP, 180 RINGING-primitiv)
MOTTA(ISUP, CPG-primitiv)
```



Figur B.26 - Testfallet xts10t6

Namn	Kategori	Resultat
xts10_options_allow	Uppkoppling	Godkänt

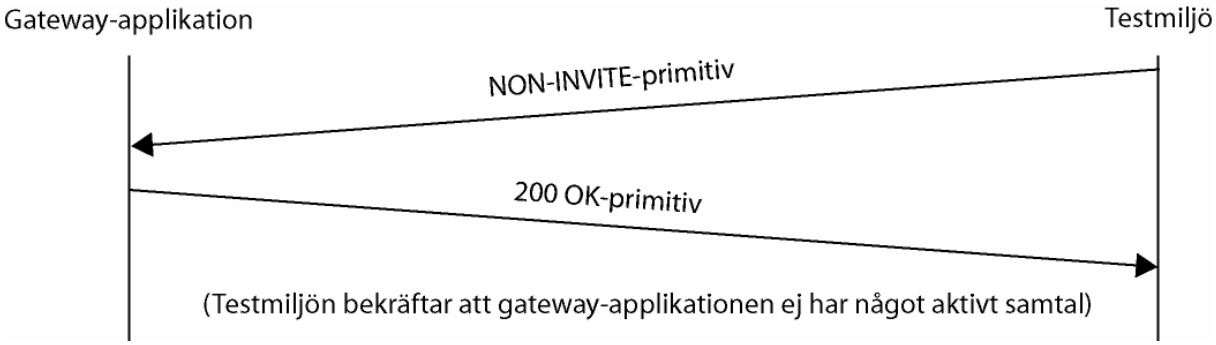
Tabell B.27 - Testfallet xts10_options_allow

Syfte:

Simulerar hur SIP-sidans telefon förhandlar med gateway-applikationen om vilka SIP-kommandon som gateway-applikationen kan hantera/tillåter (i enlighet med SIP-standarden).

Kod:

```
FÖRTEST(xts01t1, xts02t1, xts03t1, xts10t1, xts10t2, xts10t3)
SKICKA(SIP, NON-INVITE-primitiv)
MOTTA(SIP, 200 OK-primitiv)
```



Figur B.27 - Testfallet xts10_options_allow

Namn	Kategori	Resultat
mux_bind_reqt1	Interntest	Godkänt

Tabell B.28 - Testfallet mux_bind_reqt1

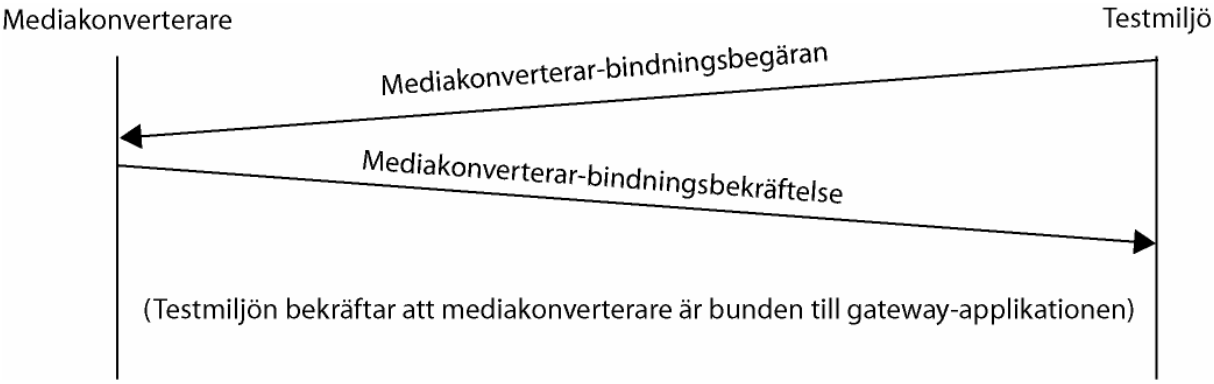
Syfte:

Testar meddelandesekvensen för ett typiskt uppkopplingsscenario där inga fel uppstår. Testmiljön försätter mediakonverteraren i ett tillstånd där inga andra moduler är bundna, skickar en bindningsbegäran och får en bindningsbekräftelse som svar.

Testmiljön bekräftar sedan att mediakonverteraren är bunden till en annan modul.

Kod:

```
SKICKA(GATEWAY, Mediakonverterar-bindningsbegäran)
MOTTA(GATEWAY, Mediakonverterar-bindningsbekräftelse)
```



Figur B.28 - Testfallet mux_bind_reqt1

Namn	Kategori	Resultat
mux_bind_reqt2	Felkontroll	Godkänt

Tabell B.29 - Testfallet mux_bind_reqt2

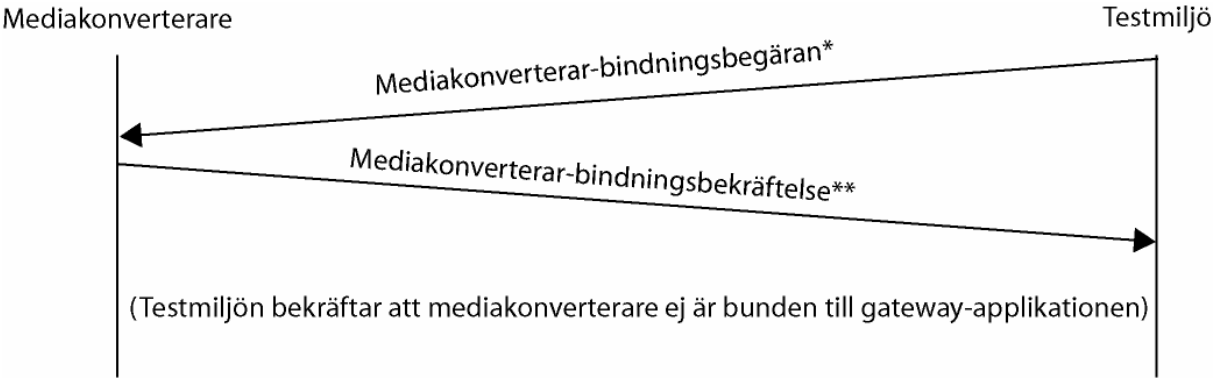
Syfte:

Testfall som demonstrerar hur mediakonverteraren reagerar på en bindningsbegäran som innehåller ett versionsnummer som mediakonverteraren inte har stöd för. Testmiljön skickar en bindningsbegäran med en felaktig version och bekräftar att mediakonverteraren rapporterar detta samt skickar tillbaka en bindningsbekräftelse innehållandes data om att en icke-stödd

version har begärts. Testfallet avslutas med att testmiljön bekräftar att mediakonverteraren inte är bunden till någon annan modul.

Kod:

```
SKICKA(GATEWAY, Mediakonverterar-bindningsbegäran)
* Här skickas en primitiv med felaktiga värden

MOTTA(GATEWAY, Mediakonverterar-bindningsbekräftelse)
** Mediakonverteraren svarar med en primitiv innehållandes en felkod
```



Figur B.29 - Testfallet mux_bind_reqt2

Namn	Kategori	Resultat
mux_bind_reqt3	Felkontroll	Godkänt

Tabell B.30 - Testfallet mux_bind_reqt3

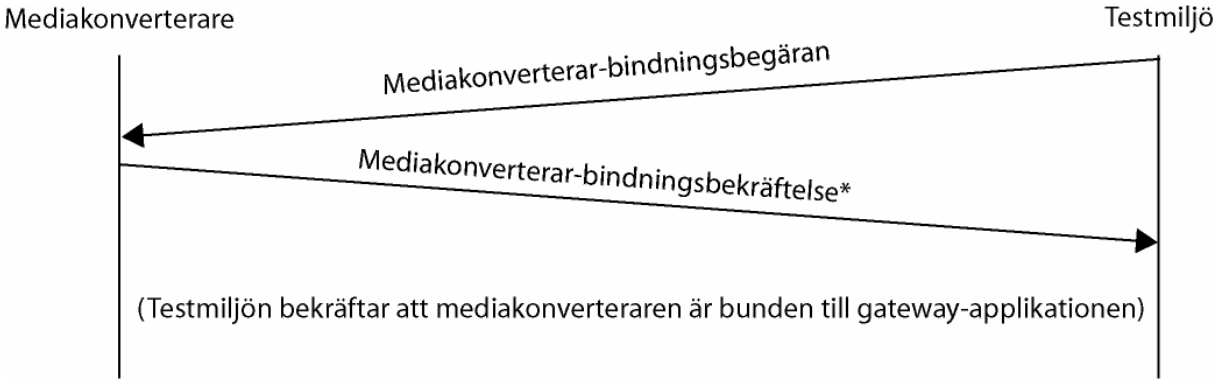
Syfte:

Testar hur mediakonverteraren reagerar när en modul redan är bunden till mediakonverteraren men sedan skickar en ny bindningsbegäran. Testmiljön bekräftar att mediakonverteraren skickar en bindningsbekräftelse innehållandes en felkod som svar på bindningsbegäran, felkoden är i det här fallet att mediakonverteraren redan är bunden till gateway-applikationen.

Kod:

```
FÖRTEST(mux_bind_reqt1)

SKICKA(GATEWAY, Mediakonverterar-bindningsbegäran)
```

```
MOTTA(GATEWAY, Mediakonverterar-bindningsbekräftelse)
(* Primitiven innehåller en felkod som indikerar att mediakonverteraren
redan är bunden till gateway-applikationen)
```



Figur B.30 - Testfallet mux_bind_reqt3

Namn	Kategori	Resultat
mux_unbind_reqt1	Interntest	Godkänt

Tabell B.31 - Testfallet mux_unbind_reqt1

Syfte:

Testar typiskt nedkopplingsscenario där gateway-applikationenen är bunden till mediakonverteraren och sedan kopplar ner. Testfallet avslutas med att testmiljön bekräftar att mediakonverteraren inte är bunden till någon annan modul.

Kod:

```
SKICKA(GATEWAY, Mediakonverterar-avbindningsbegäran)
```

Figur B.31 - Testfallet mux_unbind_req1

Namn	Kategori	Resultat
ts1t1	Interntest	Godkänt

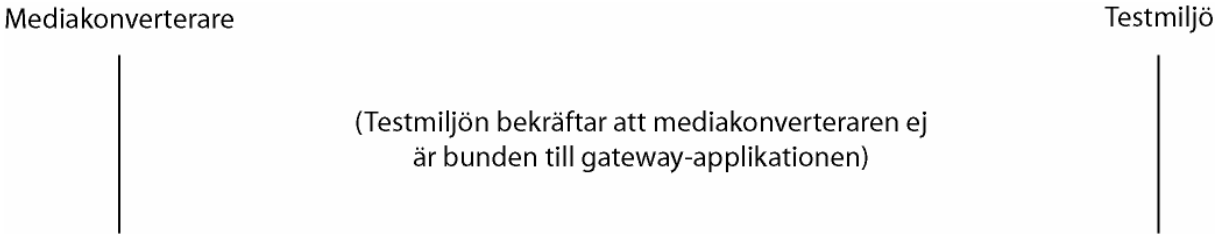
Tabell B.32 - Testfallet mux_ts1t1

Syfte:

Testmiljön försätter mediakonverteraren i det tillstånd där ingen annan modul är bunden, och bekräftar sedan att mediakonverteraren faktiskt befinner sig i det tillståndet.

Kod:

```
(Inga primitiver skickas)
```



Figur B.32 - Testfallet ts1t1

Namn	Kategori	Resultat
ts1t2	Uppkoppling	Godkänt

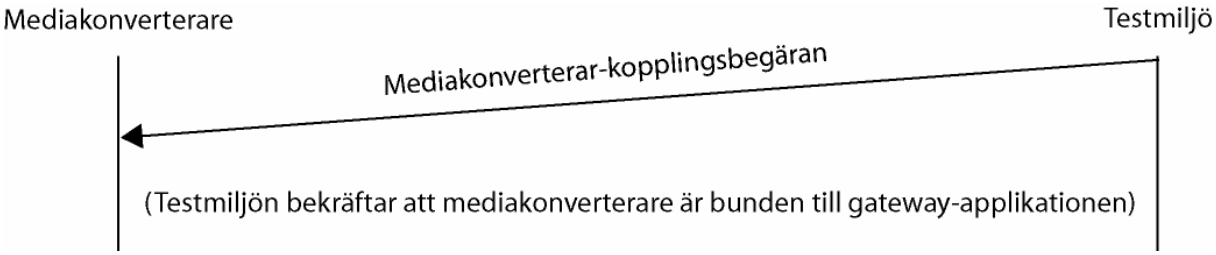
Tabell B.33 - Testfallet ts1t2

Syfte:

Testar ett typiskt uppkopplingsscenario där gateway-applikationen först binder till mediakonverteraren, sedan skickar en kopplingsbegäran. Testmiljön bekräftar att mediakonverteraren är bunden till gateway-applikationen.

Kod:

```
    FÖRTEST(mux_bind_reqt1)
    SKICKA(GATEWAY, Mediakonverterar-kopplingsbegäran)
```



Figur B.33 - Testfallet ts1t2

Namn	Kategori	Resultat
ts1t4	Nedkoppling	Godkänt

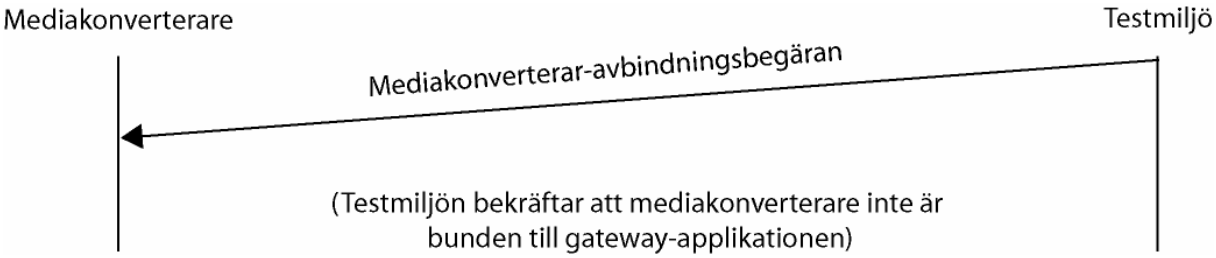
Tabell B.34 - Testfallet ts1t4

`Syfte:`

Test där testmiljön bekräftar meddelandesekvensen vid en typisk avbindning mellan gateway-applikationen och mediakonverteraren där inga fel uppstår.

`Kod:`

```
FÖRTEST(mux_bind_reqt1)
SKICKA(GATEWAY, Mediakonverterar-avbindningsbegäran)
```



Figur B.34 - Testfallet ts1t4

Namn	Kategori	Resultat
ts1t5	Interntest	Godkänt

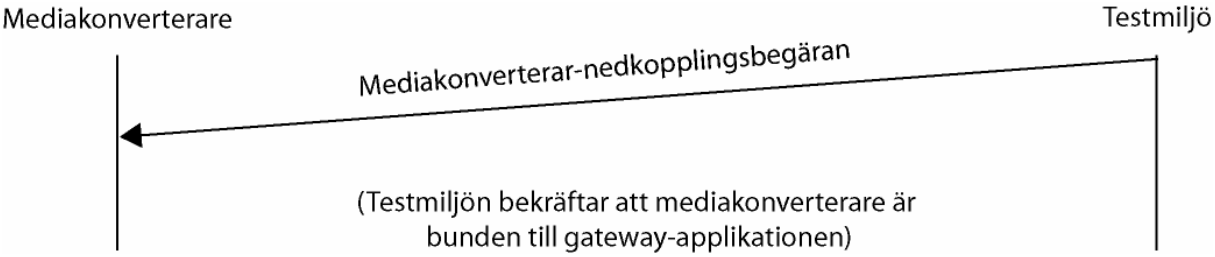
Tabell B.35 - Testfallet ts1t5

`Syfte:`

Testmiljön prövar meddelandesekvensen när mediakonverteraren mottar en nedkopplingsbegäran utan att först ha fått en kopplingsbegäran.

`Kod:`

```
FÖRTEST(mux_bind_reqt1)
SKICKA(GATEWAY, Mediakonverterar-nedkopplingsbegäran)
```



Figur B.35 - Testfallet ts1t5

Namn	Kategori	Resultat
ts1t6	Interntest	Godkänt

Tabell B.36 - Testfallet ts1t6

`Syfte:`

Testmiljön skickar en kopplingsbegäran i fel tillstånd. Ingen modul är bunden och testmiljön bekräftar att mediakonverteraren fortfarande inte har bundit till någon annan modul.

`Kod:`

```
FÖRTEST(ts1t1)
SKICKA(GATEWAY, Mediakonverterar-kopplingsbegäran)
```

Figur B.36 - Testfallet ts1t6