ELISAMA ROCHA DE CARVALHO DURÃES

CONSTRUÇÃO DE UM GATEWAY BASEADO NA PLATAFORMA CORBA PARA INSERÇÃO DAS REDES INTELIGENTES NO CONTEXTO DA NGN

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.
Orientador: Profa. Dra. Rosane Bodart Soares.

VITÓRIA
2009

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Durães, Elisama Rocha de Carvalho, 1973-

D947c Construção de um Gateway baseado na plataforma CORBA para inserção das redes inteligentes no contexto da NGN / Elisama Rocha de Carvalho Durães. – 2009.

140 f. : il.

Orientador: Rosane Bodart Soares.

Dissertação (mestrado) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Sistemas de telecomunicação. 2. CORBA. 3. Redes de próxima geração. 4. Telecomunicações. I. Soares, Rosane Bodart. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 621.3

“Ainda que eu falasse línguas, as dos homens e dos anjos, se eu não tivesse o amor, seria como sino ruidoso ou como címbalo estridente. Ainda que eu não tivesse o dom da profecia, o conhecimento de todos os mistérios e de toda a ciência; ainda que eu tivesse toda a fé, a ponto de transportar montanhas, se não tivesse o amor, eu não seria nada.”

(1 Coríntios 13: v. 1-2.)

A Marcelo, Elissa e Marceli, razão de minha vida.

A Evaldo e Marli, que me deram à vida.

A Danieli, irmã e amiga de toda minha vida.

Agradecimentos

Agradeço à Professora Doutora Rosane Bodart Soares por sua orientação, paciência, tolerância e dedicação durante todas as etapas da construção dessa dissertação.

Agradeço à minha família, em especial ao meu marido Marcelo Durães da Silva pela força, compreensão e carinho, que foram primordiais para a conclusão desse trabalho.

Agradeço aos Professores Doutores Anilton Salles Garcia e Carlos Alberto Malcher Bastos por participarem da banca examinadora.

Agradeço aos meus amigos, em especial a Talita Aparecida Silva pelo apoio na execução desse trabalho.

E agradeço a Deus, por completar essa importante etapa da minha vida.

Resumo

A inserção da Rede de Telefonia Pública nas Redes de Próxima Geração demanda a construção de conversores de protocolos instalados em gateways de sinalização. Este trabalho trata, em particular, da inclusão das Redes Inteligentes no contexto das Redes de Próxima Geração, com atenção especial para a construção do software do gateway de sinalização. Para a construção desse software foi usado o modelo de gateway do OMG e, como suporte para a provisão de QoS, acrescentou-se à arquitetura do gateway, o Serviço de Mensagens e o Serviço de Balanceamento de Carga. Uma estratégia conhecida como *Asynchronous Method Handling* é analisada e indicada como um serviço adicional para auxiliar na melhoria da escalabilidade e da vazão das mensagens trocadas no ambiente CORBA. A arquitetura do gateway foi construída sobre três ORBs – CORBA para Java (JacOrb, ORBACUS, e Orbix) com o intuito de implementar todos os recursos de ORB disponíveis e para testar a portabilidade do software.

Palavras-chave: gateway de sinalização, redes inteligentes, redes de próxima geração, escalabilidade, vazão.

Abstract

The introduction of Public Telephony Network in Next Generation networks demands the construction of protocol translators embedded in signaling gateways. This research cares about the Intelligent Network insertion in the Next Generation Network context, with special attention to construction of the signaling gateway. For the construction of this software was used the OMG gateway model, the Messaging service and Load Balancing service us the supporting to add QoS to the gateway architecture. The Asynchronous Method Handling is analyzed and suggested like another service to be included on the gateway to improve more scalability and throughput of messages changes in CORBA environment. The gateways run over three ORBs – CORBA for Java (JacOrb, ORBACUS, and Orbix) with purpose to implant all ORBs services available and to test the software.

Keywords: signaling gateway, intelligent networks, next generation network, scalability, throughput.

Lista de Figuras

Figura 1.1. Acesso da RI tradicional à camada de Serviços da NGN. 26

Figura 2.1. Camadas da Rede Telefônica Pública com RI 32

Figura 2.2. O Modelo Conceitual da RI 32

Figura 2.3. Detalhamento do Plano Funcional Distribuído 35

Figura 2.4. Arquitetura Básica da Camada de Serviços 37

Figura 2.5. Arquitetura da NGN 43

Figura 2.6. Arquitetura de Serviço para a NGN 45

Figura 2.7. Um Caminho de Evolução para as Ris 47

Figura 2.8. Arquitetura com o Gateway TINA/IN 47

Figura 2.9. Contextos de uso do GW. 48

Figura 2.10. A API Parlay 49

Figura 2.11. A Arquitetura da API JAIN para a SS7 50

Figura 2.12. Modelo da Arquitetura 51

Figura 2.13. A Arquitetura das redes *PINT/SPIRIT* 52

Figura 2.14. Interação entre PAS tradicional e PCS CORBA através do Gateway 55

Figura 2.15. A RTP atual 57

Figura 2.16. Consolidação da RTP 58

Figura 2.17. Introdução do GWS e GWT 59

Figura 2.18. Introdução do gateway de Acesso 60

Figura 2.19. Introdução de Multimídia 61

Figura 2.20. A NGN completa 62

Figura 3.1 Acesso da RI tradicional à camada de Serviços da NGN. 65

Figura 3.2. As interfaces do GWS 66

Figura 3.3. Interfaces orientadas a PDUs. 67
Figura 3.4. O processo de localização de uma aplicação e inicialização de um diálogo. 69
Figura 3.5. As interações no Gateway 70
Figura 3.6. Diálogo iniciado pelo PAS convencional 73
Figura 3.7. O Modelo SMI. 75
Figura 3.8. O Modelo Polling de CORBA para o AMI. 76
Figura 3.9. O Modelo Callback de CORBA para AMI. 76
Figura 3.10. PAS CORBA acessa PCS na RI convencional 79
Figura 3.11. O Modelo do AMH. 80
Figura 3.12. Arquiteturas para BC 83
Figura 3.13. Inclusão do Serviço BC + AMI 85
Figura 3.14. Inclusão do Serviço BC + AMI + AMH 86
Figura 4.1. A Estrutura do ORB 88
Figura 4.2 Diálogo PAS convencional e PCS CORBA 91
Figura 4.3. Execução do Servidor no JacORB 93
Figura 4.4. Execução do Cliente no JacORB – parte 1 94
Figura 4.5. Execução do Cliente no JacORB – parte 2 94
Figura 4.6. Execução do Servidor no Orbacus 4.3.0 97
Figura 4.7. Execução do Cliente no Orbacus 4.3.0 98
Figura 4.8. Execução do Servidor no Orbacus 4.3.0 99
Figura 4.9. Execução do Cliente no Orbacus 4.3.0 com AMI 100
Figura 4.10. Execução do Servidor no Orbix – Aguardando clientes 102
Figura 4.11. Execução do primeiro Cliente 102
Figura 4.12 Execução do Servidor após o primeiro Cliente 103
Figura 4.13. Execução do segundo Cliente 103
Figura 4.14. Execução do Servidor após o segundo Cliente 104
Figura 4.15. Finalizando primeiro Cliente 104

Figura 4.16. Resposta do Servidor após a finalização primeiro Cliente..... 105

Figura 4.17.Finalizando segundo Cliente..... 105

Figura 4.18 Resposta do Servidor após a finalização segundo Cliente..... 106

Figura 4.19. Tela do Simulador..... 107

Figura 4.20. Arquivo de *log* da execução do Servidor com 50 Clientes..... 112

Figura 4.21. Execução de um Cliente..... 113

Lista de Tabelas

Tabela 2.1. Etapas de padronização da RI 38

Tabela 2.2. Recomendações propostas pelo grupo ITU-T SG13 42

Lista de Siglas

ADSL	Asymmetric Digital Subscriber Line
AE	Application Entity
AEG	Application Expert Group
AGW	Access Gateway
AMH	Asynchronous Method Handling
AMI	Asynchronous Method Invocation
ApGW	Application Gateway
ASE	Application-Service-Element
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
BAS	Broadband Access Server
BC	Balanceamento de Carga
BICC	Bearer Independent Call Control
CORBA	Common Object Request Broker Architecture
CPA	Controle Por Programa Armazenado
CR	Controlador de rotas
CS	Capability Set
CS-1	Capability Set number One
DAI	Dispositivo de Acesso Integrado
DSLAM	Digital Subscriber Line Access Multiplex
EF	Entidade Funcional
ElF	Elementos Físicos
ETSI	European Telecommunications Standardization Institute
EURESCOM	European Institute for Research and Strategic Studies in Telecommunications
FACC	Função de Acesso ao Controle de Chamada
FAGS	Função de Acesso à Gerência dos Serviços
FAS	Função de Acesso ao Serviço
FCC	Função de Controle de Chamada
FCS	Função de Controle de Serviços

FDS	Função Dados de Serviço
FGS	Função Gerência de Serviços
FI	Fluxo de Informação
FRE	Função Recursos Especiais
GIOP	General Inter-ORB Protocol
GPRS	General Packet Radio Service
GT	Global Title
GW	Media Gateways
GWA	Gateway de Acesso
GWR	Gateway Residencial
GWS	Gateway de Sinalização / Aplicação
GWT	GW de Tronco
IDE	Integrated Development Environment
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IIOP	Internet Inter-Orb Protocol
IN	Intelligent Network
INAP	Intelligent Network Application Part
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
ITU	International Telecommunication Union
ITU-T	ITU - Telecommunication Standardization Sector
JAIN	Java APIs for Integrated Networks
JIDM	Open Group Joint Inter-Domain Management Task Force
LEX	Local Exchange ou Classe 5
LMDS	Local Multipoint Distribution Service
LSG	Lógica de Serviço Global
MAP	Mobile Application Part
MEGACO	Media Gateway Control
MGC	Media Gateway Controller
MGCP	Media Gateway Control Protocol
NAS	Network Access Server
NGN	Next Generation Network

OA&M	Operations, Administration, and Management
OMG	Object Management Group
ORB	Object Request Broker
PABX	Private Branch Exchange
PACS	Ponto Ambiente de Criação de Serviços
PAGS	Ponto de Acesso à Gerência de Serviços
PAS	Ponto de Acesso aos Serviços
PCB	Processamento de Chamada Básico
PCS	Ponto de Controle de Serviços
PDS	Ponto de Dados de Serviço
PDU	Protocol Data Unit
PEG	Protocol Expert Group
PGS	Ponto de Gerência de Serviços
PI	Periférico Inteligente
PINT	PSTN and Internet Interworking
PLS	Programas da Lógica de Serviços
POA	Portable Object Adapter
POI	Point Of Initiation
POR	Point Of Return
POTS	Plain Old Telephony Service – Serviço básico de telefonia
PTS	Ponto de Transferência de Sinalização
QoS	Quality of Service
RDSI	Rede Digital de Serviços Integrados
RFP	Request For Proposals
RGW	Residencial Gateway
RI	Rede Inteligente
RS	Recurso de Serviço
RTP	Rede de Telefonia Pública
SCCP	Signaling Connection Control Part
SCP	Service Control Point
SDH	Synchronous Digital Hierarchy
SGW	Gateway de Sinalização

SIB	Service Independent Building Block
SIGTRAN	Signaling Transport Group (IETF)
SIP	Session Initiation Protocol
SIP-T	SIP for telephones
SLA	Service Level Agreements
SLEE	Service of Logic Execution Environment
SMI	Synchronous Method Invocation
SPIRITS	Service in the PSTN/IN Requesting Internet Service
SS7	Sistema de Sinalização por canal comum nº 7
SSP	Service Switch Point
STP	Signaling Transfer Points
TAO	The ACE ORB
TC	Transaction Capabilities
TDM	Time Division Multiplex
TCAP	Transaction Capability Application Part
TEX	Transit Exchange ou Classe 4
TGW	Trunk Gateway
TINA	Telecommunication Information Networking Architecture
UML	Unified Method Language
VoDSL	Voice over Digital Subscriber Line

Sumário

Lista de Figuras viii

Lista de Tabelas xi

Lista de Siglas xii

Sumário xvi

1.Introdução 22

1.1 Motivação 23

1.2 A Proposta do Trabalho 25

1.3 Organização do Trabalho 27

2.Evolução das Redes de Telecomunicações: da RI Convencional à NGN 28

2.1 As Redes Inteligentes 31

2.1.1 Plano de Serviços 33

2.1.2 Plano Funcional Global 33

2.1.3 Plano Funcional Distribuído 34

2.1.4 Plano Físico 35

2.1.5 Etapas de padronização da ITU-T 38

2.2 A NGN 39

2.2.1 A Padronização da NGN 40

2.2.2 A Arquitetura NGN 43

2.2.3 A Arquitetura de Serviços da NGN 44

2.3 A Inserção das RIs na NGN 46

2.3.1 Foro TINA 46

2.3.2 Foro PARLAY / JAIN 48

2.3.3 Foro PINT / SPIRITS 51

2.4 O Gateway RI/CORBA 53

2.5 Evolução da Rede de Telefonia Fixa em direção à NGN 55

2.5.1 O ponto de partida 56

2.5.2 Otimização da infra-estrutura de rede 57

2.5.3 Instalação do Gateway de Tronco 58

2.5.4 Introdução de um GWS 59

2.5.5 Instalação do Gateway de Acesso 59
2.5.6 Introdução de Multimídia 60
2.5.7 Migração para a NGN completa 62
2.6 Considerações Finais 63

3.Construção do Software para o GWS 64

3.1 Introdução 64
3.2 A Arquitetura a ser Construída 64
3.3 As Interfaces do GWS 66
3.3.1 As Interfaces Orientadas a PDUs 67
3.3.2 As Interfaces de Interação 68
3.3.3 Exemplo de Comunicação entre um PAS Tradicional e um PCS-CORBA 70
3.4 Inclusão de Serviços Especiais de ORB 72
3.4.1 Por que inserir no modelo o serviço de mensagens? 74
3.4.2 Por que gerar Respostas Assíncronas? 77
3.4.3 Por que usar o Serviço de Balanceamento de Cargas? 81
3.5 Conclusão 85

4.Implementações e Resultados 87

4.1 Introdução 87
4.2 O Ambiente CORBA 88
4.3 O Software do GWS 90
4.4 Jacorb 92
4.5 Orbacus 96
4.6 Orbix 100
4.7 Conclusões 113

5.Conclusões 115

Referências Bibliográficas 119

Apêndice 1: Códigos Java do Servidor e Cliente 124

A1.1 Código do Servidor 124
A1.2 Codigo do Cliente 126

Apêndice 2: Código Java do Cliente modificado com o AMI para execução no Orbacus 130

A2.1 Codigo do Cliente 130

Apêndice 3: Código Java do Servidor e Cliente modificados com o BC para execução no Orbix 134

A3.1 Codigo do Servidor 134

A3.2 Codigo do Cliente 142

1. Introdução

A grande necessidade dos usuários por serviços e tráfego de dados, e o crescimento da Internet são fatores que favoreceram a disseminação do conceito da NGN. Esse conceito trata das mudanças necessárias na estrutura das redes de comunicação, e se baseia no princípio da flexibilidade. Flexibilidade esta necessária às operadoras, de forma a adaptarem seus negócios e redes ao mercado; necessária às novas operadoras, para estabelecerem negócios viáveis e lucrativos; que possibilitará prover aos usuários serviços fixos e móveis, e serviços de lazer.

Os usuários finais necessitam de serviços e aplicações diferenciados, incluindo serviços de multimídia, que não podem ser ofertados nas atuais Redes de Telefonia Públicas (RTPs). Consequentemente, as operadoras precisarão oferecer a esses usuários serviços úteis e de fácil uso, de forma a gerar mais receitas e se manterem competitivas. Assim, as NGN serão orientadas a serviços, provendo o suporte à oferta de novos serviços e a adequação dos serviços existentes para futura geração de receitas. O Protocolo de Internet (IP) será a base para essas novas redes.

Entretanto, o serviço de voz permanecerá sendo de grande importância, sendo então necessário transportar voz com qualidade sobre IP. As RTPs atuais foram concebidas para tratar tráfego de voz, e uma mudança para redes de comutação de pacotes torna-se inevitável, devido à grande necessidade de transporte de dados nas redes atuais.

A relação custo/benefício proporcionada pelas NGNs precisa ser atrativa, para possibilitar às operadoras em atuação no mercado uma migração, e que novas operadoras possam se estabelecer, já orientadas a esse novo conceito.

A convergência de dados e voz, de transporte óptico e tecnologia de pacotes, de redes fixas e móveis, devem ser facilitadas na estrutura das NGN. Convergência significa que diferentes serviços são fornecidos por um mesmo equipamento de telecomunicações, um mesmo terminal, uma mesma ligação de acesso, um mesmo suporte de transporte, um mesmo elemento de comando ou um mesmo software de aplicação [Nassif, 2004]. As NGN podem oferecer todos os tipos de aplicações a todos os tipos de usuários graças a essa convergência.

As mudanças nas regulamentações, e a crescente queda nas receitas dos serviços de voz, acompanhada de um incremento nas receitas geradas por serviços de dados de valor agregado, fizeram crescer a competição no mercado de telecomunicações. As operadoras precisarão rever seus modelos de negócios, identificando e tirando proveito das oportunidades proporcionadas pelas NGN. E as novas operadoras irão buscar modelos de negócios mais rentáveis nesse mercado novo. Como resultado, as NGN devem suportar uma grande variedade de modelos de negócios, provendo a solução mais eficiente e de melhor relação custo/benefício para cada modelo.

1.1 Motivação

O presente momento é bem adequado para a concretização das NGN. Primeiramente, os princípios das plataformas de criação de serviços e a separação da lógica do serviço foram totalmente testados em Redes Inteligentes (RIs) e estão prontos para serem estendidos nas NGN. Em segundo lugar, as tecnologias que permitem uma relação custo/benefício favorável estão agora disponíveis comercialmente, de sorte a tornar as NGN uma realidade: comutadores robustos de pacotes fabricados com tecnologias de semicondutores altamente integradas e de alto desempenho; tecnologia óptica com seu custo de largura de banda maciçamente reduzido; e novas tecnologias de acesso que provêem larguras de banda maiores aos usuários comerciais e residenciais [Nassif, 2004].

A grande preocupação das operadoras tradicionais é como migrar suas redes para uma rede IP, de maneira a minimizar os custos dessa transição e tirar maior proveito dos benefícios das NGN. Dependendo da situação específica do mercado, e do grau de saturação da rede de uma operadora, diferentes estratégias de evolução da rede podem ser aplicadas:

- *Consolidação*: otimizar a RTP instalada para reduzir todas as despesas. A consolidação pode ser combinada com uma seleção de futuros produtos de segurança para preparar a migração para as NGN.
- *Expansão*: manter a infra-estrutura e serviços da RTP existente, mas introduzir uma NGN superposta (baseada em acesso em banda larga) para atender novos clientes e introduzir novos serviços, como por exemplo, multimídia.

- *Substituição*: substituir os componentes da RTP (aqueles que estão com a vida esgotada) por componentes NGN equivalentes.

As NGN representam a proposta de evolução das atuais redes de telecomunicações centradas em voz para redes centradas em dados, caracterizadas por:

- Uma clara separação do acesso, transporte, controle e camadas de serviços;
- Interoperabilidade através de interfaces abertas entre estas camadas e todas as outras redes;
- Controle contínuo de múltiplas tecnologias de transportes (ATM, IP, TDM, *Frame Relay,* etc.);
- Uso de elementos de redes baseados em padrões (*gateway, softswitches*, servidores de aplicação, etc.) [ITU-T Y.2011, 2004].

No ambiente da RTP, as RIs se destacam pela capacidade de oferecer serviços suplementares aos de voz, e por uma separação do hardware e software nos elementos da rede. Representam a parte que deverá se interligar e compor diferentes arquiteturas de serviços dentro das condições estabelecidas para as NGN.

Porém as RIs possuem várias limitações de utilização, as quais serão abordadas no Capítulo 2. O foro denominado *Telecommunications Information Networking Architecture* (TINA) foi o primeiro que surgiu com o intuito de propor uma nova arquitetura de serviços da RI, resolvendo as limitações tecnológicas dela e evoluindo como um ambiente aberto e distribuído para novos serviços e tecnologias. Trata-se de uma RI distribuída, e deverá ser construída numa plataforma compatível com *Common Object Request Broker Architecture* (CORBA) [OMG, 1998].

Para que um cliente da RI centralizada tenha acesso aos serviços disponibilizados na RI distribuída, é necessário haver uma entidade de ligação entre a elas. Essa ligação pode ser realizada por meio de um Gateway de Sinalização/Aplicação (GWS) entre o ambiente distribuído baseado na tecnologia CORBA e a RI centralizada. Tal GWS deverá ser independente das aplicações entre as quais ele faz ponte para não ser afetado pela introdução de novos serviços. Sua funcionalidade básica é fazer a conversão de mensagens *Intelligent Network Application Part* (INAP) descritas em (ASN.1), em invocações de operações descritas em *Interface Definition Language* (IDL) executadas pelos objetos CORBA.

O *Object Management Group (OMG)* detém a especificação do GWS, devidamente documentada [OMG, 1998]. Esse documento é constituído de definições de tipos e interfaces que representam objetos com funcionalidades bem definidas no processo de interação entre a RI tradicional e os objetos CORBA. O GWS define objetos para a criação e localização de objetos CORBA, e de serviços RI por eles ofertados, além de métodos de comunicação da RI centralizada com a RI distribuída.

Dentre as pesquisas realizadas em torno da evolução das RIs, da modelagem do GWS e dos recursos importantes para sustentar as comunicações em tempo real numa rede distribuída, destaca-se aqui aquelas que contribuíram para a realização do presente trabalho:

- [Malini, 2004] e [Gobbi, 2005] implementaram alguns dos objetos do GWS, relacionados à inicialização, localização e manutenção dos diálogos entre a RI convencional e o ambiente CORBA. Eles usaram o *Object Request Broker* (ORB) JacORB para a construção desses objetos.
- [Oliveira, 2004] implementou o Serviço de Balanceamento de Cargas, com o intuito de melhorar a escalabilidade dos objetos de execução dos serviços num ambiente CORBA. Esse trabalho foi executado no ORB TAO, um ORB que utiliza C++.
- [Soares, 2003] programou os Serviços *Asynchronous Method Invocation* (AMI) e *Asynchronous Method Handling* (AMH) no TAO, de forma a permitir a troca de mensagens assíncronas entre clientes e servidores, melhorando a escalabilidade e a vazão dessa troca de mensagens num ambiente distribuído baseado no CORBA.

1.2 A Proposta do Trabalho

Tendo em vista o processo evolutivo das RIs, tendendo a fazer parte da arquitetura da NGN, a proposta deste trabalho é :

- Construir o software do GWS com os objetos necessários para interagir com os servidores de aplicações no ambiente da NGN. Esse GWS deverá ser instalado na camada de controle da NGN, permitindo a integração da RI centralizada com o ambiente distribuído baseado no CORBA;

- Integrar, se possível, todos os recursos necessários para dar suporte às comunicações do GWS com os servidores de aplicação. Esses recursos visam melhorar a vazão das mensagens trocadas no ambiente CORBA e adicionar escalabilidade aos objetos de execução dos serviços. Os referidos recursos podem ser proporcionados pelos seguintes serviços de ORB, os quais foram implementados nos trabalhos supra citados [Malini, 2004], [Gobbi, 2005], [Oliveira, 2004] e [Soares, 2003]: Serviço de Mensagens, Serviço de Balanceamento de Carga e o serviço de respostas assíncronas que usa o AMH.

Dessa forma, a arquitetura resultante seria como mostrada na Figura 1.1, onde um usuário da RI tradicional centralizada, através do Ponto de Acesso aos Serviços (PAS), pode acessar aplicações nos servidores CORBA, utilizando o GWS.

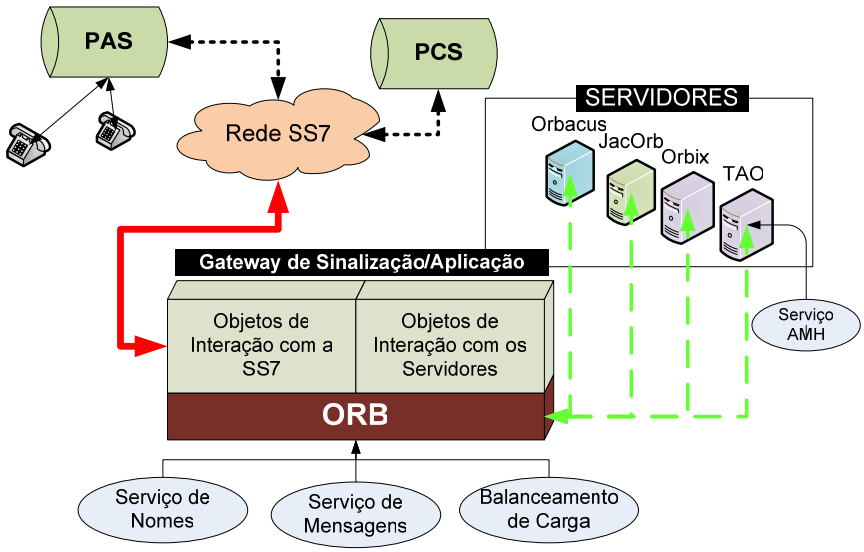

Figura 1.1. Acesso da RI tradicional à camada de Serviços da NGN.

Como existem vários provedores de ORBs no mercado, é importante também verificar a portabilidade do software do GWS e dos serviços de ORB (que em princípio são baseados em padrões) acrescentados para adicionar a escalabilidade e a vazão necessárias para obter-se uma QoS adequada aos usuários da RTP. Para esse fim os ORBs escolhidos foram: O Jacorb, Orbacus, e Orbix. Eles usam a linguagem Java, e como essa linguagem tem sido muito requisitada no mercado, considerou-se, para efeito deste trabalho de pesquisa executar os programas somente em ORBs para Java.

1.3 Organização do Trabalho

O trabalho foi organizado como descrito a seguir.

No Capítulo 2 é possível verificar todo o processo que resultou na inserção das redes inteligentes da RTP no conceito da NGN, que motivou a realização desse trabalho. Ele mostra a evolução das redes de telecomunicações, partindo das RIs até a NGN, destacando as características dessas redes e como seria possível inserir as RIs dentro da NGN, indicando um possível caminho de evolução. Apresenta também a arquitetura de um gateway entre os ambientes RI e CORBA, proposto pelo OMG.

No Capítulo 3 é mostrado como o software do GWS pode ser construído, com todos os objetos necessários para possibilitar a interação entre o Sistema de Sinalização por canal comum no. 7 (SS7) da RTP com os servidores de aplicação inseridos na RI (nova camada). São descritos também os serviços de ORB que podem ser adicionados ao GWS para fornecer certa QoS necessária no acesso aos serviços de valor agregado.

O Capítulo 4 apresenta o software que foi construído para ser inserido num GWS baseado em um ORB comercial que trabalhe com a linguagem Java. Destaca o funcionamento desse software em três ORBs distintos, o JacORB, o Orbacus e o Orbix, e verifica a capacidade de cada um deles quanto à utilização dos serviços especiais que eles podem oferecer para garantir alguma QoS. Apresenta também os testes realizados para verificar o funcionamento do GWS e dos serviços de ORB e os resultados desses testes, mostrando toda a complexidade de se trabalhar em ambientes distribuídos orientados ao CORBA.

As conclusões sobre o trabalho realizado são feitas no Capítulo 5, onde também são descritas as principais contribuições alcançadas no que concerne à introdução de uma camada distribuída na atual RI, como um passo decisivo para a migração da RTP em direção a uma NGN. São pontuadas também as limitações, obstáculos à execução do projeto e as sugestões para futuros trabalhos nessa linha de pesquisa.

2. Evolução das Redes de Telecomunicações: da RI Convencional à NGN

A telefonia fixa foi uma das primeiras formas de comunicação usadas pela sociedade em geral. Atualmente existem outras técnicas bastante difundidas, como correio eletrônico e comunicações móveis, mas a telefonia fixa ainda é muito popular.

A RTP constituída de uma hierarquia de centrais com função local, tandem e trânsito, por volta dos anos 1980 começou a expandir com a oferta de recursos de software adicionais aos serviços de voz, como por exemplo, identificador de chamadas, chamada em espera, entre outros.

Considerando que se um desses serviços fosse inserido numa central trânsito, por exemplo, o seu uso poderia provocar uma enorme sobrecarga, devido à quantidade de centrais de hierarquias inferiores e os troncos necessários para acessá-lo, eles foram programados nessas centrais de hierarquia inferiores. Essa estratégia causou outro problema, que foi a necessidade de se carregar os serviços em todos os softwares das centrais locais, para que eles pudessem estar acessíveis a todos os usuários da rede em particular. Em outras palavras, a oferta de serviços em centrais locais complicou o processo de manutenção e de introdução de novos serviços de valor agregado, além de aumentar o volume de software. Conseqüentemente, a inclusão de um novo serviço na rede era feita lenta e dificilmente.

A rede estruturada dessa forma causava também algumas implicações econômicas. Normalmente uma única companhia operava toda a rede e, conseqüentemente, monopolizava o mercado de serviços. Sem competição, não havia motivação para a inovação e nem melhoria no aprovisionamento dos serviços. Visto que a oferta de serviços era problemática, eles só eram introduzidos quando beneficiavam grandes quantidades de usuários. Portanto, muitas necessidades comerciais tornavam-se inviáveis.

O aumento no volume de software, a necessidade de flexibilizar a criação e disseminação de novos serviços na rede pública e suas conseqüências de mercado decorrentes foram decisivos para que se pudessem delinear as mudanças que precisavam ser feitas para possibilitar:

- Rapidez na oferta de serviços, direcionada às necessidades dos clientes;
- Ampliação do leque de serviços ofertados;
- Criação de um ambiente de mercado competitivo;
- Evolução suave envolvendo as redes existentes.

Tais mudanças na arquitetura da rede começaram nos anos de 1980 quando foi criada uma nova camada de software sobre a rede de transporte, separada do software das centrais com Controle por Programa Armazenado (CPA). Essa camada foi denominada de Rede Inteligente. A construção dessa arquitetura de rede mudou a forma de ofertar novos serviços. Tratava-se, em essência, de uma arquitetura que retirava a lógica de serviços (os programas) das centrais de comutação e passava para um local separado e independente.

O advento das RIs e sua consolidação através do padrão da *Telecommunication Standardisation Sector do International Telecommunication Union* (ITU-T) mudou a estratégia de introdução e disseminação de serviços na rede. Através do estabelecimento de uma plataforma aberta para a criação, implementação e gerência de serviços, tornou-se mais fácil e mais rápido ofertá-los visando atender o perfil escolhido pelo assinante, apesar de esbarrar ainda em certas dificuldades.

Uma delas foi a larga demanda por aplicações multimídia nas RTPs, que não eram possíveis dada a forma como a RI foi concebida. Isso provocou a necessidade de mais mudanças. Apesar de exigir transformações mais complexas, o processo de compatibilização da RI com a Internet foi idealizado, tendo em vista que a rede SS7, que é o veículo de mensagens para a RI, usa protocolo orientado a pacotes de dados, o qual tem compatibilidade com a Internet.

A direção tomada agora passou a ser a construção de uma arquitetura aberta e integrada com diversos provedores de rede e serviços, com objetivo de prover novas aplicações e/ou serviços, e oferecer qualidade maior e diferenciada dos serviços IP, como, por exemplo, a garantia de banda, QoS, segurança, mobilidade, banda sob demanda etc. [Filho, Sanches, 2004].

O conceito de Redes de Próxima Geração, conhecido como NGN, atende agora esses requisitos e surge como “Uma rede baseada em pacotes capaz de prover serviços de telecomunicações e capaz de fazer uso de múltiplas tecnologias de transporte em banda larga, na qual as funções relacionadas aos serviços sejam independentes das tecnologias

relacionadas à camada de transporte" [ITU-T Y.2001, 2004]. Elas possibilitam acesso flexível e mobilidade aos usuários, e provisão consistente de serviços [Heinisch, 2006].

O termo NGN é amplo nos negócios de telecomunicações, porém há um consenso comum de que toda operadora precisa de uma estratégia de próxima geração de redes para sobreviver [Filho, Sanches, 2004].

A NGN é a proposta de evolução das atuais redes de telecomunicações centradas em voz para redes centradas em dados. Mas alguns aspectos relevantes devem ser ponderados pelos diversos segmentos que estarão envolvidos (provedores de serviços, provedores de conectividade e provedores de conteúdo) tais como:

a) Necessidades a partir dos usuários finais:

- Serviços móveis com ampla cobertura (internacional);
- Alta rentabilidade das redes;
- Simplicidade na operação de serviços;
- QoS negociáveis e constante em *Service Level Agreements* (SLAs) globais;
- Eficientes parcerias para bilhetagem de serviços entre as redes.

b) Necessidades dos Provedores de Serviços:

- Criação de plataformas abertas e rápidas;
- Capacidade de informar eficientemente aos usuários sobre os serviços disponíveis;
- Garantia de QoS das rede no que se refere a: transmissão, atraso, variação de atraso, perda, prazo de reparo, segurança, bilhetagem e disponibilidade;
- Capacidade de adaptar os serviços prestados de acordo com a QoS necessária, de forma a atender o mercado no momento adequado.

c) Provedores de Redes terão necessidades de real convergência para elevação da produtividade e da eficiência e para a redução dos custos [Filho, Sanches, 2004].

Pelos itens listados se prevê que a construção da NGN se faz complexa e que alguns aspectos relevantes devem ser considerados no seu projeto.

Por tudo que foi dito até aqui se pode verificar que as redes de telecomunicações estão sempre se reestruturando, quando se trata da sua evolução em termos de serviços adicionais prestados aos usuários, e agora, acrescenta-se a necessidade de coexistir e cooperar com

outras redes para prover serviços multimídia, acessíveis o tempo todo e de quaisquer terminais.

Este capítulo foi organizado com o propósito de mostrar o passo a passo no processo evolutivo das RIs a partir de sua versão inicial até uma possível configuração dentro do conceito vigente de NGN, situar todas as discussões em torno das arquiteturas e fóruns que culminaram no modelo da NGN e descrever superficialmente o elemento principal de integração da RI convencional à NGN, que é conhecido como GWS.

2.1 As Redes Inteligentes

Em 1989, a *International Telecommunication Union (ITU)* e o *European Telecommunications Standardisation Institute (ETSI)* iniciaram os trabalhos de padronização de uma RI internacional. O processo de padronização foi dividido em fases de desenvolvimento com o intuito de definir conjuntos particulares de capacidades, conhecidos como *Capability Sets (CSs).* Cada CS continha a definição de um conjunto de serviços e/ou recursos que podiam ser construídos, de acordo com as funções disponíveis na fase de especificação em questão. Foram planejadas as seguintes áreas de especificações funcionais em uma ou mais CSs:

- Criação de Serviços;
- Gerência de Serviços;
- Interação entre Serviços;
- Processamento de Serviços;
- Gerência de Rede;
- Interoperabilidade entre Redes.

Além disso, as definições encontradas em determinada CS deveriam se compatibilizar com o conteúdo da CS anterior, objetivando, entre outros, não causar grande impacto nas redes instaladas e atingir uma arquitetura final padrão.

Os serviços oferecidos pela RI, tais como número universal, reencaminhamento de chamada em caso de linha ocupada ou congestionada, tele voto etc., foram ligados à RTP através da rede SS7.

Esse sistema de sinalização, constituído por uma pilha de protocolos de comunicações, começou a ser especificado pela ITU-T em 1980. Trata-se de um sistema em que os sinais trafegam por um caminho próprio, diferente do caminho de voz. Dentro da rede telefônica, esse sistema de sinalização é usado para fazer a comunicação entre os elementos de controle de chamada que compõem a RI, como ilustra a Figura 2.1.

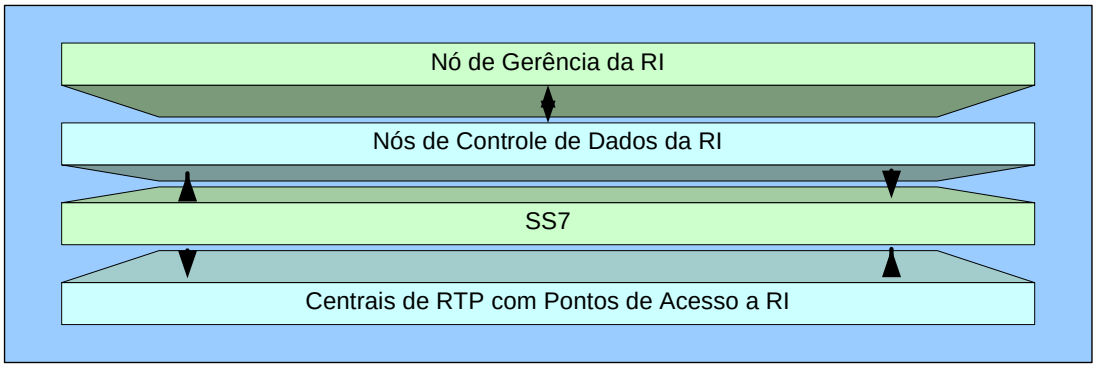

Figura 2.1. Camadas da Rede Telefônica Pública com RI

O conceito de RI padronizado pela ITU-T foi divulgado na sua série Q de recomendações [ITU-T – Recomendações série Q]. Uma RI, de acordo com a ITU-T, pode ser modelada conforme ilustra a Figura 2.2, extraída de [Soares, 2003]. Tal modelo consiste em quatro planos, onde cada um representa uma vista diferente das capacidades providas pela rede. Essas vistas descrevem, respectivamente, os aspectos de serviço, a funcionalidade global, a funcionalidade distribuída e os aspectos físicos.

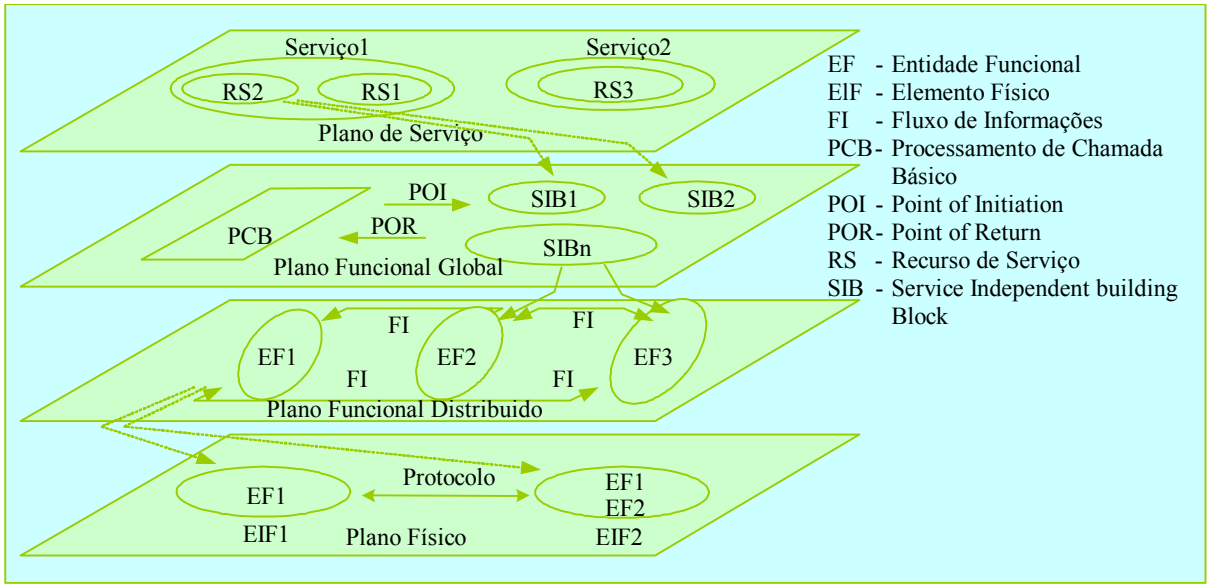

Figura 2.2. O Modelo Conceitual da RI

2.1.1 Plano de Serviços

O Plano de Serviços representa uma visão orientada exclusivamente aos serviços. Ele não contém informações acerca da implementação dos serviços na rede.

Os serviços são descritos em termos de um conjunto de blocos genéricos chamados de Recursos de Serviço (RS). Cada RS descreve um aspecto particular de um serviço, podendo ser reutilizado na composição de outros serviços.

2.1.2 Plano Funcional Global

No Plano Funcional Global os serviços e os RSs são *redefinidos* em termos de funções de rede necessárias para executá-los. Essas funções de rede são os Blocos de Construção Independente dos Serviços, referenciadas como SIBs (*Service Independent Building Blocks*). Esses SIBs são construídos para compor diferentes serviços ou RSs.

O Plano Funcional Global contém também a Lógica de Serviço Global (LSG) e o SIB Processamento de Chamada Básico (PCB). O PCB é responsável pela provisão da conectividade básica de chamada entre as partes da rede. A LSG descreve como os SIBs são encadeados para descrever os RSs ou os serviços propriamente ditos. Ela também descreve a interação entre o PCB e as cadeias de SIBs.

Para criar um serviço RI é preciso primeiro construir cadeias de SIBs que satisfaçam as exigências do serviço. Uma cadeia de SIBs começa no POI (*Point Of Initiation*) e termina no POR (*Point Of Return*).

O PCB é um SIB especializado que representa a interação entre a Função de Acesso ao Serviço (FAS) e a Função de Controle de Serviços (FCS), definidos no Plano Funcional Distribuído, e que pode conectar/desconectar chamadas e também reter dados de chamada. De acordo com o POI recebido, uma cadeia de SIBs é disparada no PCB, a qual termina no POR correspondente.

Os SIBs podem ser reutilizados em combinação com outros SIBs, para compor novos serviços. Isso facilita e torna mais rápido o processo de desenvolvimento de um serviço.

Nas recomendações contidas no *Capability Set number One* (CS-1) foram definidos 14 SIBs, especificamente para os serviços de telefonia. Eles podem ser encontrados em [Q1213, 1995].

2.1.3 Plano Funcional Distribuído

O Plano Funcional Distribuído é de interesse dos provedores de rede. Ele define a arquitetura funcional de uma rede estruturada com a RI em termos de funcionalidades de rede, conhecidas como Entidades Funcionais (EF). Cada EF representa um agrupamento único de funções em determinada localidade. Uma mesma entidade física pode conter uma ou mais EF, mas uma EF só pode ser mapeada inteiramente dentro de uma única entidade física.

As EFs especificadas pela ITU [Q1214, 1993] são mostradas na Figura 2.3, extraída de [Soares, 2003].

- Função de Acesso ao Controle de Chamada (FACC): é a interface entre o usuário e as funções de controle de chamada, com o objetivo de estabelecer, manter, modificar e liberar uma chamada ou uma instância de serviço;
- Função de Controle de Chamada (FCC): é responsável, entre outros, pelo processamento e controle das chamadas e conexões, como requisitado pela FACC. Ela também provê os gatilhos para acessar as funcionalidades da RI (por exemplo, para passar os eventos para a FAS);
- Função de Acesso ao Serviço (FAS): associada a FCC, fornece o conjunto de funções necessárias para a interação entre a FCC e a FCS. É responsável pelo reconhecimento dos gatilhos de controle de serviços e pela interação com a FCS. A FAS modifica as funções de processamento de chamada para processar as requisições de utilização dos serviços providos pela RI sob controle da FCS;
- Função de Controle de Serviços (FCS): comanda as funções de controle de chamada no processamento das requisições de serviços providas pela RI. A FCS contém a lógica e as capacidades de processamento necessárias para manipular as tentativas de serviço providas pela RI. A FCS pode interagir com outras EFs para acessar lógica adicional ou para obter informação (serviço ou dados do usuário) necessária ao processamento da instância da lógica de serviço. Ela interage com a FAS/FCC, Função Recursos Especiais

(FRE) e Função Dados de Serviço (FDS). Se necessário, pode interagir com outras FCSs;

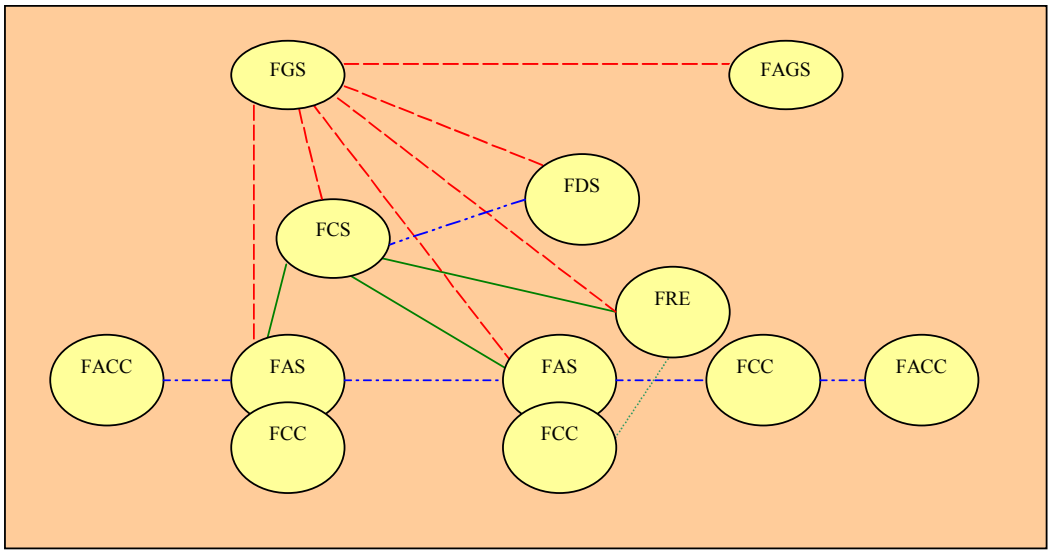

Figura 2.3. Detalhamento do Plano Funcional Distribuído

- Função Dados de Serviço (FDS): contém os dados da rede e do cliente que são acessados em tempo real pela FCS na execução de um serviço RI. Ela interage com as FCSs e com outras FDSs (se necessário);
- Função Recursos Especiais (FRE): provê os recursos especializados para a execução dos serviços RI (anúncios, por exemplo). Ela interage com a FAS (e com a FCC) e a FCS;
- Função de Acesso à Gerência dos Serviços (FAGS): é uma interface entre os gerentes de serviços e a FGS. Ela permite que o usuário (gerente) gerencie seus serviços, através de acesso a FGS;
- Função Gerência de Serviços (FGS): faz a distribuição e aprovisionamento de serviços RI e dá suporte às operações em andamento. Em particular, ela coordena as diferentes instâncias da FCS e FDS para serviços do tipo tarifas, estatística, notificação de dados de serviços, etc. A FGS gerencia, atualiza e administra as informações de serviço na FRE, FAS e FCS.

2.1.4 Plano Físico

O Plano Físico do modelo conceitual da RI descreve os Elementos Físicos (ElF) que implementam as EFs, a saber [Q1205, 1993]:

- PAS (Ponto de Acesso aos Serviços): é uma central de comutação (local ou trânsito) com capacidade para detectar requisições de serviços RI, e que permite o acesso dos usuários à rede. Ele também é capaz de se comunicar com outros ElF e responder às suas instruções. Ele contém uma FCC e uma FAS e, se for uma central local, contém também uma FACC;
- PCS (Ponto de Controle dos Serviços): contém os Programas da Lógica de Serviços (PLSs) usados para prover os serviços RI (FCS) e pode conter, opcionalmente, dados dos clientes (FDS). Ele pode estar conectado aos PASs e, opcionalmente, aos Periféricos Inteligentes (PIs). Podem se comunicar com outros PCSs;
- PDS (Ponto de Dados de Serviço): contém os dados usados pelos PLSs para prover os serviços. Ele contém a FDS. Pode ser acessado diretamente por um PCS e/ou por um Ponto de Gerência de Serviços (PGS). Pode acessar outros PDSs;
- PI (Periférico Inteligente): provê a funcionalidade FRE, possibilitando a personalização dos serviços e as interações com flexibilidade entre os usuários e a rede. Alguns exemplos de recursos especializados são: anúncios de voz personalizados, dispositivos de reconhecimento de voz, entre outros. Ele se conecta a um ou mais PASs;
- PGS (Ponto de Gerência de Serviços): executa a gerência, provisão e distribuição dos serviços. São de sua responsabilidade a administração de bases de dados, gerência de tráfego e coleta de dados da rede, entre outros. O PGS contém a FGS e pode conter a FAGS e a FACS. Ele pode acessar todas as outras entidades físicas;
- PACS (Ponto Ambiente de Criação de Serviços): é usado para definir, desenvolver e testar um serviço RI e colocá-lo no PGS. Ele contém a FACS, e interage com o PGS;
- PAGS (Ponto de Acesso à Gerência de Serviços): fornece aos gerentes e clientes dos serviços o acesso ao PGS.

Além disso, a RI conta com Pontos de Transferência de Sinalização (PTSs) e enlaces de sinalização. Os PTSs são comutadores de pacotes de altíssima capacidade que transportam as mensagens de sinalização entre os nós da rede. Um PTS pode processar cerca de 5000 mensagens de sinalização por segundo e introduz atrasos nos pacotes inferiores a 100ms [Soares,2003]. O tempo de paralisação ou inatividade de um ponto de sinalização a outro deve ser inferior a 10 minutos por ano.

A pilha de protocolos SS7 é usada dentro da RTP para fazer a comunicação entre três pontos de sinalização ou nós da RI: os PASs, os PTSs e os PCSs. Especificamente, o protocolo INAP [Q1218, 1998] define um conjunto de operações e regras para o diálogo entre os elementos da RI, tais como entre os PASs e o PCS.

As mensagens SS7 têm origem em um PAS, que estabelece ou recebe uma chamada. Para estabelecer a chamada o PAS envia uma mensagem ao PTS diretamente conectado a ele. Se o PAS de origem sabe qual rota tomar para a chamada, ele solicitará ao PTS que crie uma conexão ao PAS de destino. Se a rota for desconhecida, como ocorre com os serviços RI, o PTS terá que contatar um PCS que tenha acesso aos dados de rastreamento de chamada. O PTS usa essas bases de dados para executar uma função denominada *tradução de título global*, que determina um PAS de destino, convertendo o número discado pelo chamador em uma rota ao PAS de destino. Os PASs também podem enviar informação de tarifas aos PCSs usando a SS7. A arquitetura básica da camada de serviços da RI é mostrada na Figura 2.4.

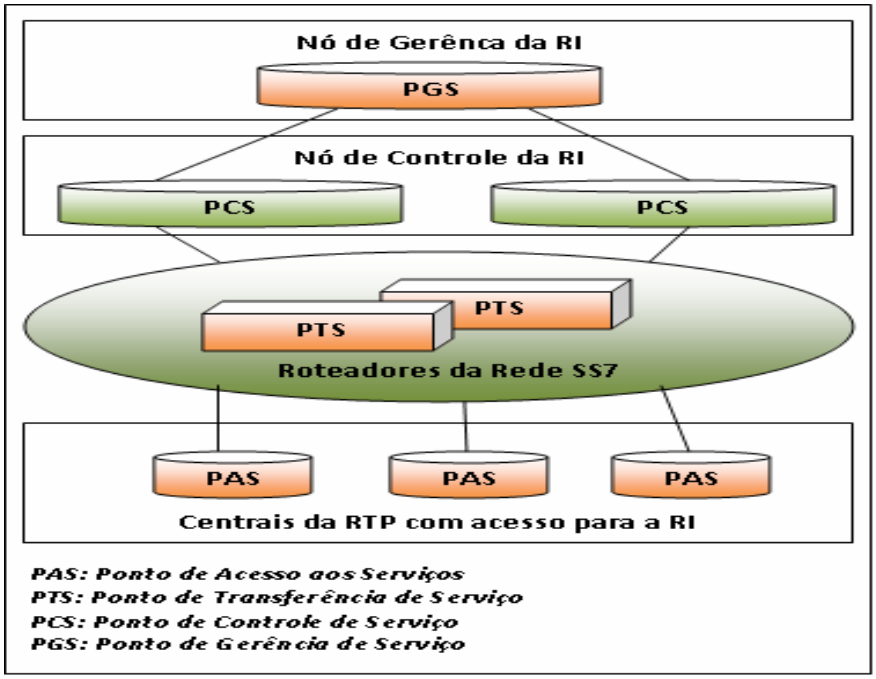

Figura 2.4. Arquitetura Básica da Camada de Serviços

O conceito da RI possibilitou o surgimento e disseminação de novos tipos de serviços (como multimídia, por exemplo), porém a experiência de muitos anos de utilização em vários países demonstrou algumas limitações e desvantagens em sua utilização [TINA-C]:

- Foi criado um PCS centralizado, responsável pela disponibilização dos recursos de software referente ao serviço adicional contratado, que não tinha flexibilidade para expansão (baixa escalabilidade);

- A RI foi implantada sobre uma rede orientada à conexão ponto-a-ponto;
- Exigia a vinculação da criação de um novo serviço ao conhecimento detalhado da rede;
- Isolamento da RI em relação a outras redes;
- Dificuldade de personalização de serviços existentes e criação de serviços específicos para um cliente;
- Dificuldade de atualização dos PASs, para cada nova versão da RI.

A tendência é a construção de uma arquitetura aberta e integrada com diversos provedores de rede e serviços. O conceito da NGN surge como uma proposta de evolução das atuais redes de telecomunicações centradas em voz para redes centradas em dados, de forma a superar as limitações atuais.

2.1.5 Etapas de padronização da ITU-T

Os padrões para as RIs foram iniciados por várias fontes (ITU, ETSI, Bellcore). A tabela 2.1 ilustra as etapas de padronização da RI na ITU. A primeira delas, a CS-1, começou a ser planejada em 1989, mas só foi aprovada em fevereiro e 1995. A CS-2 foi produzida em 1997. O modelo conceitual da RI ilustrado na Figura 2.2 foi extraído das recomendações contidas nas CS-1 e CS-2.

Tabela 2.1. Etapas de padronização da RI

Capability Set	Conjunto de Recomendações
CS-1	Q.1211, Q.1213, Q.1214, Q.1215 e Q.1218
CS-2	Q.1221, Q.1222, Q.1223, Q.1224 e Q.1225
CS-3	Q.1231, Q.1236, Q.1237 e Q.1238
CS-4	Q.1240, Q.1241, Q.1244 e Q.1248

A CS-3 extende as capacidades da RI da CS-2 no que diz respeito aos serviços, às redes e ao gerenciamento dessas redes. Destaca-se:

a) A possibilidade de múltiplos PLSs atuarem na mesma chamada;
b) Interação da RI com a rede digital de serviços integrados (RDSI);
c) Portabilidade numérica;
d) Suporte para mobilidade;

e) Suporte para RDSI banda larga;
f) Alguns aspectos de interação entre a rede IP e a RI

Com os avanços no modelamento de chamadas, na segurança, nas características de interação e no gerenciamento de mobilidade, a CS-4 oferece, além das capacidades da CS-3:

a) A possibilidade de mais de um PLS em um ou mais PCSs atuarem na mesma chamada.
b) Suporte para voz sobre IP;
c) Mais aspectos de interação entre a rede IP e a RI.

Embora seja lento o processo de adição de capacidades no padrão da RI, importantes recursos foram incluídos nas últimas recomendações. Muitos deles resultaram da contribuição de vários foros de pesquisa, como por exemplo, o foro TINA, que será apresentado adiante.

2.2 A NGN

A arquitetura proprietária dos atuais equipamentos de rede faz com que as operadoras de redes de telecomunicações e os provedores de serviços fiquem inteiramente dependentes dos fabricantes de equipamentos para as atualizações de software/hardware e para novas aplicações. Essas atualizações e a oferta de novos serviços levam um tempo relativamente grande de desenvolvimento e têm alto custo.

Essa dependência de fornecedores tem sido muito prejudicial à expansão da rede e dos serviços. O esforço que tem sido feito atualmente é no sentido de criar uma nova arquitetura de rede cuja próxima geração seja a separação do hardware e do software nos equipamentos, através de interfaces padronizadas, criando um ambiente multifornecedor para hardware e outro para software.

A idéia é construir a arquitetura de uma rede de próxima geração, a NGN, que são redes de comunicação convergentes, com interfaces abertas e padronizadas, e capacidade para transmitir voz, dados, imagens, som e vídeo ao mesmo tempo, e por isso, são chamadas também de redes multisserviços. A principal diferença entre as NGN e as redes tradicionais, de comutação por circuitos, está na estrutura de transmissão por pacotes utilizada no protocolo IP e adotada nessas novas redes. Os terminais mandam pacotes de dados, em

formato IP, para um ponto concentrador, de onde passam a circular pela rede até encontrar o destino, que é um endereço IP. Basicamente, é o mesmo procedimento hoje em uso na Internet. A principal mudança a ser feita na estrutura das operadoras, para oferecer serviços NGN, refere-se à transmissão de voz. Para que possam trafegar nas novas redes, os sinais de voz precisam ser transformados em pacotes, que se misturam aos pacotes de dados (e imagens) durante o transporte. Essa função é realizada por gateways de voz, que são instalados na camada de acesso da rede, onde também estão os roteadores e toda a infra-estrutura física da operadora [Nassif, 2004].

A migração dos serviços de voz para a infra-estrutura NGN, que utiliza pacotes para a transmissão das informações, precisa garantir a QoS relacionada a esses serviços, no que diz respeito à transmissão, atraso, variação de atraso e perda dos pacotes, bem como prazo de reparo, segurança, bilhetagem e disponibilidade dos serviços.

Uma das grandes vantagens das NGN está no uso de interfaces, *Application Programming Interfaces* (APIs), e protocolos abertos e padronizados para suportar a criação, aprovisionamento e gerenciamento de novos serviços independentemente da rede e tipo de acesso utilizado. É importante haver uma clara separação entre as funções para os serviços e as funções para o transporte nessa rede.

2.2.1 A Padronização da NGN

Em 1995, a ITU-T lançou o *Global Information Infrastructure Project* (GII Project), com o objetivo de preparar normas para a era emergente de informação. Ele basicamente descreve “a provisão de vários serviços por uma variedade de provedores, sobre uma variedade de tecnologias de rede de diferentes setores da indústria” [Heinisch, 2006].

No mesmo período em que o GII foi definido, novas tecnologias vinham amadurecendo e ficando mais avançadas, e as redes existentes precisaram evoluir em face à nova realidade das telecomunicações.

Foi então que a ITU iniciou o *NGN 2004 Project*, datado de Fevereiro de 2004, com o objetivo de coordenar atividades relacionadas à NGN, estabelecer guias de implementação e normas para a realização da NGN. A principal preocupação desse trabalho foi garantir a interoperabilidade e capacitação da rede para suportar aplicações de forma global sobre a NGN.

O Grupo de Estudo SG-13 da ITU-T iniciou este projeto específico para NGN, alinhado com o GII. A NGN é vista como a realização concreta dos conceitos do GII. A ITU-T através do SG-13 deu início ao *NGN 2004* e com base nas produções do GII propôs as normas mostradas na Tabela 2.2.

A recomendação Y.2001 12/2004 define a NGN como "Uma rede baseada em pacotes capaz de prover serviços de telecomunicações e capaz de fazer uso de múltiplas tecnologias de transporte com QoS em banda larga, na qual as funções relacionadas a serviço sejam independentes das tecnologias relacionadas ao transporte".

Em meados de 2004 foi levantada por parte do ITU-T a urgência em trabalhar em normas globais para NGN e levar alguma clareza ao esforço de padronização da NGN. Operadoras em todo mundo estão implementando estratégias NGN e planos de investir bilhões de dólares em novas redes baseadas em IP. Operadoras, fabricantes de sistemas e membros do ITU-T, acreditam que normas internacionais facilitarão um mercado global para sistemas de mais baixo custo, proporcionando interoperabilidade global, com implementações mistas e casadas.

Tabela 2.2. Recomendações propostas pelo grupo ITU-T SG-13

Y.NGN-Overview	Visão geral da NGN
Y.GRM-NGN	Princípios gerais e modelo de referência geral para Redes de Próxima Geração
Y.NGN-FRM	Modelo de arquitetura funcional
Y.NGN-SRQ	Requisitos de serviços da NGN
Y.NGN-MOB	Requisitos de gerenciamento de mobilidade e arquitetura da NGN
Y.NGN-MAN	Estrutura para gerenciamento da rede IP
Y.NGN-MIG	Migração de redes (incluindo redes TDM) para NGN
Y.NGN-CONV	Consideração regulatória para a NGN
Y.e2eqos	Arquitetura de QoS fim-a-fim para redes IP dentro da NGN
Y.123.qos	Uma arquitetura de QoS para redes de acesso Ethernet baseadas em IP
Y.NGN-TERM	Terminologias de Redes de Próxima Geração: Termos e definições

Em sete de maio de 2004 a ITU-T estabeleceu um novo grupo, o FGNGN - *Focus Group on NGN*, para cuidar das necessidades urgentes de produção de normas globais para NGN. Em junho de 2004, todos os *Drafts* das normas relacionados à NGN, iniciados pelo SG13, foram transferidos para o FGNGN - *Focus Group on NGN*. O *NGN 2004 Project* foi congelado.

O trabalho deste grupo foi inicialmente construir a arquitetura de convergência das redes fixa e móvel existentes, para prover transparência entre essas redes. O progresso do trabalho se daria nas áreas de QoS, autenticação, segurança e sinalização.

Os documentos resultantes do FGNGN cobrem principalmente sete áreas de trabalho da Padronização da NGN I:

- Serviços e Capacidades;
- Arquitetura funcional e requirements;
- Qualidade de Serviço (QoS);
- Aspectos de controle;
- Segurança;
- Migração das redes atuais para NGN;
- Requerimentos para futuras redes baseadas em pacotes.

Seguindo o FGNGN, os trabalhos futuros em NGN progredirão sob a responsabilidade do NGN-GSI - *NGN Global Standards Initiative*, envolvendo todos os grupos de estudo e outras organizações trabalhando em NGN dentro da ITU-T. As metas e objetivos desse grupo são:

- Endereçar o mercado que precisa das normas NGN;
- Produzir normas globais para NGN;
- Manter visibilidade dos trabalhos espalhados entre diferentes grupos técnicos na ITU-T;
- Disseminar mais o papel principal da ITU-T entre os grupos, no trabalho de padronização da NGN.

Pode-se notar que o conceito geral de NGN já existia mesmo antes do termo NGN ser criado pelo ITU-T, isto graças ao *GII Project*.

As normas propostas pela ITU-T ainda estão em elaboração, aprovação ou em fase de publicação. A ITU-T ainda tem um longo trabalho de padronização pela frente. E para isso, tem sido necessária uma maior consideração no planejamento futuro, através de um plano de trabalho bem coordenado, para focar o trabalho da NGN e evitar trabalhos adicionais nos itens já fechados.

2.2.2 A Arquitetura NGN

A arquitetura NGN está decomposta em camadas funcionais, como ilustra a Figura 2.5.

A camada de acesso provê interfaces para que os diversos tipos de usuários (telefones, computadores pessoais, modems, PABX, etc) acessem as aplicações e serviços na NGN. Na camada de núcleo é realizado o encaminhamento e o transporte das informações do usuário até o destino (comutadores, multiplexadores). A camada de controle é responsável pelo estabelecimento das rotas nos equipamentos de acesso e de núcleo para o encaminhamento do tráfego, e pela sinalização. A camada de serviços e aplicações trata das aplicações dos usuários, autenticações, autorizações, a tarifação e a gerência [Tronco, 2006].

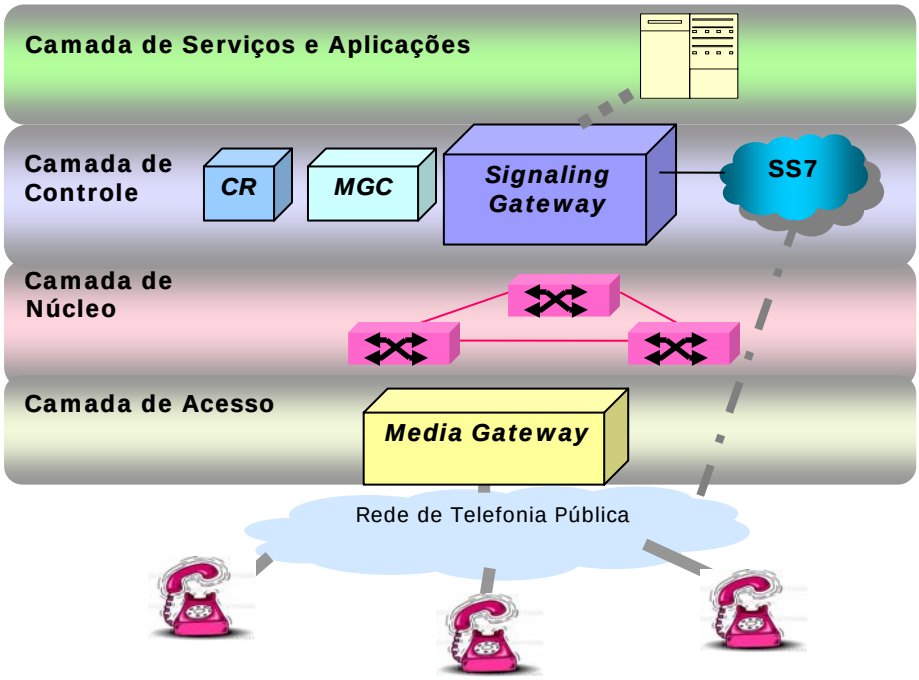

Figura 2.5. Arquitetura da NGN

A estrutura da NGN se baseia em alguns elementos chave, mostrados na Figura 2.5:

- *Media Gateways (GW)*: responsáveis pela conversão do tráfego de voz e dados de sua forma original, na forma IP. Pode ser utilizado para conectar os equipamentos dos usuários à NGN - *Media Gateway* de Acesso, ou para conectar a NGN à RTP - *Media Gateway* de Tronco.

- *Media Gateway Controller (MGC), Servidor de Chamadas ou Softswitch*: Fornece serviços de tratamento das chamadas dos assinantes telefônicos para o encaminhamento da informação da rede de telefonia sobre a rede IP. Ele também implementa as facilidades e serviços suplementares ofertados. Para isso, o *Softswitch* deve ser capaz de entender a lógica de cada serviço, e traduzi-la em comandos adequados a cada elemento da rede. Portanto, o *Softswitch* é o elemento responsável pela inteligência da NGN, controlando os media gateways e todo o processamento das chamadas dentro da rede de pacotes.
- *Controlador de rotas (CR)*: efetua o cálculo das rotas para encaminhamento das chamadas dentro da rede IP.
- *Gateway de Sinalização (SGW)*: converte a sinalização SS7 da RTP para a sinalização IP.
- *Rede de Pacotes (Núcleo)*: comporta os equipamentos de encaminhamento e transporte IP (comutadores e multiplexadores).

A separação na arquitetura, em camadas distintas, entre a manipulação da mídia (pelos *gateways*) e a manipulação da sinalização e controle envolvidos na chamada (pelo *Softswitch)* tem o ensejo de prover a máxima flexibilidade para a evolução da rede.

Os GW proverão a interoperabilidade entre redes NGN de operadoras diferentes e a de uma rede NGN com as redes convencionais. Os terminais que se conectarão a ela incluem terminais telefônicos analógicos, máquinas de fax, terminais *ISDN*, celulares, dispositivos terminais *GPRS* (*General Packet Radio Service*), terminais *SIP* (*Session Initiation Protocol*), Telefones Ethernet através de *PCs* (*Personal Computers*), *Set-top boxes* digitais (TV digital), *cable modems* etc.

Uma grande facilidade da NGN será a mobilidade generalizada, que deverá prover a habilidade de um usuário e/ou equipamento terminal utilizar diferentes tecnologias de acesso em diferentes localidades enquanto pode se movimentar, permitindo a ele usar e gerenciar consistentemente suas aplicações/serviços dentro dos limites da rede existente.

2.2.3 A Arquitetura de Serviços da NGN

A NGN visa à oferta de novos serviços com mais agilidade e flexibilidade, diminuindo os custos de operação da infra-estrutura das redes existentes.

A Figura 2.6, extraída de [Heinisch, 2006], detalha as diferentes nuvens (RTP, wireless, IP) se interligando à camada de aplicação da NGN através das interfaces abertas. O ambiente de criação dos serviços pode ser um provedor independente onde o acesso é autenticado, controlado e filtrado; ou o próprio provedor da rede, organização subordinada ou associada. Os provedores de serviço podem estar localizados em qualquer lugar na rede, o processamento é distribuído, com software orientado a objeto para localizar automaticamente objetos de serviço alvo, enviar mensagens e receber eventos.

Figura 2.6. Arquitetura de Serviço para a NGN

A plataforma do servidor de aplicação é composta por um grupo de servidores onde cada categoria deles corresponde a um serviço básico sob controle de terceiros. Alguns serviços podem ser realizados em redes diferentes, as ações dos servidores são somente relacionadas a serviços e não a camadas mais baixas específicas da rede. É consenso que na arquitetura de serviços, a NGN deve usar tecnologia de comutação por pacote (especialmente redes baseadas em *IP*), permitindo a convergência da camada de rede às superiores [Heinisch, 2006].

A arquitetura de serviços da NGN deve apresentar controle distribuído, adaptar-se ao processamento distribuído natural das redes *IP*, eliminar a estrutura centralizada (nos PCSs) da SS7 e suportar transparência à localização. O controle deve ser aberto, suportar criação e atualização de serviços por terceiros. A provisão de serviço separada da rede deve estimular um ambiente competitivo na oferta de uma diversidade de serviços. A segurança e a proteção serão embutidas na arquitetura, que são requisitos básicos de uma arquitetura aberta.

Como se pode ver, a NGN deverá prover o que for preciso (desde infra-estrutura a protocolos) para criar, desenvolver e gerenciar todos os tipos de serviços possíveis, usando diferentes tipos de mídia (áudio, visual e audiovisual), com todos os tipos de codificação e serviços de dados sensíveis a atrasos (conversacional, *unicast*, *multicast* e *broadcast*, mensagem, dados simples, serviços de transferência, *real-time* e *non-real-time*), e tolerantes a eles. Serviços com diferentes demandas de largura de banda devem ser suportados dentro das capacidades das tecnologias de transporte. Dentro da NGN há uma ênfase crescente na customização de serviços pelos provedores.

2.3 A Inserção das RIs na NGN

Tomando por base o que foi exposto sobre a NGN, fundamentadas na criação de *APIs*, *gateways* e de metodologias de integração das redes, para inserir os serviços de telecomunicações nesse contexto, foi preciso considerar três grandes mudanças estruturais na adequação da arquitetura da RI mostrada na Figura 2.4 à NGN.

1. Modelar o *GW* de interligação, para que ele possa ser construído fisicamente sobre qualquer plataforma distribuída (neste trabalho se pressupõem que a plataforma será um ORB, de acordo com as especificações contidas na documentação do OMG) [OMG, 1998];
2. Construir as *APIs* de controle de serviços para serem montadas sobre a RTP;
3. Desenvolver metodologias de convergência da RIs com as redes IP.

Os principais foros que se voltaram para a execução dessas tarefas foram o *TINA, o PARLAY, o JAIN, o PINT e o SPIRITS.*

2.3.1 Foro TINA

O TINA foi criado no início dos anos 1990 e sua meta inicial era solucionar as limitações das RIs, introduzindo, gradualmente, sistemas distribuídos nos elementos de execução dos serviços, suportados por uma plataforma compatível com CORBA. Esse processo está ilustrado na Figura 2.7.

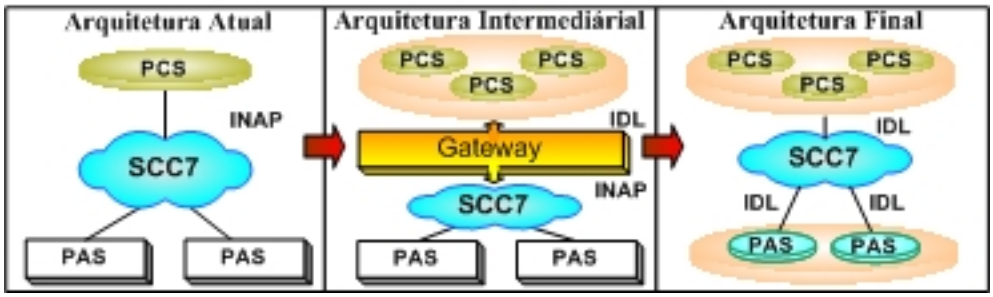

Figura 2.7. Um Caminho de Evolução para as Ris

Inicialmente a introdução de objetos distribuídos ocorreria na camada de serviços, em substituição aos atuais Pontos de Controle de Serviços (PCSs), preservando a estrutura de acesso (PASs). Assim, os PCSs passariam a ser distribuídos em servidores CORBA e acessados pelos PASs da RI convencional através de um *GW*.

Na segunda e última etapa ocorreria a substituição de todos os elementos da RI por objetos de serviço CORBA, mantendo a rede SS7 como rede de transporte entre os diferentes nós CORBA.

Na sua fase final, a arquitetura deveria também prover os recursos necessários para que seus usuários pudessem acessar e usar os serviços da RI convencional. Chegou-se à especificação de uma arquitetura de rede como ilustrada na Figura 2.8 [OMG, 1998].

Um *GW* foi colocado primeiramente entre o domínio dos servidores CORBA (PCSs distribuídos) e o domínio SS7 para realizar a conversão de protocolos e adaptar as formas de execução dos domínios CORBA e SS7.

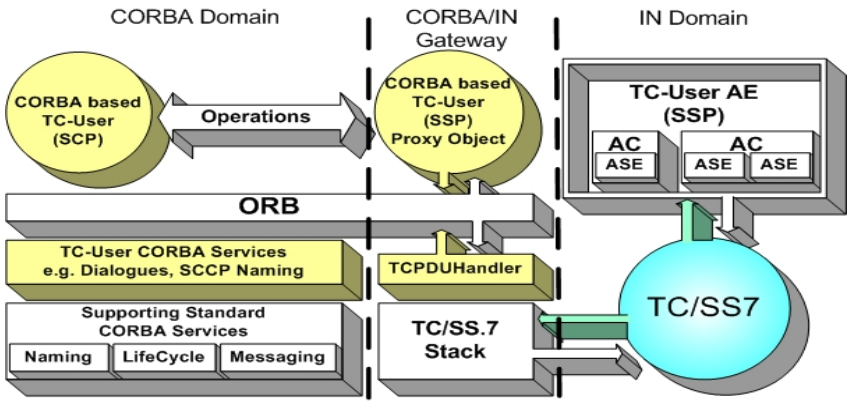

Figura 2.8. Arquitetura com o Gateway TINA/IN

Os detalhes do processo de migração da RI para uma rede de serviços TINA foi proposto através do projeto P508 do EURESCOM [Project P909, 1997], [Project P508, 1997],

[Project P847, 1999], [Project P607, 1997], e gerou vários trabalhos de pesquisa, entre eles [Malini, 2004], [Gobbi, 2005], [Oliveira, 2004] e [Soares, 2003].

A arquitetura do GW foi definida de tal forma que ele poderá ser usado em duas situações, como ilustra a Figura 2.9:

- Acesso de um PAS distribuído a um PCS da RI convencional;
- Acesso de um PAS da RI convencional a um PCS distribuído.

O trabalho [Soares, 2003] analisou e explorou esses contextos de uso do GW, e introduziu uma combinação de recursos com a finalidade de melhorar significativamente o despenho do modelo de GW. Foram definidas estratégias para prover/otimizar a escalabilidade e a vazão na arquitetura ilstrada na Figura 2.9. Essas estratégias apoiaram-se no método, padronizado pelo OMG, de invocação assíncrona de mensagens para otimizar a escalabilidade e numa técnica inovadora de processamento assíncrono de mensagens nos ORBs [Deshpande, Schmidt, O'Ryan, Brunsch, 2002], a ser implementada para prover uma vazão em tempo real.

Figura 2.9. Contextos de uso do GW.

Além de agregar contribuições ao modelo de arquitetura do GW, o mesmo trabalho de pesquisa sugere a aplicação dessas técnicas de provisão conjunta de escalabilidade e vazão, a quaisquer outras arquiteturas de GW que necessitem de operação em tempo real. Entretanto, tal análise foi feita usando um ORB para C++. Não se sabe se um ORB que faz o mapeamento para JAVA terá os recursos suficientes para suportar processamentos assíncronos de mensagens (para aplicações em tempo real).

2.3.2 Foro PARLAY / JAIN

O consórcio denominado Parlay foi fundado em 1998 por várias operadoras de redes e provedores de equipamentos, na sua maioria membros do TINA-C.

O intuito do grupo Parlay foi possibilitar a criação de novos aplicativos através da combinação dos recursos das redes de telecomunicações com os serviços da Internet, e proporcionar novas soluções comerciais que agreguem valores aos usuários finais e todos os participantes envolvidos com os negócios de telecomunicações [Parlay Group].

O objetivo principal do Parlay foi a especificação das APIs de controle de serviços orientadas a objetos, que sejam independentes da tecnologia de comunicação sobre a qual poderão ser montadas (RTP, *wireless* e IP). As APIs estão sendo especificadas em *Unified Modeling Language* (UML) e deverão suportar as tecnologias de *midleware* dominantes tais como as plataformas CORBA e JAVA.

Como mostrado na Figura 2.10, uma API Parlay permite que aplicações localizadas fora do domínio do operador de rede acessem componentes de serviços particulares que estejam dentro daquele domínio.

As APIs incluem interfaces de *framework* e interfaces de serviços. As interfaces de *framework* fornecem uma função de descoberta da API e aspectos de segurança, para assegurar que as aplicações externas sejam conhecidas e tenham autorização para acessar o componente de serviço requisitado. As interfaces de serviço mapeiam os componentes de serviços da API juntamente com os elementos físicos dentro do domínio do operador de rede, para prover o serviço requisitado. Os serviços incluem controle de chamadas, interação com o usuário, mensagens, localização e funcionalidade de conectividade de dados.

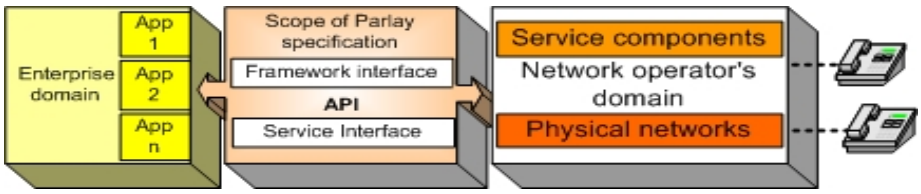

Figura 2.10. A API Parlay

Java APIs for Integrated Networks (JAIN) vem de um consórcio de indústrias formado por provedores e vendedores de equipamentos de rede e telecomunicações, que trabalham no desenvolvimento de ambientes padrões Java para criação e execução de serviços no domínio das telecomunicações.

O objetivo da tecnologia JAIN é a integração das Redes Inteligentes, da Internet e das redes *wireless*, que são as Redes Integradas [Keijzer, Tait, Goedman, 2000]. JAIN visa:

- Portabilidade dos Serviços: as interfaces proprietárias são reconfiguradas para interfaces Java uniformes, para o desenvolvimento de aplicações portáveis;
- Convergência: permite que aplicações e serviços sejam executados na RTP, nas redes IP ou ATM, e redes *wireless*;
- Acesso seguro para as redes: permite que Aplicações que residam fora da rede acessem diretamente os recursos e componentes da rede.

A iniciativa JAIN direciona os mercados de telecomunicações e da Internet, formados por muitos sistemas proprietário fechados, para um ambiente aberto capaz de hospedar uma grande variedade de serviços. JAIN criou dois grupos de especificações:

- *Protocol Expert Group* (PEG): que define APIs Java para os protocolos TCAP, OA&M, ISUP, MAP, SIP, MGCP, H.323, e INAP;
- *Application Expert Group* (AEG): que define APIs para o Controle de Chamadas (JAIN *Call Control*), Ambiente de Execução da Lógica de Serviço (JAIN SLEE) e JAIN Parlay, que permite o acesso seguro às redes.

Particularmente, com referência a RI, a arquitetura da API JAIN para a SS7 define um conjunto de componentes de software que permitem que uma aplicação executada no ambiente Java acesse serviços providos pelo protocolo SS7.

Uma API JAIN para a SS7 define os três maiores componentes de software Java, que são o *Stack*, *Provider* e *Listener*, e eles são definidos na forma de interfaces Java [Bhat, Gupta, 2000]. A Figura 2.11 mostra esses componentes.

Figura 2.11. A Arquitetura da API JAIN para a SS7

Existem especificações de APIs baseadas em Java para os protocolos TCAP, INAP, ISUP, MAP, OA&M. Com relação às RIs, são de particular interesse as APIs para INAP [Sun Microsystems, 1999-1] e para TCAP [Sun Microsystems, 1999-2]. O desenvolvedor da aplicação fica livre para construir programas usando Java, C++, Visual C++ ou Visual Basic.

O objetivo da combinação JAIN - Parlay é definir uma versão na linguagem Java de uma API Parlay usando os melhores princípios da tecnologia Java, enquanto mantém a integridade da API Parlay original.

Uma das vantagens relativas ao uso da versão JAIN de Parlay é a possibilidade de explorar os benefícios da tecnologia Java, tais como portabilidade de serviços, facilidade de programação e uso de estruturas como Java Beans.

2.3.3 Foro PINT / SPIRITS

O grupo de trabalho conhecido como *PSTN/Internet INTernetworking* (PINT) faz parte do *Internet Engineering Task Force* (IETF) e foi criado em 1997 para desenvolver metodologias para convergência das RIs com as redes IP. Seu principal objetivo consistia em possibilitar que os assinantes da Internet também utilizassem as RIs, inclusive as redes móveis.

A Figura 2.12 ilustra o modelo da arquitetura PINT. Tal modelo contém duas nuvens, uma pertencente à arquitetura PINT com seus servidores ligados aos clientes PINT e um GW que se conecta a outra nuvem pertencente à rede de telefonia. Tal GW é usado para interconectar a ilha PINT com o Sistema Executivo pertencente à rede de telefonia, que é responsável pelas conexões na RTP. Fisicamente o sistema executivo poderia ser um *Service Node* ou um *Service Control Point* da RI [Grech, Mckinney, Stanaway, Varney, Vemuri, 2000].

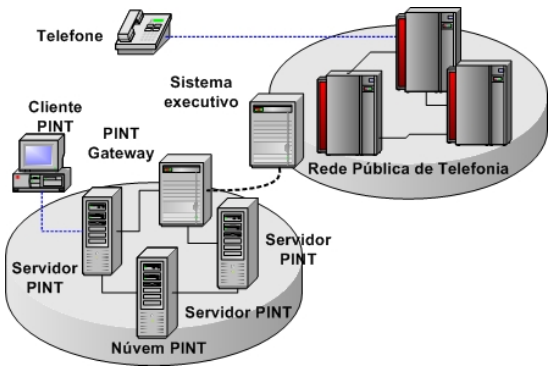

Figura 2.12. Modelo da Arquitetura

O propósito geral da iniciativa PINT é integrar os recursos da internet aos serviços de telefonia e realizar a padronização do acesso dos usuários da internet aos PCSs da RI. Ela também se propõe a desenvolver novos serviços executados parcialmente no domínio da Internet e parcialmente no domínio da telefonia convencional.

Outra rede que faz o inverso da rede PINT, também criado pelo IETF é denominada *PSTN/IN Requesting Internet Services* (SPIRITS). Sua função é ampliar os serviços das RIs com as capacidades IP. Portanto, sua meta é tornar os aplicativos, e conteúdos da Internet, acessíveis aos usuários da rede de telefonia convencional. São exemplos de serviços SPIRITS, *Internet Call Waiting, Internet Caller-ID Delivery, Internet Call Forwarding,*os quais são iniciados por elementos da RI [Faynberg, Gato, Lu, 2002].

Tais serviços tratam de como os serviços suportados pelos elementos da rede IP podem ser inicializados através das requisições RI, assim como os arranjos de protocolos pelos quais a RTP pode requisitar ações a serem transportadas na rede IP em resposta aos *triggers* (eventos) que ocorrem dentro da RTP/RI. A rede SPIRITS se preocupa com a arquitetura e com os protocolos responsáveis pelo transporte seguro das informações dos *triggers* (requisições para ações, notificações de eventos etc.) da RTP/RI para a rede IP, e as respostas da rede IP para a RTP/RI. Sua arquitetura inclui três entidades independentes: o cliente, o servidor e o sistema de requisição da RTP/RI [Gurbani, Faynberg, 2004].

O cliente é o elemento que requisita as notificações ou ações a serem executadas na rede IP. O servidor é o elemento que recebe as notificações ou requisições da RTP/RI e as responde opcionalmente, enquanto inicia a execução dos serviços requisitados no domínio IP. O servidor e o sistema de requisições residem no domínio IP.

A Figura 2.13 ilustra o modelo de arquitetura das redes PINT/SPIRITS juntas [Faynberg, Gato, Lu, 2002]. O cliente SPIRITS fica localizado no PCS ou se comunica com este através da interface D invocando um serviço para o GW SPIRITS pela interface C, que processa a mensagem e a envia ao servidor SPIRITS. O pedido de serviço para o servidor PINT, que pode estar localizado no PCS ou no Sistema Executivo, sempre é iniciado pelo cliente PINT, independente da arquitetura em que se encontra.

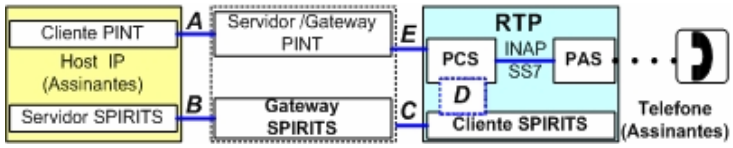

Figura 2.13. A Arquitetura das redes *PINT/SPIRIT*

O cliente PINT e o servidor SPIRITS estão localizados no *host* IP e podem ser programados para executar um processo.

A meta desse grupo de trabalho é descrever os blocos de construção para os serviços RTP/IP que iniciam com as requisições da RTP/IN e não padronizar esses serviços propriamente ditos. Seu foco está orientado aos eventos e não no projeto orientado a serviços.

2.3.4 Comparação entre os foros

A Tabela 2.3 ilustra os objetivos de cada foro, bem como a situação atual deles.

Tabela 2.3.Comparação entre os foros

Foro	Objetivo Principal	Observações
TINA	Solucionar as limitações das RIs, introduzindo, gradualmente, sistemas distribuídos nos elementos de execução dos serviços, suportados por uma plataforma compatível com o CORBA.	Em 2000 o TINA-C encerrou seus trabalhos, e como muitos de seus membros também faziam parte do OMG, esse continuou trabalhando com os mesmos objetivos do TINA-C [Business Wire, 2000].
Parlay	Especificar APIs de controle de serviços orientadas a objetos, que sejam independentes da tecnologia de comunicação sobre a qual poderão ser montadas (RTP, *wireless* e IP) e deverão suportar as tecnologias de *midleware* dominantes tais como as plataformas CORBA e JAVA.	A versão mais atual é *Parlay X 3.0 Third Party Call Control*, que permite, por exemplo Mensagens Multimídia, Localização do Terminal, Conferência Multimídia, entre outros serviços [Parlay X3.0, 2007].
JAIN	Definir um conjunto de APIs Java que permitem o rápido desenvolvimento de produtos e serviços em comunicações baseados nas redes de próxima geração.	Associado ao protocolo SIP, a API JAIN SIP encapsula as funções de diferentes versões desse protocolo com uma interface Java padronizada, de modo que o processamento de chamadas inicialização de sessões estão disponíveis na rede. [JSR 32, 2006]
PINT	Possibilitar que os assinantes da internet também utilizassem as RIs, inclusive as redes móveis.	PINT transmite de voz sobre pacotes IP (VOIP) para chamadas entre a RTP e a Internet [PINT, 2008].
SPIRITS	Ampliar os serviços das RIs com as capacidades IP, tornando os aplicativos, e conteúdos da internet, acessíveis aos usuários da rede de telefonia convencional.	O documento [IETF document, 2002] lista os requisitos gerais para esse protocolo, bem como aqueles pertinentes à RI, RI Wireless, e blocos de construção PINT.

2.4 O Gateway RI/CORBA

A idéia da RI distribuída se apoiou nos ambientes definidos pelo padrão CORBA, para suportar as aplicações referentes aos PASs, PCSs, PDSs e PGSs. Uma vez que essa

arquitetura esteja funcionando plenamente, tais entidades físicas poderão não existir mais como as conhecemos hoje, já que as funções que atualmente lhes são atribuídas estarão espalhadas pela rede, comunicando-se através do General Inter-ORB Protocol (GIOP) sobre a SS7 [Soares, 2003]. Esse novo conceito de RI traz vários benefícios, que culminam numa maior facilidade e rapidez para inserir e gerenciar serviços, a um custo bem menor do que o atual.

Há um longo caminho a percorrer até que se tenha uma RI totalmente distribuída. As restrições financeiras e tecnológicas estão impulsionando a implantação dessa arquitetura por partes. Neste trabalho, supõe-se que as funcionalidades suportadas por PCS+PDS+PGS serão executadas em um ambiente distribuído baseado no CORBA, enquanto os PASs permanecem centralizados. corresponde ao passo inicial na implantação de soluções TINA para a RI. Especificamente, são tratadas as comunicações entre os PASs convencionais e os PCSs distribuídos.

CORBA é um padrão do OMG. Através dele, diferentes aplicações executadas em plataformas distintas (do ponto de vista de hardware e software) podem interagir entre si facilmente. Neste trabalho foram usados apenas ORBs compatíveis com o CORBA, para códigos em JAVA.

Para a realização do cenário de integração proposto, é necessária uma entidade de ligação entre a RI distribuída e a RI convencional (centralizada nos PCSs), com a função básica de converter as mensagens da SS7 em invocações de métodos no ambiente CORBA. Trata-se de um gateway, cuja especificação já é de domínio do OMG [OMG, 1998], e que é o foco deste trabalho. O objetivo é implementar suas funcionalidades, modeladas a partir das especificações do OMG.

O gateway RI/CORBA abordado aqui é o GWS, constituído de definições de tipos e interfaces que representam objetos com funcionalidades muito bem definidas no processo de interação entre a RI convencional e os objetos CORBA. Sua inserção na comunicação entre o ambiente CORBA e a RI convencional pode ser vista na Figura 2.14.

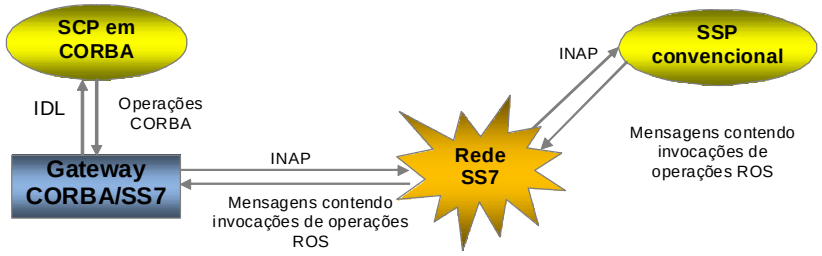

Figura 2.14. Interação entre PAS tradicional e PCS CORBA através do Gateway

Um estudo completo sobre a inserção deste GWS na proposta de evolução das RIs pode ser encontrado em [Soares, 2003], um dos textos-base para a produção deste trabalho.

No final do ano de 1997, o grupo OMG apresentou uma *Request for Proposals* (RFP) na qual solicitava propostas para a interação entre os domínios SS7 das RIs e o domínio CORBA. Como resposta a essa RFP, um grupo de empresas e centros de pesquisa apresentou no ano de 1998 o documento [OMG, 1998], no qual foi especificada uma arquitetura de objetos distribuídos que atenderia aos requisitos do OMG para o problema proposto. A partir da especificação proposta nesse documento foi desenvolvida a programação do GWS apresentada neste trabalho.

O gateway proposto deverá realizar as conversões necessárias de sintaxe e semântica entre os domínios SS7 e CORBA. Para atingir esses propósitos ele foi modelado com um conjunto de interfaces e objetos, e poderá também contar com serviços adicionais de ORB do ambiente CORBA, tais como serviço de localização de objetos (*Naming Service*), serviço para criação e destruição de objetos (*Life Cycle Service*).

2.5 Evolução da Rede de Telefonia Fixa em direção à NGN

Está claro que o caminho tomado pelas companhias operadoras da rede de telefonia fixa apontou para a construção de uma arquitetura de rede distribuída com uma camada comum para o transporte de dados e voz baseada na rede de pacotes. Não há dúvidas também de que para crescer no mercado de serviços de dados e multimídia, essas operadoras não poderão desconsiderar a perspectiva de inserir suas redes dentro dos propósitos da NGN.

Dependendo de como as RTPs de uma operadora estejam estruturadas, ela pode adotar diferentes estratégias de migração para se inserir no contexto da NGN. São vários ângulos que precisam ser questionados na tomada de uma direção:

Como otimizar o que já está instalado para reduzir despesas operacionais e reduzir gastos no passo-a-passo de integração com a NGN?

Os serviços ofertados atualmente deverão ser mantidos com a criação de uma camada NGN de acesso banda-larga para serviços multimídia?

Quais componentes da RTP deverão ser substituídos por componentes equivalentes da NGN?

Sem querer tecer considerações sobre os aspectos relacionados com a gerência dessas redes ou mesmo com a QoS das redes de transporte, o cenário de migração apontado aqui serve para exemplificar como os produtos dos vários foros descritos anteriormente poderão ser incorporados no processo de migração de uma RTP em direção a uma NGN completa. Pelo menos dentro do que existe hoje e das promessas tecnológicas, esse parece ser um caminho realizável.

O que se observa em todas as pesquisas que estão sendo feitas, é que a intenção não é a de substituir as redes atuais, pelo contrário, no que concerne à RTP o interesse é em mantê-la viva e forte dentro de um mercado cada vez mais competitivo. O que se quer é expandir capacidades para gerar novos nichos no mercado, graças à convergência de voz com dados. O cenário descrito a seguir se baseia numa proposta da Alcatel [Alcatel, 2001].

Não faz parte do escopo deste trabalho estudar as tecnologias mais propensas para compor cada camada da NGN, nem discutir os protocolos e APIs. O exemplo de cenário a seguir tem como móvel apenas mostrar a extensão da revolução que ocorrerá na construção de uma arquitetura de convergência do IP, telefonia e redes óticas cuja ênfase será a criação de serviços inovadores para os diversos usuários finais.

2.5.1 O ponto de partida

A Figura 2.15, extraída de [Alcatel, 2001], representa o estagio atual da RTP. Parte-se do princípio que o tráfego de voz é transportado sobre TDM e que a rede de sinalização para as centrais de comutação é a SS7, como indicado em [A]. Os serviços agregados estão instalados em parte nas próprias centrais locais e em parte nos PCSs da RI, como indicado em [B]. Supõe-se também que a operadora de rede ofereça conectividade aos provedores de serviços Internet através de serviços de discagem (RTP ou RDSI) ou através de acesso banda

larga por redes baseadas em pares de cobre tipo *Asymetric Digital Subscriber Line* (ADSL), por exemplo, mostrado em [C].

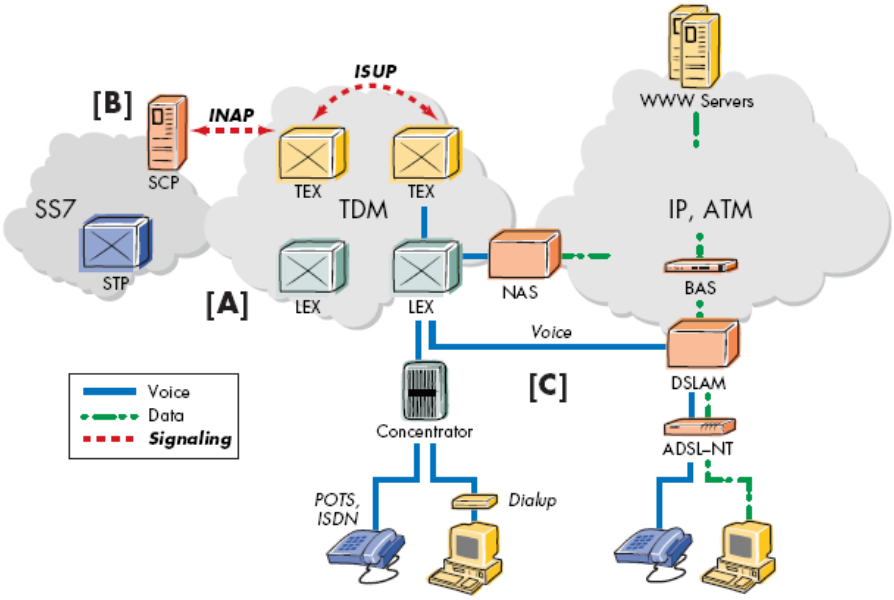

Figura 2.155. A RTP atual

2.5.2 Otimização da infra-estrutura de rede

O próximo passo representa o fortalecimento da estrutura existente, instalando centrais de comutação mais avançadas e com maior capacidade [D], criando mais nós de acesso às centrais [E] e, otimizando o acesso banda larga através de voz sobre ADSL [F].

O PCS da RI pode ser usado nesse passo como um meio de integrar voz e dados em aplicações comuns, como ilustrado em [G]. Exemplos de convergência das aplicações RI-Internet são: chamada em espera na Internet, clique para discar (*click to dial*), chamada amplificada na Web (*Web- augmented calling*), mensagem unificada, etc. Para se comunicar com os servidores da Internet, o PCS terá que adotar as soluções PINT e SPIRITS descritas anteriormente, mostrado em [G]. A Figura 2.16, extraída de [Alcatel, 2001], ilustra essas alterações na RTP. No item 2.5.4 será explicada a inserção do ApGW e do servidor de aplicações, ilustrados em [H].

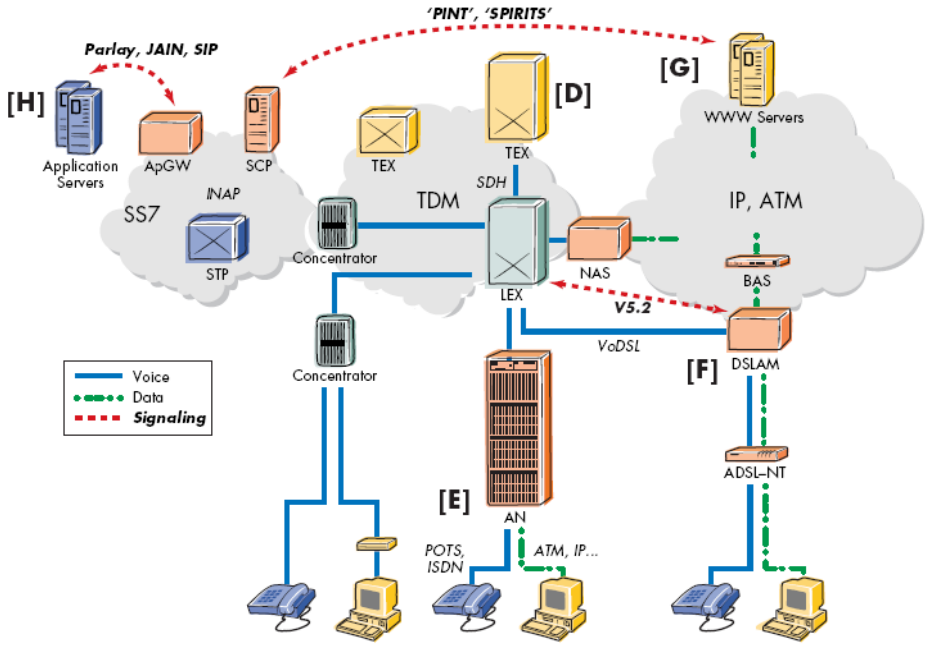

Figura 2.166. Consolidação da RTP

2.5.3 Instalação do Gateway de Tronco

Tendo em vista que uma das metas básicas da NGN é a migração para uma infra-estrutura de pacotes, com menor custo operacional, presumivelmente, o transporte de voz poderá migrar suavemente para a tecnologia IP ou ATM. O foco das operadoras de rede, nesse ponto, será a instalação de GW de Tronco (GWT) na rede TDM, mostrado na Figura 2.17, pela indicação [I].

Inicialmente as atuais centrais locais poderão ser estendidas com o GWT para converter os sinais de voz TDM em pacotes IP, garantindo assim a proteção dos investimentos em TDM, bem como o acesso aos serviços da RI.

Pode-se ainda adicionar um GWT externo, mostrado em [J], onde não existam centrais de comutação estendidas com GWT, controlado por um softswitch (através de protocolo MEGACO ou H.248). Do ponto de vista funcional, um softswitch (Classe 4) atua semelhantemente a uma central de comutação Trânsito, com os mesmos recursos de roteamento, interfaces de sinalização (INAP, ISUP) e com acesso aos serviços da RI, ilustrado em [K].

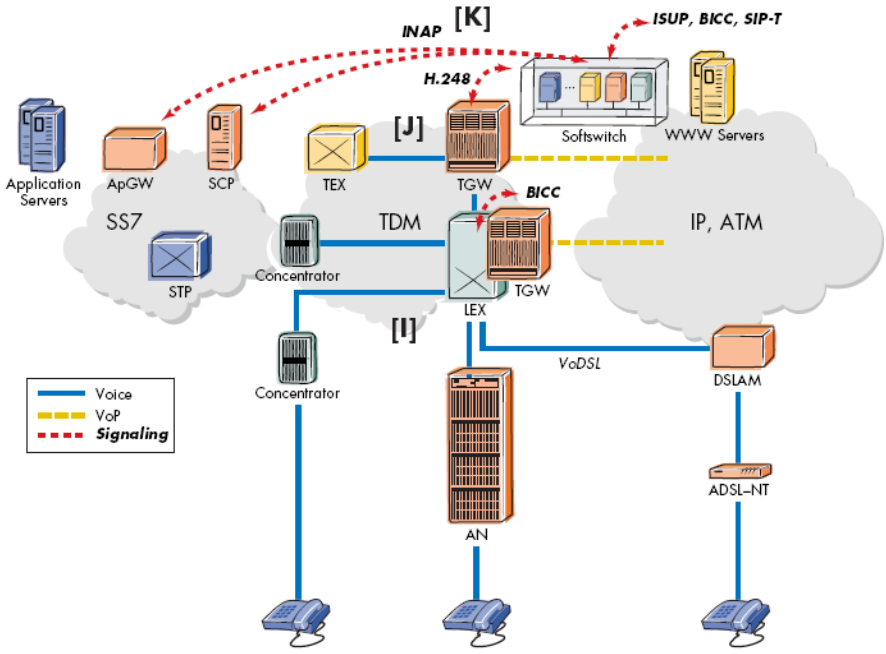

Figura 2.177. Introdução do GWS e GWT

2.5.4 Introdução de um GWS

Para se inserir no conceito de NGN e ganhar com a introdução de serviços abertos, a operadora de rede poderá introduzir nova camada na RI existente, instalando gateways de aplicação ou de sinalização (na Figura 2.17 o GWS é representado por ApGW), como também são chamados, para acessarem servidores de aplicação externos (propriedade de terceiros), através de interfaces abertas (tais como Parlay, JAIN e SIP mostradas na Figura 2.16 no passo [H]).

Neste ponto do cenário é que serão inseridos os recursos de software propostos e construídos neste trabalho, usando a plataforma CORBA, como indicado na Figura 2.16. A arquitetura do gateway proposta será apresentada no próximo capítulo.

2.5.5 Instalação do Gateway de Acesso

Nos mercados onde se prevê crescimento rápido ou onde há um investimento agressivo em acesso banda larga (ADSL, LMDS, cabo), as operadoras podem introduzir a tecnologia de voz sobre pacote para capturar o crescimento na rede de acesso, ou como um

meio de assumir parte da necessidade de processamento das centrais de DSL, como mostra a Figura 2.18, extraída de [Alcatel, 2001].

Um Softswitch Classe 5, como mostrado em [L], pode ser compartilhado por diferentes alternativas de gateways de voz, dependendo da topologia dos usuários de voz e requisitos de serviços, dentre outros, usando os mesmos protocolos (H.248, MEGACO).

Os assinantes ADSL podem instalar um GW residencial (GWR) ou um dispositivo de acesso integrado (DAI), ilustrado em [M], com capacidade para codificar voz sobre pacote.

Alternativamente, um operador ADSL pode resolver estender as ADSLAM com funcionalidades de GW de voz sobre pacote (AGW), como indica [N], [O] e [P].

Para incluir novos terminais de geração de voz (os telefones IP), o *softswitch* poderá também terminar outros protocolos, tais como, H.323 e SIP.

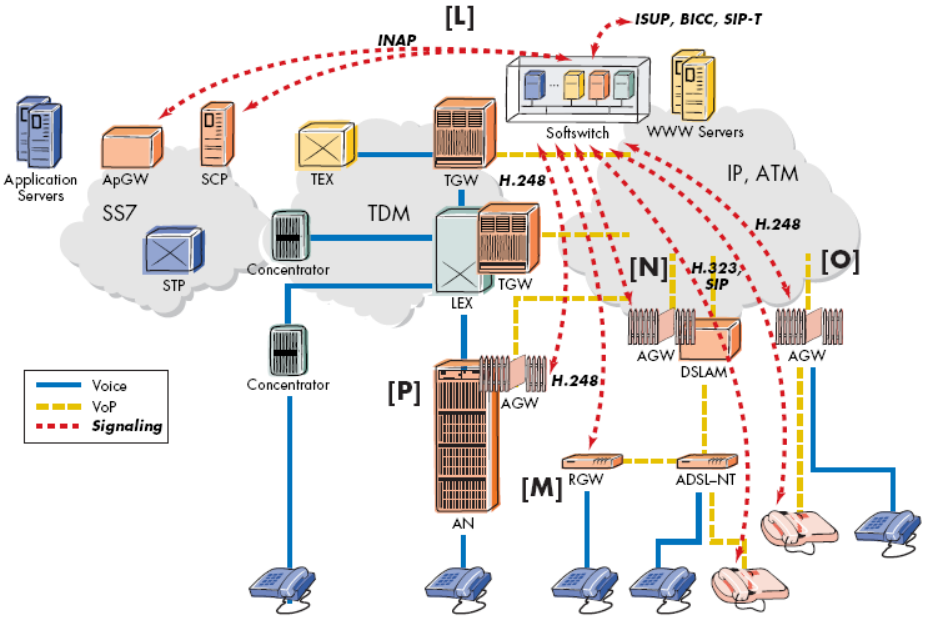

Figura 2.188. Introdução do gateway de Acesso

2.5.6 Introdução de Multimídia

É bem provável que a curto ou médio prazo o serviço de voz seja predominante mesmo na NGN. Entretanto, o acesso banda larga possibilita um grande leque de serviços de dados e multimídia, como apresentado na Figura 2.19 extraída de [Alcatel, 2001].

- Clientes IP, indicados em [R], com Softswitch Multimídia [S]:

Um pré-requisito para o emprego de serviços multimídia é a disponibilidade de terminais apropriados. Os computadores pessoais de hoje são um bom ponto de partida, mas é esperado que a convergência de computadores, consumidores e tecnologias de comunicações resultará em novos dispositivos multimídia. Esses novos terminais se comunicarão com o softswitch por meio de protocolos como H.323 e SIP.

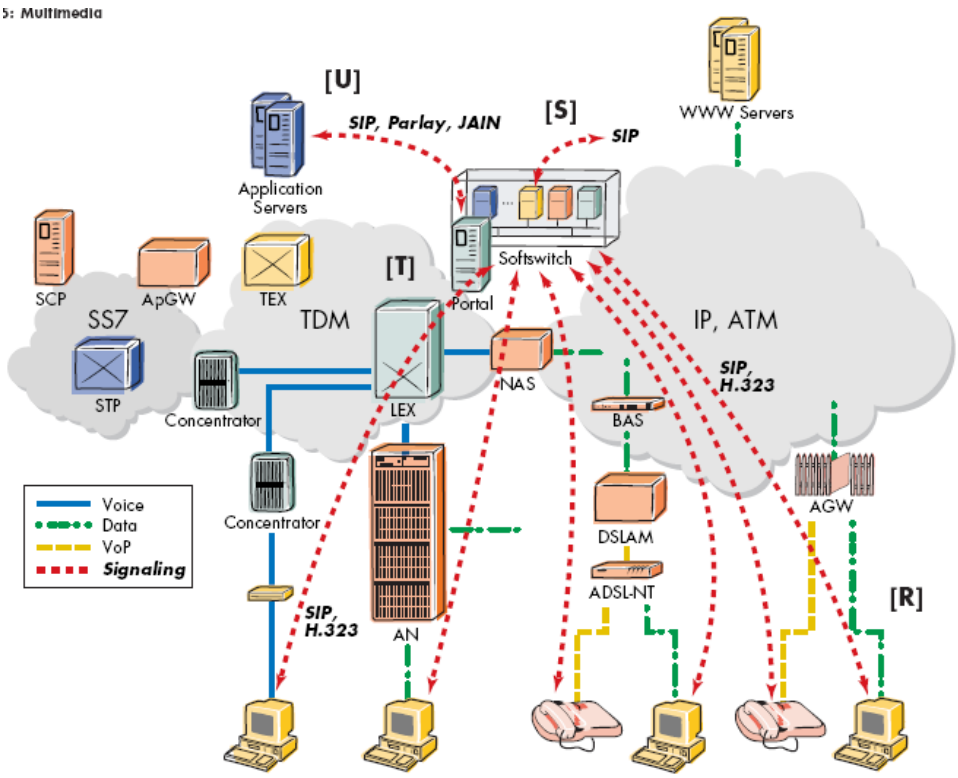

Figura 2.199. Introdução de Multimídia

- Portais e interfaces abertas [T]:

Com a introdução de novos modelos de negócios e de novos participantes (operadores de redes virtuais, provedores de aplicações, provedores de conteúdo) haverá necessidade de proporcionar o acesso adequado (para autorização, autenticação, definição de perfil, cobrança etc.) e as referidas plataformas (para negociação das capacidades de terminais, de largura de banda, agregação de conteúdo etc). A introdução de portais proporciona aos operadores de redes tanto quanto aos novos participantes, novas oportunidades de negócios, além de separarem claramente o que é controle de rede das funções voltadas para os serviços. Em uma arquitetura NGN completa, aplicações e rede se comunicarão

através de protocolos padronizados, como por exemplo o SIP, e APIs, como JAIN e OSA/PARLAY.

- Novas aplicações [U]:

O aumento do portfólio de aplicações de dados e multimídia é um pré-requisito para os provedores de serviços de telecomunicações gerarem, crescerem e diferenciarem novas fontes de renda.

2.5.7 Migração para a NGN completa

Como um passo final de migração em direção à NGN completa, os equipamentos remanescentes da RTPC serão transformados ou substituídos por componentes de rede compatíveis com a NGN, apresentado a Figura 2.20, extraída de [Alcatel, 2001].

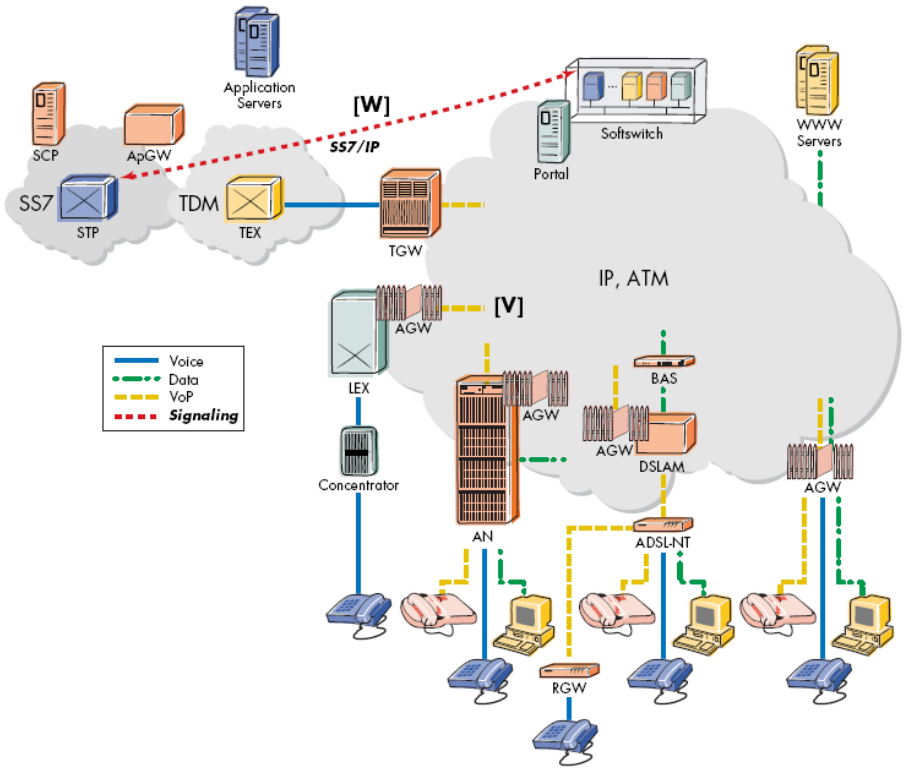

Figura 2.200. A NGN completa

No final de suas vidas úteis, os equipamentos TDM e nós de acesso poderão ser transformados ou substituídos por GWT, GWA e softswitches, como mostra [V].

Enquanto as camadas superiores são mantidas intactas (SCCP, ISUP, TCAP, INAP), as camadas inferiores da rede SS7 serão substituídas por outras, tal como definido pelo grupo SIGTRAN (SIGTRAN – Signaling Transport) do IETF (não abordado neste trabalho), ilustrado em [W].

2.6 Considerações Finais

A NGN não substituirá as redes existentes, mas expandirá gradualmente suas habilidades de geração de fontes de renda graças à convergência de voz e dados.

A introdução de gateways nas camadas de rede, começando pelos GWSs, representa a iniciativa que ajudará a consolidar a NGN como a Arquitetura de Rede com capacidade de ofertar um amplo leque de produtos em todas as camadas de rede, como mostrado brevemente no item 2.5.

Dentro do espectro de soluções ventilado, verifica-se a importância da pesquisa proporcionada neste trabalho, que reflete a preocupação em produzir um software a ser inserido num GWS, para prover as comunicações entre a rede SS7 e um servidor de aplicações adicionado no âmbito da RI. Além da construção desse software, é de fundamental importância que as comunicações veiculadas por meio dele não degradem o desempenho da SS7, nem afetem a confiabilidade de suas operações na RI. Sendo assim, a construção desse software deve vir acompanhada de recursos de preservação da QoS instalada na RI.

3. Construção do Software para o GWS

3.1 Introdução

Tomando como base a etapa de inclusão do GWS na RI, exposta no Capítulo 2, o problema de especificação do software dele pode ser separado em duas partes: uma envolvendo a comunicação do GWS com o servidor, que corresponde à estratégia de introdução de uma camada na RI usando uma plataforma distribuída para prover serviços abertos aos usuários da RTP por parte de provedores de aplicação externos; outra parte é a própria construção do GWS para acessar os servidores de aplicação. Cuidados especiais nesse processo são necessários para resguardar a confiabilidade e o desempenho da rede, por que:

- Não pode haver atrasos significativos no acesso ao servidor de aplicação e nem na entrega dos serviços;
- Não pode haver perda de informações;
- A parte referente a RI distribuída deverá manipular um volume de tráfego sempre crescente;
- Vazão e escalabilidade são parâmetros cruciais nas comunicações entre o GWS e os servidores de aplicação.

Pensando nisso tudo, buscou-se o modelo do GWS especificado no OMG, como base para a construção do software usado para mediar as comunicações entre o GWS com os servidores externos (servindo como se fossem PCSs distribuídos) formando uma RI distribuída. Essa pode ser uma camada superior construída sobre a RI atual como um passo inicial no estabelecimento da NGN.

3.2 A Arquitetura a ser Construída

Usando as especificações do OMG para o modelo de GWS baseado em uma plataforma CORBA a ser usado nas aplicações do tipo cliente/servidor, propõe-se neste

trabalho a construção de um software contendo os elementos necessários para a mediação das comunicações entre os usuários da RTP, via PASs, com os serviços abertos agregados por provedores externos à RTP (atuando como PCSs distribuídos). Propõe-se também a inclusão de recursos adicionais ao modelo de GWS, para melhorar o desempenho dele e dos servidores distribuídos, com o fim de acrescentar capacidades de vazão e de escalabilidade referentes ao tráfego de mensagens na RI distribuída [Soares, Durães, 2008-1]. A Fig. 3.1 ilustra a arquitetura proposta.

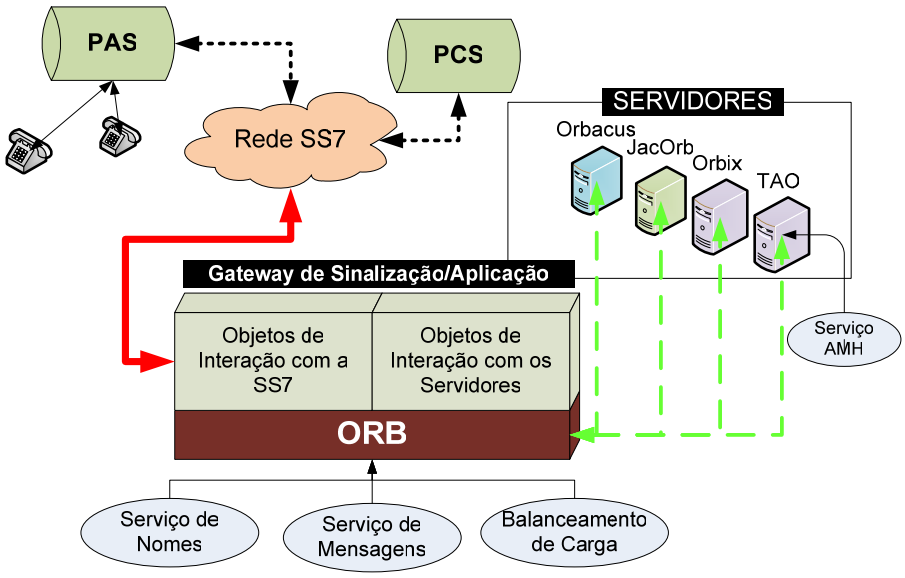

Figura 3.1 Acesso da RI tradicional à camada de Serviços da NGN.

De um lado os usuários da RI convencional acessam os serviços executados no atual PCS centralizado através de PASs localizados em centrais de comutação trânsito da RTP. Eles também acessam os serviços avançados abertos, da camada de aplicações da NGN (Servidores de Aplicações) por meio do GWS. O software do GWS pode ser separado em duas partes: uma que interage com a SS7 e faz a tradução de linguagens e de endereços (não tratada neste trabalho), e outra que contém os objetos de interação com os servidores CORBA externos, os quais em conjunto formam a Arquitetura de Serviços da NGN. Esta parte pertencente inteiramente ao domínio da rede distribuída é que se pretende detalhar.

Uma das vantagens da tecnologia CORBA é que ela é flexível quanto à linguagem usada na construção do software do cliente e do servidor. O software do GWS (atuando como cliente) pode ser escrito em Java ou C++, independente das linguagens usadas nos programas

dos servidores. Na Fig. 3.1, o JacOrb, o ORBACUS e o Orbix são exemplos de ORBS para Java e o TAO, para C++.

3.3 As Interfaces do GWS

O GWS é um elemento chave para que os usuários da RTP acessem os serviços criados na camada de aplicações da NGN. Sua função primária é de traduzir endereços, mensagens e diálogos do domínio SS7 para o domínio CORBA e retornar os serviços executados para a RI convencional. Em outras palavras, o GWS representa um conversor de invocações de operações e seus retornos entre os dois domínios. Essa conversão, entretanto não é direta, pois um ambiente possui características bem distintas de execução dos serviços do outro. Além disso, os parâmetros de QoS da RI precisam ser preservados, particularmente no que se refere ao tempo de acesso aos serviços. Portanto, no domínio CORBA, deverá existir algum mecanismo que possibilite a troca de mensagens em tempo real.

Verificou-se que uma arquitetura adequada para suportar as comunicações entre os dois ambientes, dependia de algoritmos para a tradução dos dialetos INAP para IDL. Outra questão igualmente imprescindível seria a adaptação das semânticas de execução das aplicações entre os dois ambientes. Em outras palavras, a conversão das primitivas de diálogo e de controle de conexão da SS7 em interações compatíveis no ambiente distribuído [OMG, 1998]. Assim, o software de conversão do GWS foi separado em duas partes, conforme ilustra a Figura 3.2 extraída de [Oliveira, 2004].

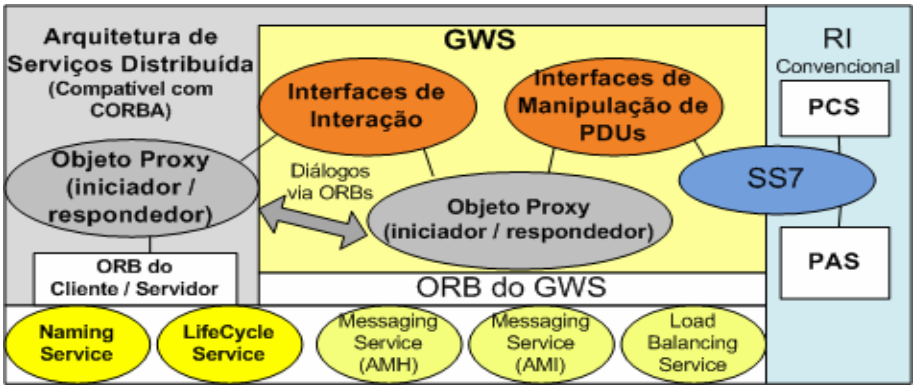

Figura 3.2. As interfaces do GWS

Observa-se a existência de algumas interfaces orientadas a *Protocol Data Unit* (PDUs), necessárias para padronizar o acesso da RI ao GWS. Outras interfaces e objetos são devotados às interações do GWS com o ambiente externo CORBA.

3.3.1 As Interfaces Orientadas a PDUs

Na arquitetura de interação SS7/CORBA da RI, o grupo de interfaces orientadas a PDU compreende três interfaces: TcPduUser, TcPduProviderFactory e TcPduProvider. Essas interfaces, ilustradas na Figura 3.3 extraída de [Soares, 2003], são definidas como segue:

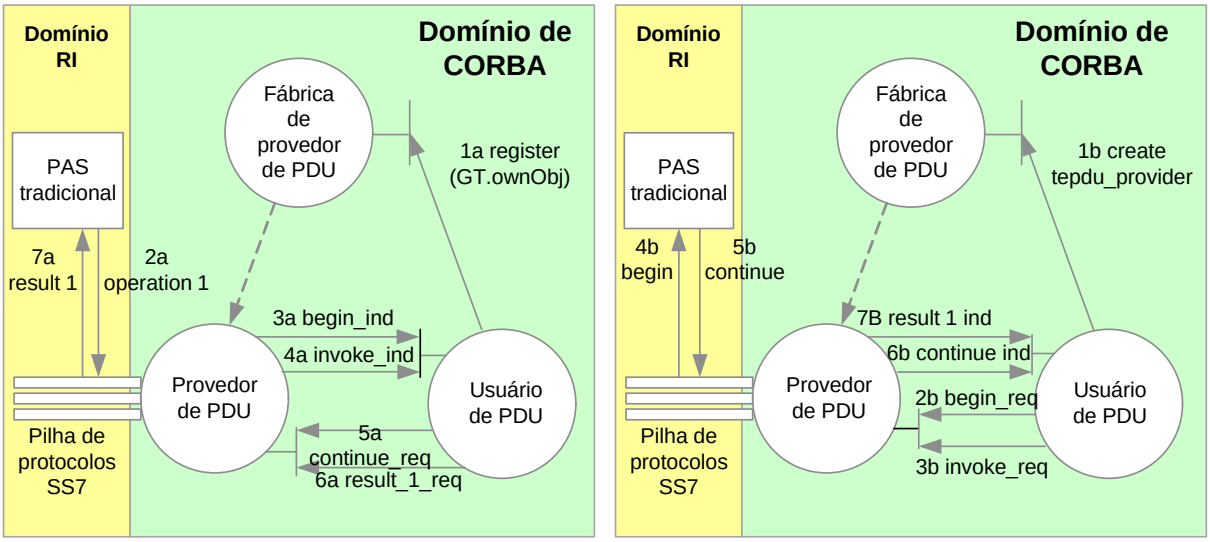

Figura 3.3. Interfaces orientadas a PDUs.

- TcPduProviderFactory: possibilita que um objeto TcPduUser registre seu interesse em receber notificações de mensagens destinadas a um endereço em particular (no caso é um PCS-CORBA) e um determinado contexto de Aplicação. Também possibilita a criação de objetos TcPduProvider;
- TcPduProvider: encapsula a pilha de protocolos da SS7 no domínio CORBA;
- TcPduUser: Inicia mensagens para a SS7 ou recebe mensagens da SS7 via TcPduProvider.

A Figura 3.3 ilustra também como essas interfaces podem ser usadas. No lado esquerdo da figura a chamada inicia no lado da RI convencional e no lado direito inicia no lado CORBA.

A comunicação dos PASs com a parte do GWS que contém os objetos criados a partir da definição dessas interfaces não foi simulada neste trabalho. A razão para tal é que seria necessário desenvolver uma ontologia para a conversão de protocolos SS7 para IDL (sintaxe e semântica) e gerar tráfego de mensagens reais de serviços RI. Essa abordagem representa um esforço a mais de pesquisa e de programação muito complexos, motivo pelo qual não foi planejado como produto desta dissertação de mestrado.

3.3.2 As Interfaces de Interação

As interfaces de interação executam três funções básicas: localização das aplicações, inicialização e manutenção dos diálogos. Elas foram definidas como segue e estão ilustradas na Figura 3.4:

- TcUser: interface que herda as características do serviço de ciclo de vida de CORBA (*COS LifeCycleObject*). Ela define quais são as operações suportadas pelos objetos PASproxy e PCS, para criar, terminar e abortar uma associação;
- TcFactoryFinder: interface que fornece as operações necessárias para que um objeto registre com um Serviço de Nomes (*Naming Service*) uma operação para localizar objetos fábrica. Ela também provê um método para substituir as informações no serviço de nomes;
- TcUserGenericFactory: interface que define operações para a criação de objetos (*create*). É um meio comum para a geração de todas as fábricas de objetos. As fábricas de PASProxy e de PCS mostradas na Figura 3.4 são geradas a partir dela;
- TcRepository: Interface que padroniza o acesso à informação gerada durante a conversão de especificações, fornecendo os mapeamentos dos códigos de operações, erros e extensões, em nomes de referência na linguagem IDL;
- GwAdmin: Provê um ponto de contato para o acesso a todas as funções relacionadas ao gateway. Particularmente, ela define os atributos de leitura para as interfaces TcFactoryFinder e TcPduProvider.

Toda entidade de aplicação pode ser localizada pelo seu título global. Ele é usado como um contexto de nomes para o Serviço de Nomes. A localização deste contexto de nomes é publicada na interface TcServiceFinder (Localizador de Serviço). Através deste serviço pode-se encontrar a fábrica associada a determinado título global.

O processo de inicialização de um diálogo pode ser visualizado na Figura 3.4. Um objeto TcUser usa a interface GwAdmin para obter a referência ao objeto TcFinderFactory (1) e invoca uma operação *resolve* para obter a referência ao objeto fábrica correspondente ao GT (global title) passado como parâmetro na invocação (2). A interface TcFinderFactory usa o serviço de nomes para executar a resolução de nomes e localizar a fábrica em questão (no caso, SCPFactory) (3). A referência é usada para criar um objeto SCP (4).

As nomenclaturas na Figura 3.4 são as mesmas do documento original. O SSP Proxy corresponde ao PAS Proxy e o SCP ao PCS mencionados anteriormente.

Para converter o identificador da operação INAP recebida em um nome em IDL, o SSPProxy usa as informações contidas no TcRepository (5). De posse da informação o SSPProxy pode usar o repositório de interfaces de CORBA para obter a operação em IDL a ser invocada (6).

O SSPProxy invoca a operação IDL equivalente à operação INAP original e a troca de invocações de operações continua até o final da associação.

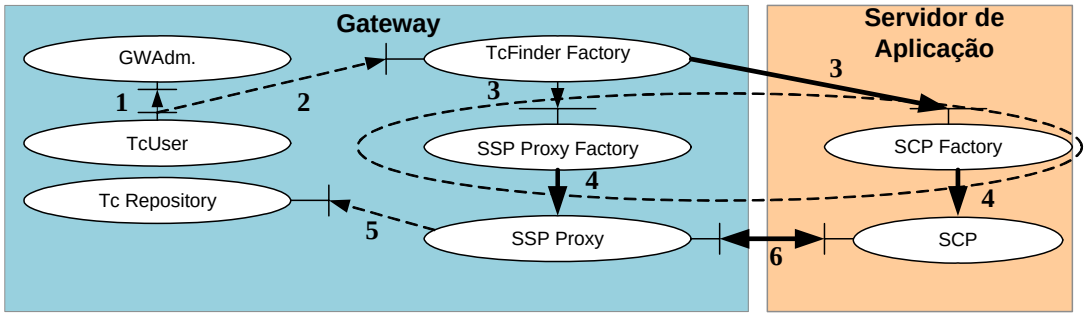

Figura 3.4. O processo de localização de uma aplicação e inicialização de um diálogo.

Os processos de localização e inicialização do diálogo entre o SSPProxy e o SCP asseguram que um objeto tenha a referência do outro, como forma de controle da manutenção do diálogo. Além disso, quando o diálogo inicia, é alocado um identificador de associação a_id, de forma que todas as chamadas de operações do diálogo carregam esse parâmetro. Desta forma, o GWS pode distinguir qual diálogo no domínio da SS7 deve ser usado para enviar PDUs relacionadas a uma invocação CORBA em particular. Esse tipo de controle também possibilita que um determinado objeto (SCP, por exemplo) multiplexe vários diálogos, provendo alguma *escalabilidade.* No entanto, devido ao grande volume de tráfego que deverá ser suportado, é preciso incluir mais recursos no GW e no servidor de aplicações para obter-se um grau de escalabilidade maior.

O exemplo a seguir descreve o processo de criação do objeto SSPProxy não mostrado na Figura 3.4 e o diálogo completo entre as interfaces de interação.

3.3.3 Exemplo de Comunicação entre um PAS Tradicional e um PCS-CORBA

A Figura 3.5 ilustra um exemplo onde um PAS (SSP na figura) na RI tradicional interage com um objeto PCS (SCP na figura) no ambiente CORBA. Trata-se de um exemplo fictício, mas é todo baseado na referência [OMG, 1998]. Os passos básicos para a realização dos diálogos são descritos como segue:

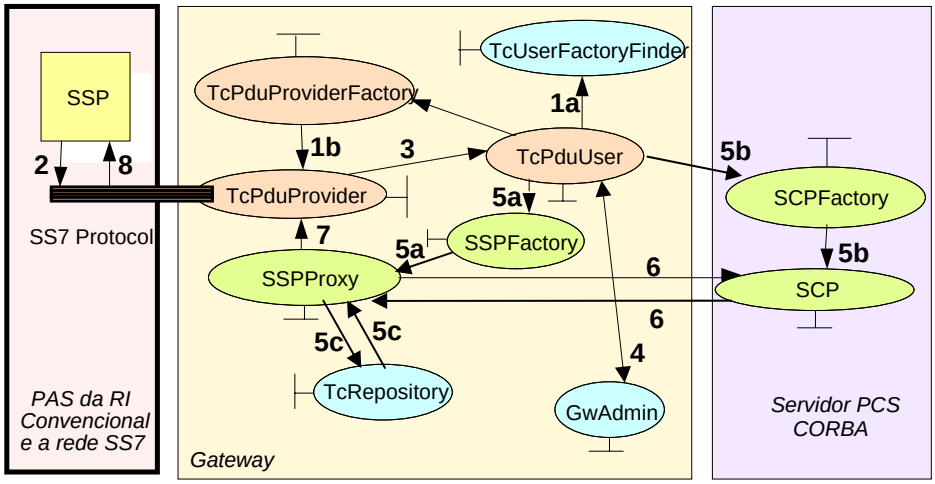

Figura 3.5. As interações no Gateway

1.a Um objeto *TcPduUser* registra no objeto *TcPduProviderFactory* seu interesse em ser notificado das mensagens recebidas da RI, destinadas a certo GT no ambiente CORBA. Nesta operação, o usuário de PDU envia o GT de destino e sua própria referência ao objeto para receber as notificações;

1.b O objeto *TcPduProviderFactory* cria um objeto manipulador de PDUs denominado *TcPduProvider* para aquele GT, caso não exista um;

2. O SSP convencional envia, em algum momento, uma PDU BEGIN, contendo os endereços de origem e destino, a invocação da operação e os identificadores de diálogo d_id e de invocação invoke_id;

3. O *TcPduProvider* invoca uma operação BEGIN equivalente em IDL begin_ind na interface do *TcPduUser*, passando a invocação de operação e os parâmetros recebidos junto com o BEGIN;

4. O *TcPduUser* poderá usar a interface *GwAdmin* para obter a referência ao objeto localizador de fábricas *TcUserFactoryFinder* no qual ele invoca uma operação

resolve para obter as referências às fábricas da interface *TcUserGenericFactory*. Para executar a resolução de nomes, o *TcUserFactoryFinder* usa o Serviço de Nomes;

5a. A referência ao objeto iniciador do diálogo, que retornou ao *TcPduUser*, é usada pela *SSPFactory* para criar o objeto *SSPProxy* apropriado no gateway**;**

5b. A referência ao objeto respondedor do diálogo, que retornou ao Usuário de PDU**,** é usada pela *SCPFactory* para criar um objeto SCP no ambiente CORBA;

5c. A operação em IDL é obtida no *TcRepository*;

6. O *SSPProxy* inicia o diálogo com o *SCP*;

7. O *SSPProxy* retorna o resultado ao *TcPduProvider*, que por sua vez gera uma PDU CONTINUE;

8. A PDU CONTINUE é enviada ao SSP da RI convencional.

As interações continuam até que a associação chegue ao fim ou se desfaça por alguma situação anormal.

O diagrama de Estados da Figura 3.6 ilustra a seqüência de mensagens trocadas no cenário da Figura 3.5, onde um PAS tradicional inicia um diálogo com um PCS-CORBA.

1 : register (dest GT, ownObjRef) – operação de registro com parâmetro GT e a própria referência ao objeto;

2 : create-tc-provider – operação de criação do TcPduProvider;

3 : BEGIN PDU[origGT, destGT, did, invoke (invoke id, oper1, param)] – PDU BEGIN, com parâmetros GT de origem, GT de destino, identificador de diálogo e a invocação da operação (com identificador da invocação, a operação 1 e outros parâmetros);

4a: begin-ind (origGT, destGT, did, y) – operação begin gerada, indicando a recepção da PDU para o objeto CORBA que vai iniciar o diálogo;

4b: Solicita à GW Admin a referencia a um objeto TcUserFactoryFinder;

4c: retorna a referencia ao objeto;

4d: invoke-ind (did, ivk-id, …) – operação invoke gerada, indicando a recepção de um componente de invocação, contém entre seus parâmetros o identificador de diálogo e de invocação;

5: resolve (destGT) – operação revolve que provoca o retorno de uma referência para um objeto fábrica, e os mesmos das interfaces associadas aos objetos que serão criados;

6: [AeFactory, resp-iface, init-iface] – parâmetros de retorno: Nome da Fábrica, interface respondedora, interface iniciadora;

7: create-ae[init-iface] – operação para instanciação do objeto iniciador do diálogo;

8: PAS proxy é criado;

9: creat-ae[resp-iface] – operação para instanciação do objeto respondedor do diálogo;

10: PCS é criado;

11: get-operation-name (::resp-iface,opcode) – operação para localizar o nome da operação IDL, dados a interface respondedora e o código da operação;

12: ::resp-iface-name::opname – nome da operação retornada;

13: IDL operation for opname – operação IDL enviada;

14: IDL operation-res – resultado da operação retornado;

15: result-req[did, RESULT-L] – operação que transporta o resultado da operação e corresponde à primitiva TCAP RESULT-L.

3.4 Inclusão de Serviços Especiais de ORB

Nos ambientes orientados a métodos de invocação, como no CORBA, as operações podem ser bidirecionais ou ocorrer numa direção única. Nas invocações em direção única, não há resposta de retorno, enquanto que para operações bidirecionais têm-se usado tipicamente modelos de invocação síncrona ou síncrona deferida. Entretanto, tais modelos não são adequados para as aplicações em tempo real, como ocorre na RI. As operações síncronas têm problemas de escalabilidade porque para cada invocação do tipo consulta/resposta pendente há sempre um cliente encadeado. O modelo síncrono deferido, por outro lado, é ineficiente para aplicações em tempo real e difícil de programar, porque usa a interface de invocação dinâmica, que aloca cópia de dados e memória excessivamente [Soares, 2003].

Para os sistemas em tempo real, os quais impõem altos parâmetros de QoS, o OMG especificou o Serviço de Mensagens que tem como recurso chave um modelo especial de invocação, conhecido como AMI [Arulanthu, 2000].

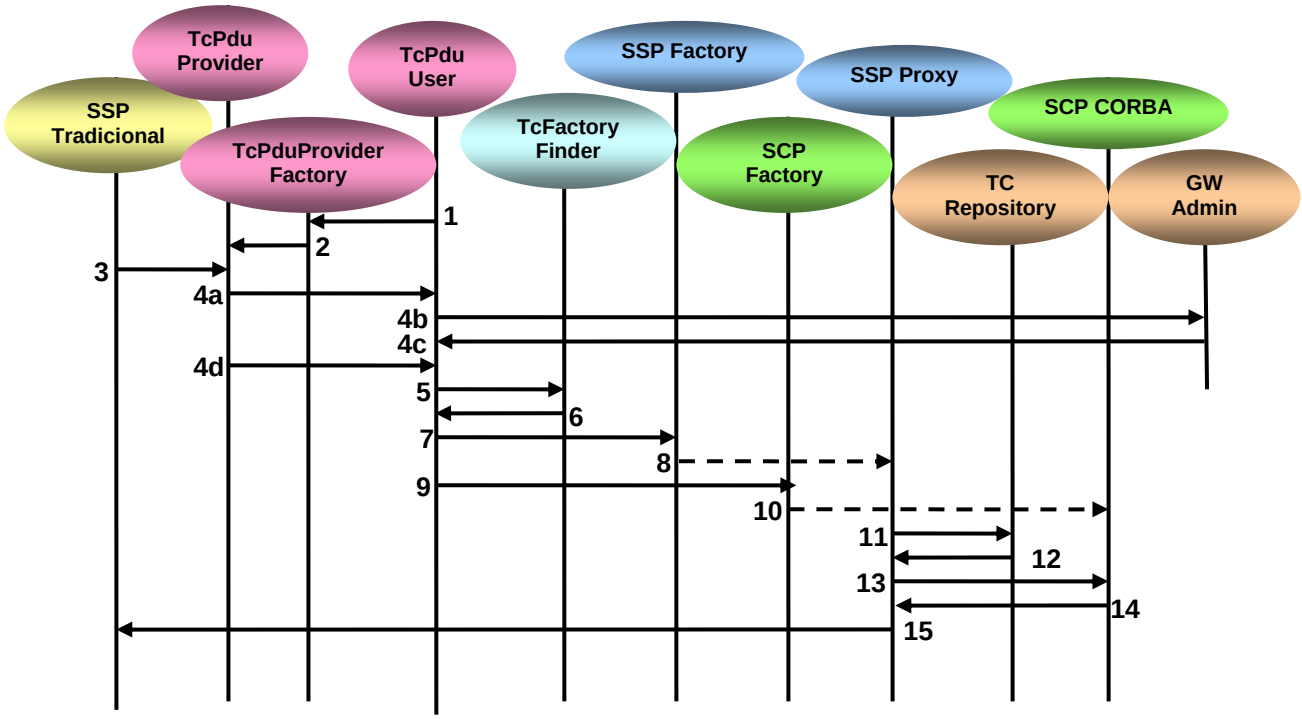

Figura 3.6. Diálogo iniciado pelo PAS convencional

O mecanismo de invocação de mensagens AMI só atua em aplicações do cliente, este procedimento não tem efeito sobre o servidor intermediário (GWS), que fica bloqueado para cada associação em que se envolve, enquanto o servidor final (PCS CORBA) não terminar de processar a operação.

Um novo modelo de concorrência denominado AMH foi criado para prover escalabilidade e melhorar a velocidade de processamento dos dados (*vazão*) do sistema como um todo [Deshpande, Schmidt, O'Ryan, Brunsch, 2002], pois se trata de um método de manipulação de mensagens em que os servidores CORBA processam assincronamente as requisições dos clientes.

Este método AMH foi adaptado para ser usado nas chamadas do GWS para um PAS distribuído (um cliente CORBA), num cenário de aplicação em que todos os elementos da RI estariam dentro da plataforma CORBA ou em situações em que os usuários da NGN desejassem acessar os serviços RI tradicionais [Soares, 2003]. Mas, este não é o contexto de uso do GW aplicado neste trabalho.

Cabe, entretanto, verificar se este método de respostas assíncronas serve para o cenário de utilização do GWS proposto aqui e se é possível adaptá-lo à arquitetura conforme foi

executada. Além disso, este método foi criado especificamente para ser embutido no orb TAO, escrito em C++. Pelo menos, em teoria, nada impede que ORBs diferentes sejam interligados para fornecer um determinado serviço especial, seja ele padrão ou não.

No entanto, supondo que um recurso como o AMH não esteja acessível, ou até mesmo nos casos em que ele possa ser ativado, há ainda a possibilidade de adicionar o Serviço de Balanceamento de Cargas padrão de ORB no GWS para aumentar a escalabilidade dos objetos dos servidores de aplicação (PCSs distribuídos) e melhorar a vazão nas mensagens trocadas entre o GWS e esses servidores.

Portanto, para melhorar a escalabilidade e a vazão no modelo do GWS existem recursos próprios do ORB, tais como o AMI, o AMH e o serviço de balanceamento de cargas, que podem ser ativados, evitando-se assim soluções externas que, em geral, precisam ser atualizadas todas as vezes que ocorrem modificações de projeto.

Por essa razão, preferiu-se neste trabalho seguir esta linha de usar os serviços do ORB que for instalado na própria arquitetura do GWS, (ou seja, sem criar mecanismos particulares), supondo, em princípio, que eles respondam da mesma forma em qualquer ORB comercial. Sem querer antecipar resultados, por constatação, pode-se afirmar com segurança que tal não acontece ainda. Da mesma forma que os serviços da RI não seguem um padrão como deveriam, os serviços especiais dos ORBs não são executados fielmente ao padrão do OMG.

Outra razão para seguir esta opção se deve ao fato de já ter encontrado um número razoável de trabalhos isolados envolvendo aplicações do GWS, que poderiam ser agregados num único caso de uso, como o que é proposto aqui [Malini, 2004], [Gobbi, 2005], [Oliveira, 2004] e [Soares, 2003].

3.4.1 Por que inserir no modelo o serviço de mensagens?

A forma de comunicação tradicional de clientes e servidores por meio de ORBs é síncrona (SMI - *Synchronous Method Invocation*), tanto no pedido do serviço quanto no retorno, como mostrado na Figura 3.7. Assim, a fonte do cliente invoca uma operação no servidor intermediário (1), que processa a requisição e invoca a operação no servidor final (2).

O servidor final processa a operação, retorna os dados ao servidor intermediário (3), que por sua vez endereça os dados ao cliente para processamento (4) (5).

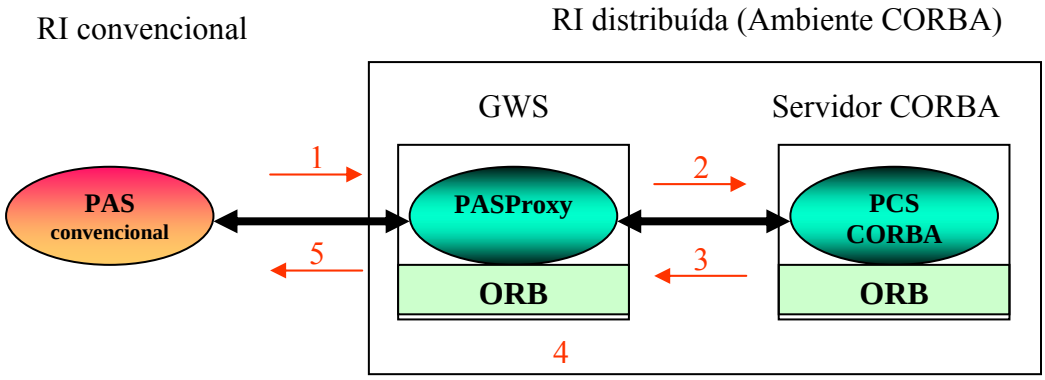

Figura 3.7. O Modelo SMI.

Essa forma de comunicação, contudo, é ineficiente quanto se trata de serviços para a rede TDM, porque não pode haver tempos de espera longos e nem demora de execução. Para esses casos o OMG padronizou o Serviço de Mensagens, usado nas requisições do cliente. Esse recurso possibilita que as mensagens enviadas do GWS para um servidor (representando um PCS distribuído) sejam tratadas de forma assíncrona.

O Serviço de Mensagens usa o AMI, possibilitando que os clientes continuem seus processamentos enquanto aguardam os retornos das invocações. Com essa técnica, chamadas de longas execuções não interferem no encaminhamento de outras.

O OMG desenvolveu duas versões do modelo AMI, o modelo *polling* e o *callback*, ilustrados nas Figuras 3.8 e 3.9 respectivamente, retiradas de [Soares, 2003].

O modelo *polling* se caracteriza por um objeto denominado *poller* que o cliente usa para verificar o estado da requisição, ou seja, se a resposta retornou ou não. A Figura 3.8 ilustra o modelo. O cliente invoca a operação no servidor como mostrado nos passos (1) e (2). A resposta retorna imediatamente (3). Algum tempo depois, ele recupera com o objeto *poller* o resultado enviado (4).

Neste modelo, cada operação AMI retorna um *poller*, que contém os dados e métodos. As operações em um *poller* são as chamadas aos métodos, em vez de invocação de operação. O cliente pode usar esses métodos para checar o estado da requisição, tal que ele possa obter a réplica do servidor. Se a réplica não tiver chegado, o cliente pode parar aguardando sua

chegada ou retornar ao encadeamento de chamadas imediatamente e checar o *poller* quando for conveniente.

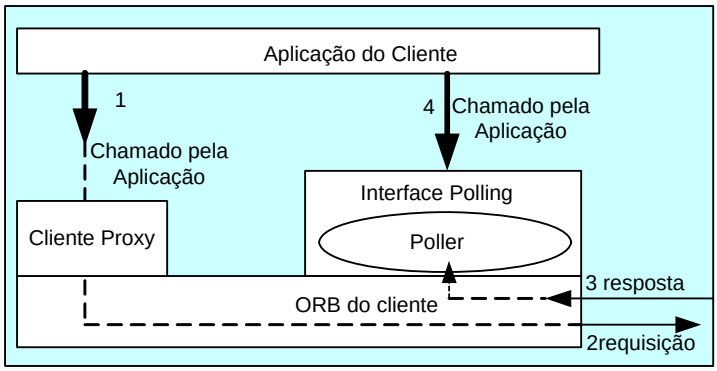

Figura 3.8. O Modelo Polling de CORBA para o AMI.

No modelo callback, como ilustrado na Figura 3.9, quando um cliente invoca uma operação assíncrona em um objeto (1) (2), ele passa uma referência como parâmetro para um objeto local denominado *reply handler object*. Quando a resposta retorna (3), o ORB chama este objeto (4) para notificar à aplicação que a resposta está disponível.

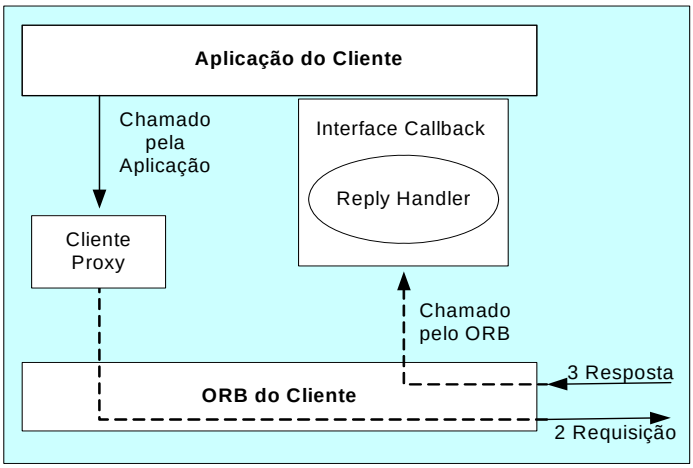

Figura 3.9. O Modelo Callback de CORBA para AMI.

Neste modelo, a referência ao objeto *reply handler* não é passada ao servidor, mas é armazenada no ORB do cliente. Quando a réplica do servidor chega, o ORB do cliente a despacha por uma operação callback apropriada no *reply handler* provido pela aplicação cliente.

Através dessas estratégias um cliente pode fazer várias requisições em objetos residindo em um ou mais servidores sem a necessidade de bloquear o sincronismo das respostas. Além disso, os clientes podem iniciar longos métodos de invocação

concorrentemente, sem incorrer em sobrecarga que seria associada ao uso de um encadeamento separado para cada requisição.

Nota-se que os modelos AMI não afetam o comportamento do servidor, ele só desacopla o envio de requisições da recepção de respostas. Portanto, um cliente pode fazer requisições assincronamente, mas o servidor só pode manipulá-las sincronamente. Assim, o tempo que seria gasto esperando pelas réplicas, pode ser usado pelos clientes para executar outras tarefas.

De um modo geral, o modelo *callback* é mais eficiente do que o modelo *polling* porque o cliente não precisa invocar as chamadas aos métodos repetidamente em um *poller* para recuperar o resultado da operação [Soares, 2003]. Comparando os modelos AMI com os modelos de invocação tradicionais, eles têm três vantagens principais:

1º - Usam a interface de invocação estática do ORB, simplificando o processo de programação comparativamente ao uso da interface de invocação dinâmica empregada no método de invocação síncrono deferido;

2º - Reduz o tempo que o cliente gastaria nas aplicações em tempo real aguardando os resultados;

3º - Provê escalabilidade às aplicações porque reduz o número de clientes que precisariam ser encadeados às operações pendentes, caso fosse usado o método de invocação síncrono.

3.4.2 Por que gerar Respostas Assíncronas?

O método padrão de retorno de invocações dos servidores CORBA é síncrono. Entretanto, pode-se incorporar um recurso conhecido como AMH para forçar o retorno de mensagens do servidor para o GWS de forma assíncrona [Deshpande, Schmidt, O'Ryan, Brunsch, 2002]. Esse recurso, se usado, possibilita que os objetos do GWS fiquem liberados para outras tarefas enquanto aguardam a execução de um serviço, melhorando a vazão das informações no ambiente distribuído e, consequentemente, a velocidade de processamento.

O AMH é o mais recente modelo de concorrência desenvolvido para contornar as limitações dos métodos convencionais, para arquiteturas em camada que manipulam grandes

volumes de tráfego e com elevado número de clientes, como é o caso das RIs. Assim como o mecanismo AMI, ele também usa a interface de invocação estática.

O AMH pode ser compreendido como o AMI estendido dos clientes para os servidores. A construção do AMH foi motivada pela baixa escalabilidade dos modelos de concorrência com *threads* (encadeamentos) e para melhorar o problema de *vazão* de algumas aplicações particulares em sistemas distribuídos de múltiplas camadas, como explicado em [Deshpande, Schmidt, O'Ryan, Brunsch, 2002]. Além do mais, os programas com múltiplos *threads* são difíceis de escrever.

O mecanismo AMH pode ser usado tanto para abrandar as limitações de escalabilidade dos sistemas em camadas como aumentar a vazão. Seus benefícios são similares ao do AMI, ou seja:

- Os servidores intermediários podem processar novas requisições sem ter que esperar pelas respostas dos servidores finais. A escalabilidade é melhorada neste caso, pelo fato de que não há necessidade de gerar um número muito grande de *threads*, se for o caso;
- Os servidores não têm que necessariamente manipular as requisições na ordem de chegada, independente do modelo de concorrência que estiver sendo usado (*single-threaded* ou *multi-threaded*);
- O AMH pode ser usado para projetar e codificar servidores *single-threaded* mais escaláveis do que os *multi-threaded*.

O AMH foi desenvolvido com o intuito particular de resolver problemas de baixa escalabilidade encontrada em determinadas arquiteturas cujas mensagens dos clientes passavam necessariamente por um servidor intermediário que as encaminhava para o servidor final adequado para a execução do serviço em questão.

O AMH foi adaptado também para o cenário de uso do GWS na RI distribuída [Soares, 2003], como ilustra a Figura 3.10, tendo em vista que esta arquitetura tem semelhanças com a arquitetura particular para o qual ele foi criado.

É importante salientar que o AMH é atrelado ao Serviço de Mensagens (AMI) no TAO, ORB exclusivo em que este método reside.

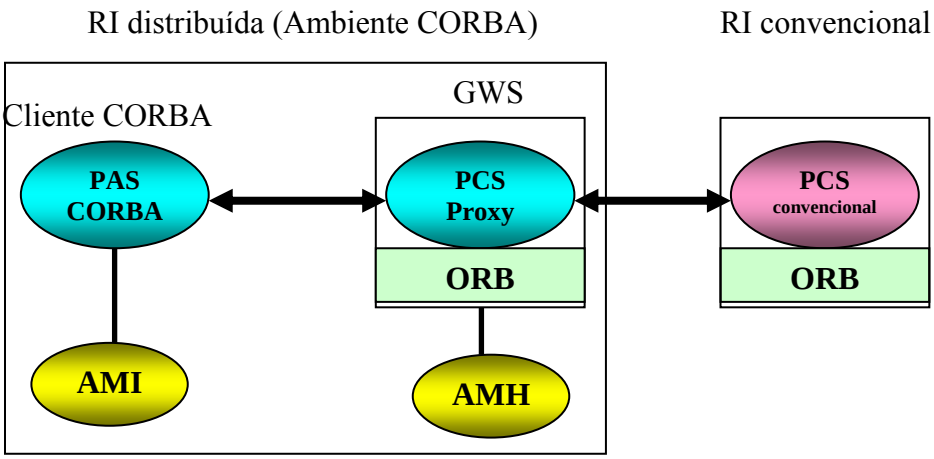

Figura 3.10. PAS CORBA acessa PCS na RI convencional

O AMH possibilita que o programa de um sistema em tempo real transfira o controle de um método, de uma parte do programa para outra. O AMH armazena informação suficiente sobre a requisição do cliente em um local seguro e imediatamente devolve o controle ao ORB. Quando o servidor está pronto para retornar a resposta, por exemplo, quando o servidor intermediário recupera as réplicas de um servidor final, ele pode acessar a informação armazenada para enviar a resposta. Visto que o servidor não fica bloqueado esperando para enviar a resposta, ele pode manipular muitas requisições concorrentemente, mesmo dentro de uma única linha de comando (*thread*).

A Figura 3.11 ilustra os passos dados na manipulação de uma requisição por um servidor intermediário habilitado com AMH.

1. O cliente invoca uma operação no servidor intermediário;
2. 3. O servidor usa o AMH para criar um objeto chamado de *ResponseHandler* e armazenar nele a informação sobre a requisição do cliente - Método (rt,in), Create ();
4. O servidor intermediário retorna o controle ao ORB imediatamente;
5. O servidor intermediário processa a requisição e invoca uma operação AMI no servidor final (e retorna o controle ao ORB);
6. Quando a resposta do servidor final fica disponível, ele a despacha para o *ReplyHandler*;
7. O *ReplyHandler* usa o contexto da informação armazenada no *ResponseHandler* para especificar os parâmetros de saída “out” – Método (out);

8. O *ResponseHandler* retorna a resposta ao cliente.

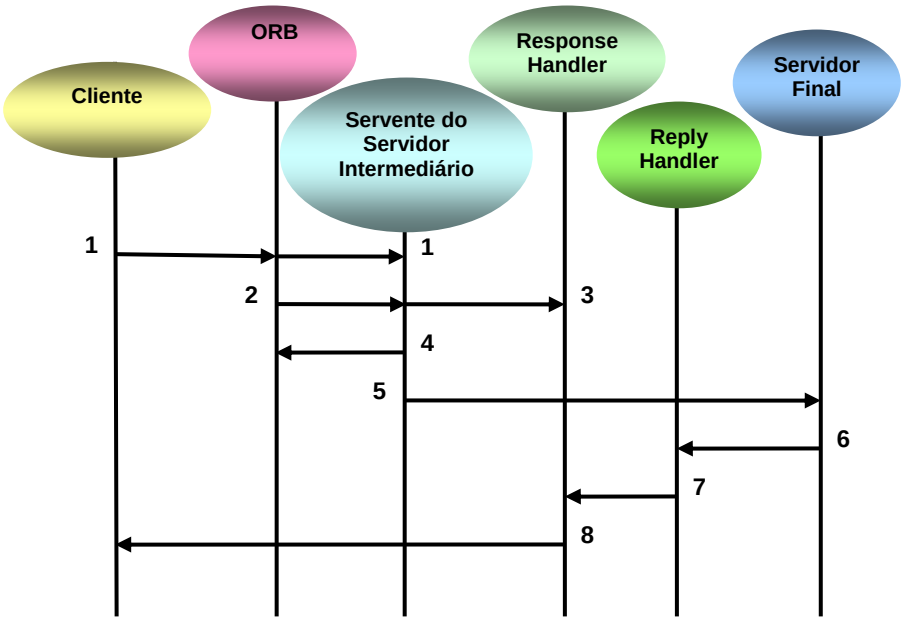

Figura 3.11. O Modelo do AMH.

Visto que o servidor intermediário não precisa ficar bloqueado aguardando a recepção de uma resposta, ele pode manipular muitas requisições concorrentemente usando apenas um encadeamento simples de controle [Soares, Duraes, 2008-2] .

O AMH não é ainda um serviço padrão de ORB, portanto não está presente em todos os ORBs existentes. Até o momento, ele é um serviço proprietário do ORB TAO, um ORB que utiliza a linguagem C++, como já foi dito. Ele foi testado em [Soares, 2003], para um exemplo de comunicação entre cliente e servidor, pensando exatamente em utilizá-lo como estratégia de implantação de uma RI totalmente distribuída. A adaptação do método AMH naquele exemplo foi bastante complexa. Mas é um recurso que possibilita a criação de ambientes distribuídos baseados em CORBA nas redes de telecomunicações, em particular como à tradicional RI, pois permite que grande quantidade de tráfego seja processada proveniente de inúmeros clientes compatíveis com o CORBA.

Para que se possa implementar o AMH no servidor intermediário (GWS) é necessário que o cliente utilize o AMI em suas requisições feitas a esse servidor. O AMH só funciona em conjunto com o AMI. Ele também não foi testado em servidores finais, como seria o caso do PCS distribuído, no cenário em estudo neste trabalho.

3.4.3 Por que usar o Serviço de Balanceamento de Cargas?

O processo de crescimento de aplicativos e de recursos para a implementação de sistemas distribuídos vem estimulando estudos voltados para o balanceamento de carga (BC) nas redes envolvidas com a oferta de serviços para os usuários finais.

Os benefícios que podem ser obtidos com as técnicas de BC são a melhoria da escalabilidade e da vazão dos sistemas [Othman, O'ryan, Schmidt, 2001-1]. A confiabilidade das comunicações também pode ser melhorada se o sistema se adaptar dinamicamente às mudanças de configuração que surgirem, ou às falhas de software. Tais técnicas também possibilitarão o uso mais eficiente dos recursos, permitindo que novos servidores sejam inseridos somente quando necessário.

Os mecanismos de BC tentam distribuir a carga de trabalho dos clientes eqüitativamente entre servidores, e podem ser aplicados em três níveis: de Rede, de Sistema Operacional e de *Middleware* [Othman, O'ryan, Schmidt, 2001-1]. Nesse trabalho será abordado o BC no nível de *Middleware,* nos ORBs CORBA.

O ORB pode determinar qual cliente será encaminhado para determinada réplica de objeto em um determinado servidor. Uma réplica é definida como uma instância duplicada de um objeto em particular, em um servidor gerenciado por um BC.

O BC executado na camada *Middleware* pode ser realizado por-sessão, por-requisição ou por-demanda. A carga de trabalho do cliente é distribuída eqüitativamente sobre os vários servidores na rede, a fim de melhorar os tempos de resposta às requisições dos clientes.

No tipo de ligação por-sessão, as requisições de clientes continuarão sendo emitidas à mesma réplica durante uma sessão. No tipo de ligação por-requisição, cada requisição de cliente será remetida a uma réplica potencialmente diferente. No tipo de ligação por-demanda, as requisições dos clientes podem ser ligadas à outra réplica sempre que o balanceador de carga julgar necessário. Este método força um cliente a emitir sua próxima requisição a uma réplica diferente da qual ele está enviando sua requisição no momento [Bodart, Durães, Soares, Garcia, 2004].

Ao projetar um serviço de BC, é importante selecionar um algoritmo apropriado que decidirá qual réplica processará cada requisição de entrada. Em geral, podem-se classificar as políticas de BC em duas categorias:

- Não-adaptativa: pode usar políticas não adaptativas, como um algoritmo simples de *round-robin* ou um algoritmo de randomização (*random*) [Othman, O'ryan, Schmidt, 2001-1], para selecionar qual réplica controlará uma requisição em particular.
    - *Round-Robin:* o localizador usa esse algoritmo para selecionar os clientes e servidores ativos, sendo o primeiro cliente encaminhado para o primeiro servidor, o segundo cliente encaminhado para o segundo servidor, e assim sucessivamente.
    - *Random:* o localizador seleciona randomicamente um servidor ativo para manipular o cliente.
- Adaptativa: pode usar políticas adaptativas que utilizam informações em tempo real para selecionar a réplica que controlará uma requisição particular (algoritmo *active load balancing).*
    - *Active Load Balancing:* a carga dos objetos é monitorada de forma que serão selecionados para atender a uma requisição aqueles com a menor carga naquele momento.

Combinando os níveis, formas e categorias descritas anteriormente, pode-se criar arquiteturas alternativas de BC. A Figura 3.12 ilustra três arquiteturas que foram exploradas em [Oliveira, 2004].

Na arquitetura de BC por-sessão não-adaptativa (Fig. 3.12a), um balanceador de carga é projetado no CORBA de forma a selecionar a réplica designada quando uma sessão de cliente-servidor é estabelecida pela primeira vez, ou seja, quando um cliente obtém uma referência para um objeto CORBA (chamado aqui de réplica) e se conecta àquele objeto.

Diferentes clientes podem ser direcionados a diferentes réplicas de objeto usando um recurso de ativação, como um objeto Repositório de Implementação do CORBA, ou um Serviço de Localização, tal como o Serviço de Nomes.

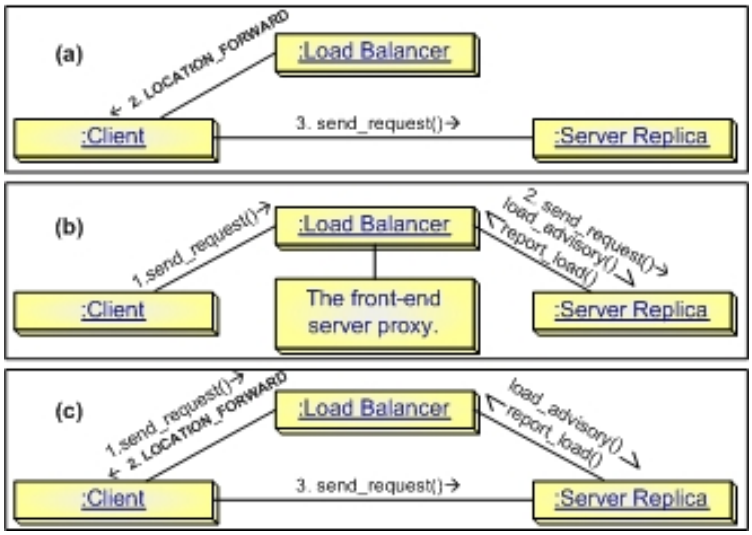

Figura 3.12. Arquiteturas para BC

Essa arquitetura de balanceamento pode satisfazer as necessidades de transparência das aplicações, aumentando a segurança do sistema e a interoperabilidade do CORBA. Porém, arquiteturas de BC por-sessão não-adaptativas não são adequados quando se necessita que a manipulação dinâmica das requisições de operação dos clientes seja feita adaptativamente.

Na arquitetura de BC por-requisição adaptativa (Fig. 3.12b), existe um servidor que é um *proxy*, o qual recebe todas as requisições dos clientes. Neste caso, este servidor contém o balanceador de carga. Ele seleciona uma réplica no servidor final de acordo com sua política de BC e envia a requisição para aquela réplica. O servidor *proxy* espera a resposta da réplica chegar, e a retorna para o cliente. Mensagens de consulta de carga são enviadas do balanceador para as réplicas, fazendo com que elas aceitem as requisições ou as redirecionem para o balanceador de carga.

O principal benefício do uso dessa arquitetura é o aumento da escalabilidade e a equidade. Por exemplo, o servidor intermediário pode examinar a carga em cada réplica antes de selecionar a réplica para cada requisição, distribuindo a carga eqüitativamente. Portanto, ela é indicada para balanceamentos adaptativos. No entanto, ela introduz *overhead* devido à inclusão de mensagens de consulta [Othman, O'ryan, Schmidt, 2001-2].

Outro problema é que em certas aplicações, como é o caso do exemplo apresentado neste trabalho é importante que o servidor intermediário (GWS) possa executar outras tarefas, enquanto aguarda a resposta do servidor final, o que não é possível com essa arquitetura.

Na arquitetura de BC por-demanda adaptativa (Fig. 3.12c), o cliente recebe inicialmente uma referência para o objeto do balanceador de carga. Usando um mecanismo chamado *LOCATION_FOWARD*, o balanceador pode direcionar a requisição inicial do cliente para uma determinada réplica. Os clientes continuarão usando a nova referência para objeto, que é obtida como parte da mensagem *LOCATION_FORWARD,* para se comunicarem com esta réplica diretamente até que eles recebam outros redirecionamentos ou terminem suas conversações.

Essa arquitetura pode monitorar a carga da réplica continuamente, diferentemente das arquiteturas não-adaptativas. Usando a informação de carga, um balanceador poderá determinar como distribuí-la de maneira equilibrada, redirecionando o cliente para a réplica menos sobrecarregada. A informação sobre o estado da carga (nominal ou sobrecarga) pode ser definida pelo usuário.

Dependendo do projeto podem surgir problemas, devido ao fato de que as réplicas no servidor precisam ser preparadas para receber mensagens de um balanceador de carga, e que os clientes precisam ser redirecionados para o balanceador.

Embora essas mudanças sejam necessárias e não afetem a lógica da aplicação, os desenvolvedores de aplicativos precisarão modificar os componentes de inicialização e ativação dos servidores CORBA para responder às mensagens de consulta mencionadas anteriormente.

Entretanto, para aplicações em que o servidor intermediário precise dar vazão, simultaneamente, para muitas requisições, independente do recebimento dos resultados das requisições, provenientes do processamento nos servidores finais, essa arquitetura parece ser a mais adequada.

Esse recurso foi estudado em [Oliveira, 2004] e testado em um estudo de caso programado sobre um ORB TAO.

O recurso de BC, quando não disponibilizado no ORB nativo, pode ser utilizado encadeando um ORB adequado para esse fim. Essa é uma das facilidades propostas pela tecnologia CORBA [Othman, O'ryan, Schmidt, 2001-2].

A implementação de uma estratégia de BC, para proporcionar a manipulação de grandes volumes de tráfego pelos servidores da arquitetura de redes, em conjunto com os métodos AMI e AMH visa registrar altos níveis de desempenho para os aplicativos das RIs.

3.5 Conclusão

A tendência atual de transformação dos sistemas de telecomunicações em grandes ambientes distribuídos tem revelado a necessidade premente de adaptação das técnicas de implementação desses sistemas às aplicações em tempo real as quais exigem alto desempenho (altos índices de escalabilidade e vazão), com alta confiabilidade (segurança).

Para o cenário em que o PAS está na RI convencional e o PCS no ambiente CORBA, pretende-se introduzir o serviço de BC adaptativo sob demanda no GWS (para a geração dos PASs Proxy) e no Servidor (para a geração dos PCSs CORBA). Conforme ilustra a Figura 3.13, extraída de [Bodart, Durães, Soares, Garcia, 2004], as réplicas serão geradas de acordo com a demanda de tráfego.

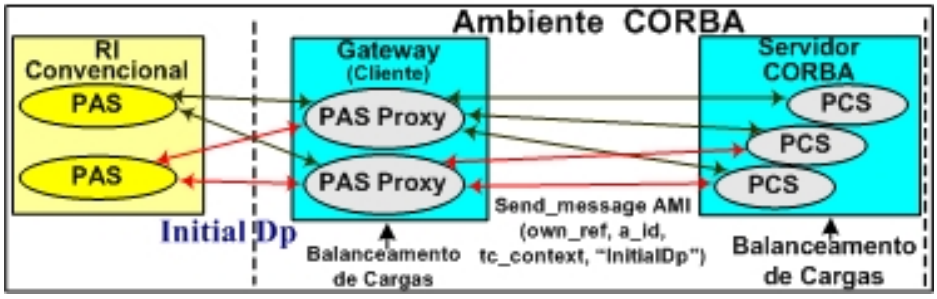

Figura 3.13. Inclusão do Serviço BC + AMI

Quando o tráfego entre clientes PASs e um PASProxy se torna intenso, é necessário que sejam criadas réplicas do PASProxy no GWS, para garantir que o sistema não fique sobrecarregado. A implementação do serviço AMI permite que um PASProxy mantenha vários diálogos provenientes de diferentes usuários da RI convencional com um mesmo objeto PCS CORBA. O serviço de BC neste caso, vem aumentar a escalabilidade e vazão, conseguidas com o serviço AMI, através da geração de outros PCSs CORBA para se

comunicarem com aquele PASProxy, ou criar outros PASs Proxy para fluir o tráfego em direção a um determinado endereço no servidor.

Para o cenário em que o PAS está no ambiente CORBA e o PCS na RI convencional, a implementação do serviço de BC no GWS, para a geração econômica de PCSs Proxy, proporcionaria a otimização da escalabilidade e vazão nas RIs, principalmente se fosse implantado em conjunto com os serviços de mensagens AMI e AMH. Porém esse cenário só seria possível de se implementar no ORB TAO, que é o único até o momento que possui o serviço AMH. Esta associação é ilustrada na Figura 3.14, extraída de [Bodart, Durães, Soares, Garcia, 2004]. Além do mais, esse cenário está fora do escopo deste trabalho, porque a plataforma CORBA para Java não comporta o AMH.

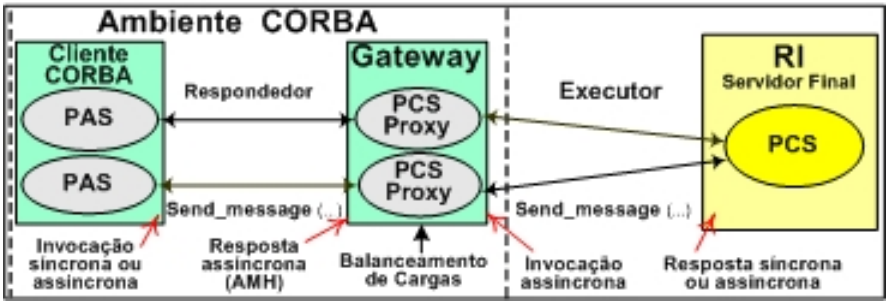

Figura 3.14. Inclusão do Serviço BC + AMI + AMH

4. Implementações e Resultados

4.1 Introdução

Na Arquitetura de Serviços da NGN, a possibilidade de substituir a execução tradicional dos serviços feita por meio de sistemas centralizados, por Servidores CORBA especializados, altamente escaláveis, motivou a migração das RI para uma plataforma distribuída.

A idéia principal no projeto da camada distribuída sobre a RI convencional foi de aproveitar diferentes recursos incorporados aos ORBs CORBA para proporcionar não só a escalabilidade e uma maior velocidade de processamento dos serviços executados, mas também garantir a vazão eficiente do tráfego de pacotes na plataforma distribuída. A base para a construção do software foi o modelo de GWS do OMG e a incorporação de serviços especiais, tais como o Serviço de Mensagens, o Serviço de Balanceamento de Carga e o recurso AMH do TAO. Os dois primeiros serviços são considerados padrões, já o AMH é um método de respostas assíncronas que foi incorporado no orb TAO, agregado ao serviço de mensagens (AMI), além de ser exclusivo do TAO.

Este capítulo trata da construção do software a ser incorporado na camada distribuída da RI, que inclui a programação de parte do GWS e da comunicação deste com os objetos inseridos em um servidor de aplicações responsáveis pelo retorno dos serviços executados.

São explicadas também aqui as dificuldades encontradas na produção do código completo devido, principalmente às limitações nos ORBs testados, bem como todas as adaptações necessárias para fazer o código funcionar sobre cada um dos ORBs testados. São explicados quais os recursos adicionais disponíveis ou não nesses ORBs para atingir os objetivos de adicionar QoS à arquitetura do GWS.

4.2 O Ambiente CORBA

Em CORBA [CORBA 2.4 Specifications], uma aplicação consiste de objetos interagindo via mecanismos de comunicação do ORB. Para cada interação entre dois objetos, um atua como cliente e o outro como servidor. Como cliente, tem acesso a uma referência de objeto e invoca algumas operações definidas na interface do servidor. Como servidor, é representado pela implementação do objeto recebendo a operação com os parâmetros, executando-a e devolvendo o resultado ao cliente. A referência ao objeto é um valor que o identifica.

A invocação de uma operação feita por um cliente a uma implementação de objeto no servidor envolve vários mecanismos, como ilustra a Figura 4.1.

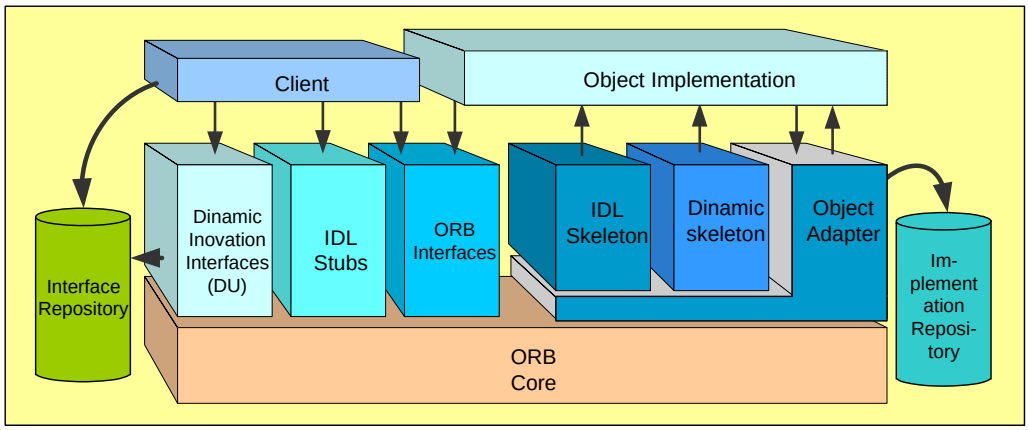

Figura 4.1. A Estrutura do ORB

O ORB é responsável pela interoperabilidade entre aplicações executadas em diferentes máquinas localizadas em ambientes distribuídos heterogêneos, interligando múltiplos sistemas de objetos, gerenciando a transferência de controle e de dados entre os clientes e as implementações de objetos. No caso de uma operação não ser executada com sucesso, gera-se uma exceção que deve ser tratada pelo cliente

Todas as transferências entre o cliente e o servidor são suportadas pela plataforma CORBA e ficam ocultas dos objetos em questão. O ORB também se encarrega de iniciar e ativar um objeto antes de entregar uma requisição a ele.

Algumas das características importantes de CORBA são as transparências de localização, de implementação e de execução dos estados dos objetos.

O componente essencial para prover portabilidade entre diferentes sistemas de objetos é o Adaptador de Objetos (POA – *Portable Objects Adapter*). Cada objeto pode ser implementado em uma linguagem de programação diferente, sobre sistemas operacionais diferentes.

Para desenvolver uma aplicação compatível com CORBA, é preciso descrever seu comportamento em termos de objetos visíveis, através de interfaces. Para toda interface são produzidos um stub para o cliente, e um skeleton para o servidor.

Quando um cliente precisar dos serviços de um objeto ele poderá invocar a operação correspondente no stub. Essa mesma técnica pode ser usada para chamar os serviços de aplicação e os serviços CORBA. Sempre que uma requisição é recebida sobre a rede, o adaptador de objetos invoca a operação correspondente através do skeleton. O adaptador de objetos tem controle completo sobre o servidor, e invoca um método particular dependendo da aplicação.

O desenvolvimento de aplicações CORBA obedece a alguns passos básicos, descritos a seguir:

1. Escrever uma especificação IDL;
2. Compilar essa especificação com o compilador IDL para gerar classes Java (interfaces, classes helper e holder, bem como stubs e skeletons);
3. Escrever uma implementação para a interface gerada no passo 2;
4. Escrever uma classe principal que instancie a implementação do servidor e registre-a no ORB;
5. Escrever uma classe cliente que recupere uma referência para o objeto servidor e faça invocações remotas, isto é, chamadas CORBA.

O CORBA apresenta vantagens como plataforma para sistemas distribuídos. Primeiro, o modelo de objeto e a linguagem IDL representam uma estrutura sólida para o desenvolvimento de sistemas orientados a objetos. Segundo, as especificações CORBA suportam a heterogeneidade de softwares e hardwares. Terceiro, os recursos do repositório de interfaces e os outros mecanismos inseridos possibilitam a criação de aplicações que se ajustam dinamicamente às mudanças em seus ambientes. O CORBA foi incentivado, em parte, à sua compatibilidade com as tecnologias da Internet.

Por outro lado, vários problemas podem ser apontados:

- O paradigma de comunicação *request/reply* é insuficiente para as comunicações assíncronas orientadas a mensagens, como é o caso das aplicações das RIs. Para esses casos foram especificados outros serviços, como o Serviço de Mensagens Assíncronas.
- O CORBA não foi elaborado para suportar implementação de sistemas em tempo real, como é o caso das aplicações de telecomunicações.
- Nem todos os serviços padrões do OMG estão inseridos em diversos ORBs comerciais.

Apesar de vários problemas ainda por resolver, as especificações do grupo OMG têm como destaque a proposta de assegurar a portabilidade, a reutilização e a interoperabilidade dos softwares. Essa meta pode ser alcançada através da padronização das interações dos componentes nos sistemas distribuídos. Um dos principais componentes é o ORB, que encapsula as soluções para os desenvolvedores de aplicação.

4.3 O Software do GWS

Como foi dito no Capítulo 3, a comunicação dos PASs com a parte do GWS que contém as interfaces orientadas a PDU não foi simulada neste trabalho, à qual depende de uma ontologia para a conversão das primitivas pertinentes às diferentes versões do protocolo SS7 para a linguagem IDL (sintaxe e semântica). Essa abordagem representa um esforço de pesquisa e de programação de grande complexidade, motivo pelo qual não foi planejado como parte desta dissertação de mestrado.

Foram criados os objetos necessários à interação com os servidores de aplicação. Ele tem várias responsabilidades, entre elas, a localização das aplicações, a inicialização e a manutenção dos diálogos entre os dois ambientes: Eles são criados a partir das seguintes interfaces, descritas no Capítulo 3:

- Interface *TcUser*;
- Interface *TcUserGenericFactory*,
- Interface *TcFactoryFinder*;
- Interface *GwAdmin*,
- Interface *TcServiceFinder*,

- Interface *TcRepository*.

O diagrama de Estados da Figura 4.2 ilustra a seqüência de mensagens executadas pelo programa gerado, entre os objetos de interação do GWS, considerando que a chamada tenha sido iniciada e um SSP tradicional, mas o diálogo gerado parte do TcUser dentro do GWS para um objeto SCP CORBA, que está no servidor de aplicações.

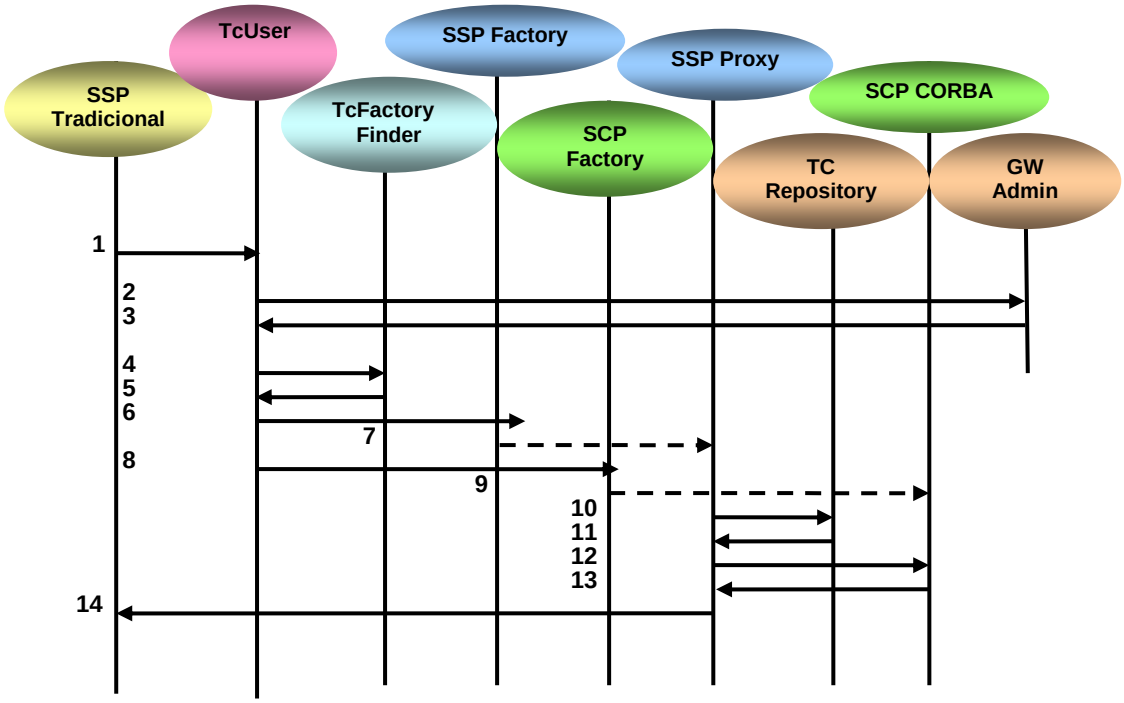

Figura 4.2 Diálogo PAS convencional e PCS CORBA

1. O SSP convencional envia, em algum momento, uma PDU BEGIN, contendo os endereços de origem e destino, a invocação da operação e os identificadores de diálogo d_id e de invocação invoke_id;
2. 3. O *TcUser* poderá usar a interface *GwAdmin* para obter a referência ao objeto localizador de fábricas *TcFactoryFinder* ;
4. 5. O *TcUser* invoca uma operação resolve no *TcFactoryFinder* para obter as referências às fábricas da interface *TcUserGenericFactory*, usando o Serviço de Nomes;
6. A referência ao objeto iniciador do diálogo retornou ao *TcUser;*
7. O objeto *SSPProxy* é criado no gateway;
8. A referência ao objeto respondedor do diálogo retornou TcUser
9. O objeto *SCP* no ambiente CORBA é criado;

10. 11. O *SSPProxy* obtém do *TcRepository* a operação em IDL correspondente àquela proveniente do SSP tradicional que originalmente estava em INAP;
12. 13. O *SSPProxy* inicia o diálogo com o *SCP*;
14. O *SSPProxy* retorna o resultado ao do SSP tradicional.

Toda aplicação CORBA deve contar com um *servidor* e pelo menos um *cliente*. Neste trabalho, foi gerada uma classe *Server*, que faz a criação dos objetos do GWS e a ligação destes no serviço de nomes (quando necessário), e um *Client*, que desempenha o papel do iniciador de diálogos, recuperando uma fábrica a partir do serviço de nomes (via interface *TcFactoryFinder*), criando objetos respondedores e invocando operações sobre eles.

A seguir serão mostrados quais foram os ORBs utilizados na implementação do software do GWS, os serviços que foram incorporados para melhorar o desempenho deles, os testes realizados, os resultados obtidos e as dificuldades encontradas para cada caso.

4.4 Jacorb

O JacORB é um ORB Java disponibilizado como software livre, desenvolvido no Departamento de Ciências da Computação da Freie Universität Berlin (FUB) [JacORB]. Por muito tempo, o JacORB foi a única implementação Java/CORBA disponível gratuitamente, e logo ganhou projeção mundial, inicialmente quase que exclusivamente em projetos acadêmicos, e depois no uso comercial.

Devido ao número limitado de desenvolvedores, a filosofia do projeto nunca foi ter a arquitetura CORBA implementada por completo, mas sim implementar módulos com qualidade e fiéis ao padrão. Portanto, apesar de ainda prover poucos serviços CORBA, o JacORB é um ORB seguro e estável [The JacOrb Team, 2007]. Nesse trabalho foi utilizada a versão 2.3 do JacORB.

O software do GWS foi primeiramente programado nesse ORB, e funcionou. As Figuras 4.3, 4.4 e 4.5 mostram a execução do servidor e do cliente, respectivamente, que foi realizada no Ambiente Integrado de Desenvolvimento (*Integrated Development Environmen t*- IDE) - NetBeans, versão 6.0.1 [NetBeans 6.0.1, 2007].

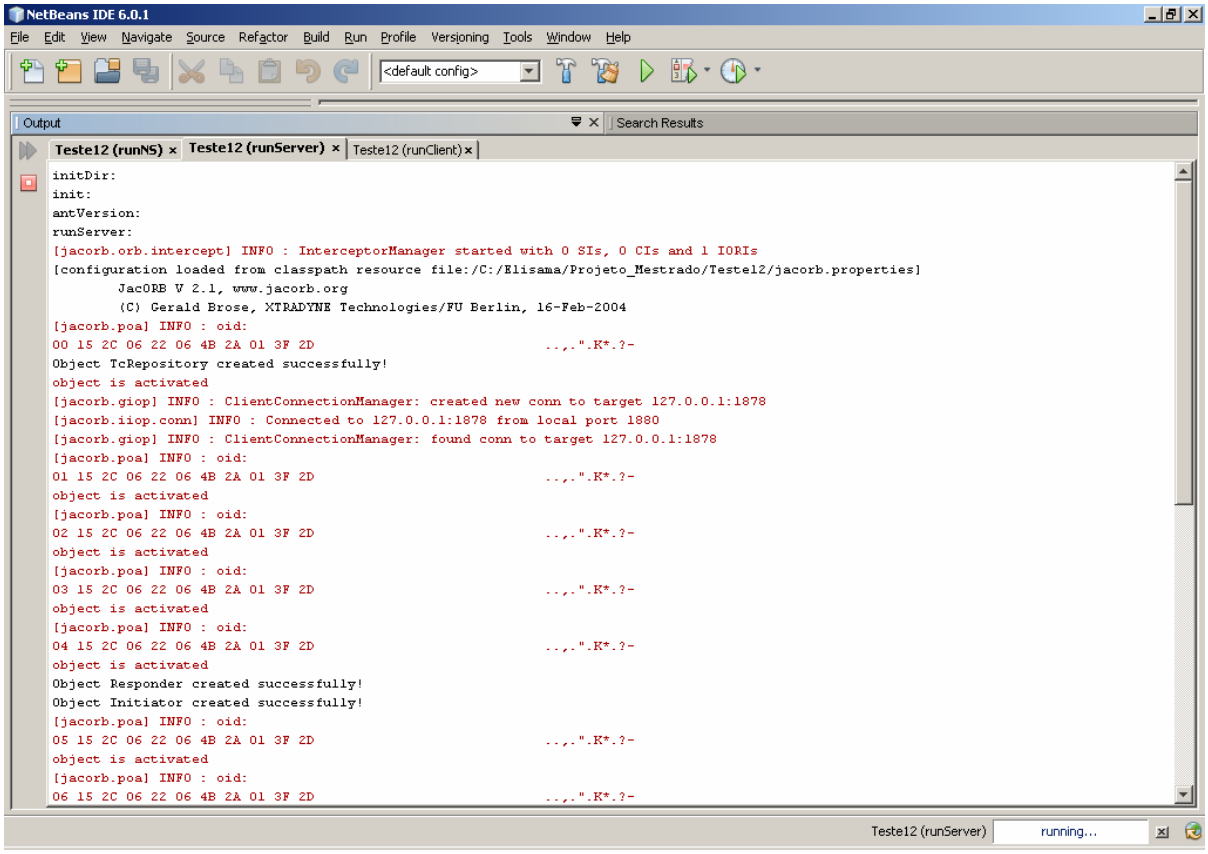

Figura 4.3. Execução do Servidor no JacORB

Quando o servidor é executado, são criados todos os objetos do GWS, e é feita a ligação deles com o serviço de nomes, quando necessário.

O cliente trabalha como iniciador dos diálogos, e quando é executado recupera uma fábrica a partir do serviço de nomes (via interface *TcFactoryFinder*), cria os objetos respondedores e invoca operações sobre eles.

Os códigos em Java do servidor (*Server*) e do cliente (*Client*) estão no Apêndice A.

Com o software do GWS funcionando, o próximo passo foi a inclusão do Serviço de Mensagens Assíncronas (AMI), com o intuito de aumentar a escalabilidade dos objetos servidores, e melhorar a vazão nas mensagens trocadas entre o GWS e esses servidores.

O Jacorb possui o modelo *callback* para o AMI. Nesse modelo, as respostas às requisições são direcionadas para o objeto ReplyHandler, o qual é registrado no lado do cliente quando uma invocação assíncrona é iniciada [The JacOrb Team, 2007].

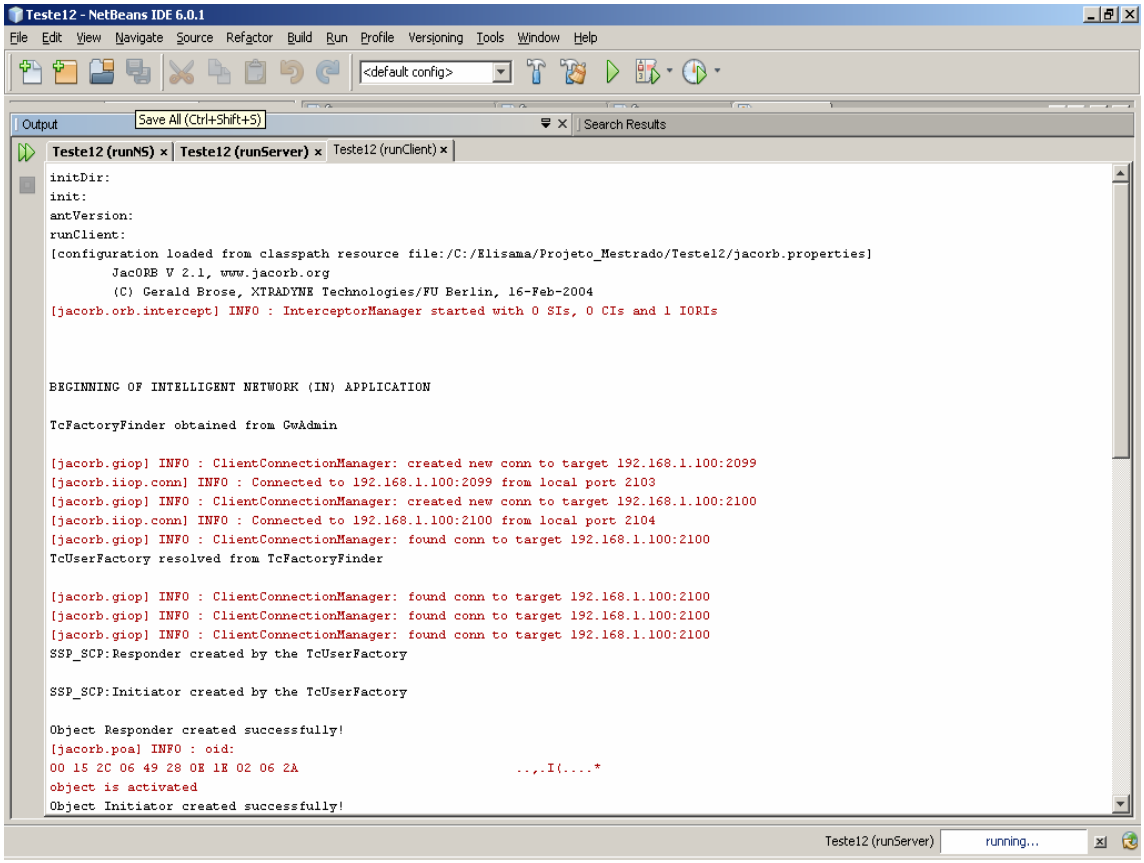

Figura 4.4. Execução do Cliente no JacORB – parte 1

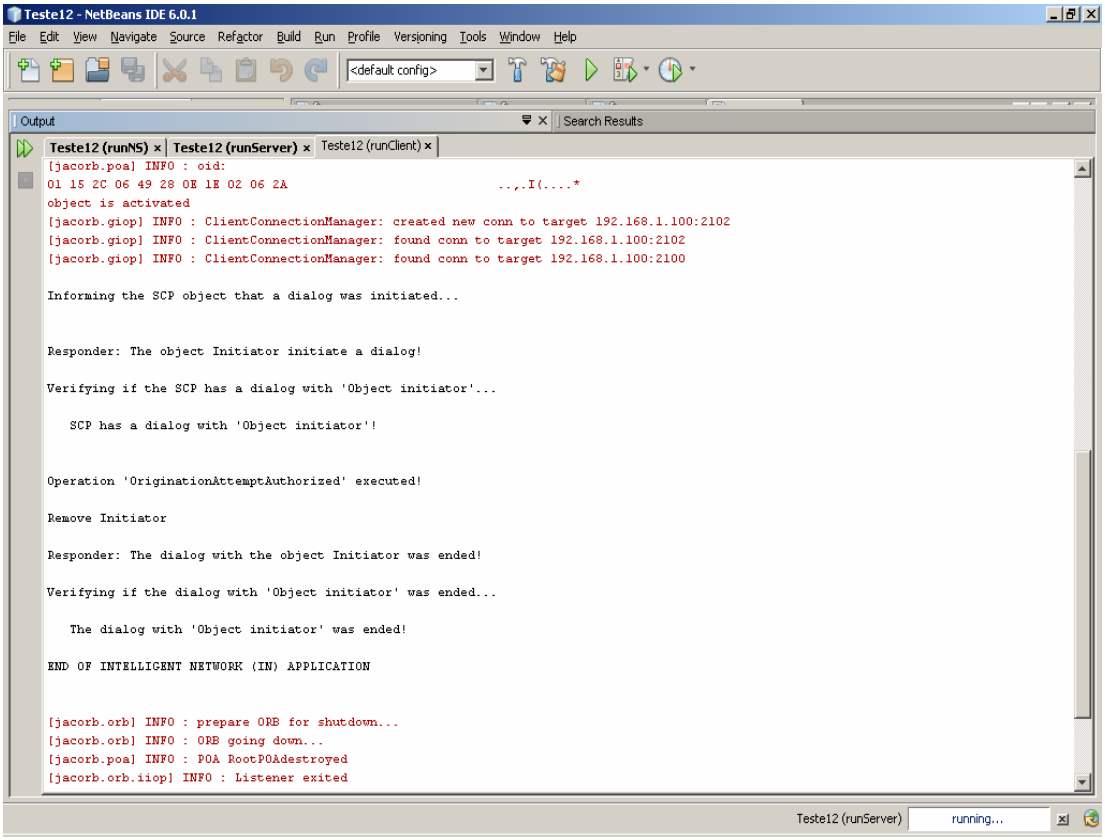

Figura 4.5. Execução do Cliente no JacORB – parte 2

O objeto ReplyHandler é criado quando a definição em IDL da operação que será requisitada no servidor é compilada usando o parâmetro *-ami_callback.* Esse objeto é chamado *AMI_ServerHandler*. Para o GWS, a IDL compilada foi TC_User.idl.

> idl -ami_callback TC_User idl

Para cada operação requisitada no servidor, o objeto *AMI_ServerHandler* tem uma operação com mesmo nome que recebe o valor de retorno e parâmetros de saída da operação original. Há também um método adicional nomedaoperação*_excep*, que é chamado se ocorre erro ou exceção na excussão daquele método (nomedaoperação).

O compilador IDL também gera um método especial *sendc*_nomedaoperação, com os seguintes parâmetros:

1. Uma referência para o objeto *AMI_ServerHandler*;
2. Todos os parâmetros de entrada ou entrada/saída da operação original, modificados para somente entrada (os parâmetros de saída são omitidos desta operação).

Para uma invocação assíncrona, uma instância do objeto *AMI_ServerHandler* precisa ser criada, registrada no ORB e passada para a operação *sendc*, como é mostrado a seguir:

```
ORB orb = ...
Server s = ...
// o objeto AMI_ServerHandler é instanciado e é obtida uma referência para ele:
AMI_ServerHandler h = new AMI_ServerHandlerImpl()._this (orb);
// a operação sendc é invocada:
((_ServerStub)s).sendc_operation (h, 4, 5);
```

A operação *sendc* é definida somente no stub, então é necessário fazer uma operação "cast" para invocá-la especificamente no JacOrb. Essa operação é usada para conversão de tipos de variáveis na linguagem Java.

Como o serviço de mensagens só afeta o cliente, o servidor do JacOrb foi executado corretamente. No entanto, o cliente apresentou falha na execução, apesar do referido serviço ter sido escrito no programa cliente, exatamente como indicado no manual de instruções.

O JacORB foi o ORB utilizado nos trabalhos anteriores que motivaram a realização deste. Uma das razões para essa escolha é sua simplicidade para se programar. Esse foi o motivo também para que nesse trabalho ele tenha sido o primeiro ORB testado com o

software do GWS. Porém ele não possui em sua especificação o serviço de Balanceamento de Carga. Assim, buscou-se um ORB que possibilitasse a utilização desse serviço e do AMI.

Pesquisando sobre esses ORBs, foi encontrado o Orbacus, um ORB que tem uma versão para Java, linguagem utilizada no trabalho, e possui em suas especificações para os serviços AMI e Balanceamento de Carga.

4.5 Orbacus

O Orbacus é um ORB desenvolvido pelo *IONA Technologies* [IONA Technologies], compatível com as especificações CORBA 2.4, e possui uma versão para Java e outra para C++. A versão para C++ pode trabalhar com altas cargas e diz ter excelente desempenho, enquanto a versão para Java enfatiza a portabilidade e a integração total com a plataforma Java 2. Ambas as versões são compatíveis com *Internet Inter-ORB Protocol* (IIOP), que possibilita a integração de diversas plataformas baseadas em componentes de software distribuído.

O Orbacus é distribuído em código fonte, o que permite a identificação de possíveis problemas em uma aplicação ou infra-estrutura, fazer modificações na medida do necessário, de forma a obter o melhor aproveitamento da estrutura e funcionamento interno do ORB.

O Orbacus está sendo utilizado numa grande variedade de aplicações em mais de 2000 usuários finais e vendedores de softwares independentes nas áreas de telecomunicações, *e-commerce,* finanças, de defesa e indústrias de petróleo, entre outras [Orbacus Technical Review, 2007].

Inicialmente foi utilizada a versão 4.3.0 do Orbacus para Java [Orbacus 4.3.0, 2005] na implementação do software do GWS, utilizando os mesmos códigos do servidor (*Server*) e do cliente (*Client*) testado anteriormente no JacORB. As Figuras 4.6 e 4.7 ilustram a execução do servidor e do cliente, respectivamente. Foi utilizado o IDE NetBeans, versão 6.0.1 [NetBeans 6.0.1, 2007] nesses testes.

Com o software do GWS funcionando no ORBACUS, foi incluído nele o método AMI constituinte do Serviço de Mensagens do CORBA, para tratar das chamadas assincronamente.

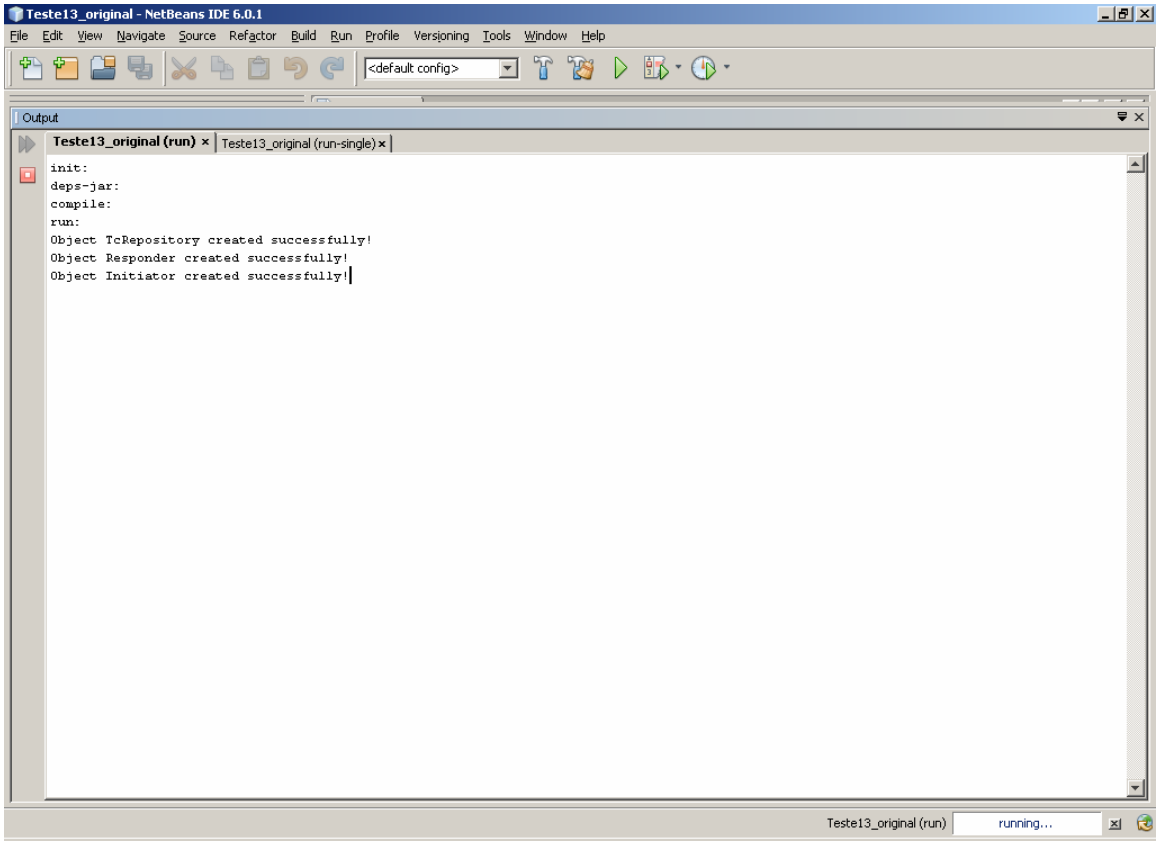

Figura 4.6. Execução do Servidor no Orbacus 4.3.0

Para usar o AMI, foi necessário gerar, a partir da IDL, os objetos que possibilitam o acesso assíncrono do cliente ao servidor. Isso é feito compilando a IDL do objeto que define a operação a ser invocada no servidor, usando o parâmetro *--with-async*. A IDL do GWS compilada foi TC_User.idl.

> jidl TC_User.idl --with-async

Um objeto *AMI_xxHandler* é gerado na compilação da IDL. Em sua implementação o usuário deve instanciar um objeto *callback*, derivado do *AMI_xxHandler*, (o Orbacus possui esse modelo de AMI) e passar para o ORB uma requisição AMI. O ORB pode então usar este objeto *callback* para informar à aplicação cliente que a requisição foi completada [Orbacus 4.3.0, 2005].

O software do servidor não precisa ser modificado para a inclusão do AMI, e no cliente apenas a chamada à operação precisa ser alterada, utilizando a operação assíncrona. O Apêndice B mostra o código em Java do cliente e do servidor com a inclusão do AMI no Orbacus.

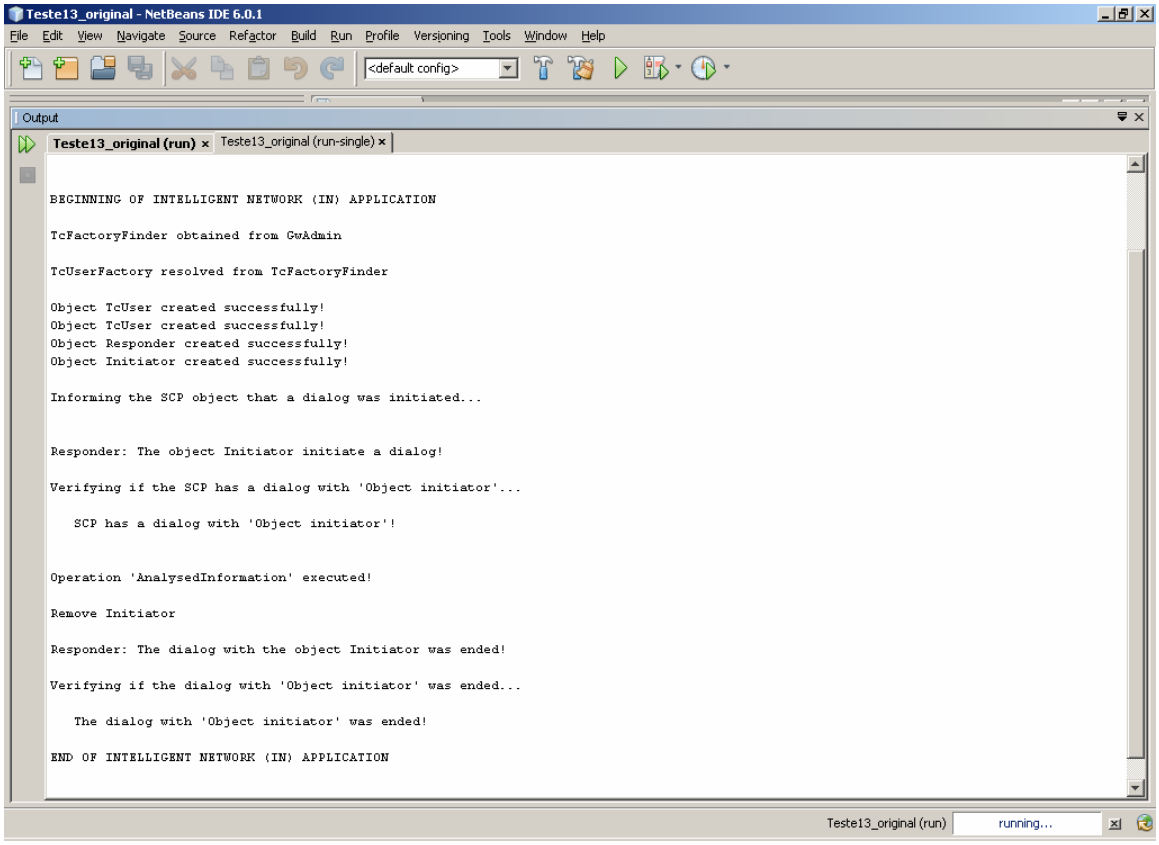

Figura 4.7. Execução do Cliente no Orbacus 4.3.0

As Figuras 4.8 e 4.9 apresentam a execução do servidor e cliente, respectivamente, do software do GWS, com o serviço AMI implementado no cliente.

O próximo passo em direção a uma arquitetura com melhor desempenho para aplicações RI seria a inclusão do BC no software do GWS. Primeiramente pensou-se em testar somente o BC no GWS, e em seguida acrescentar o AMI, que já havia funcionado.

Um grupo de objetos é criado, e ele é responsável pelo gerenciamento do serviço BC. Esse grupo é formado pelas réplicas dos objetos que deverão ter sua carga balanceada. Todas as réplicas devem ser ativadas no POA, referenciado como um membro POA, e o servidor correspondente será um servidor balanceado em carga.

Referências de objetos criadas por um membro POA farão referência à instância desse membro dentro do servidor balanceado de carga. Quando um cliente faz uma requisição num objeto usando a referência criada pelo membro POA, o serviço:

- Recebe a requisição;
- Determina o grupo de balanceadores de carga;
- Seleciona um membro desse grupo;

- Retorna uma nova referência para o cliente.

O cliente então estabelece a conexão com o servidor usando a nova referência de objeto, e se comunica diretamente com o servidor, sem a intervenção do serviço.

Mas, embora o Orbacus 4.3.0 possua o serviço BC, sua inclusão na arquitetura do GWS não foi possível, pois quando o servidor era executado, surgia uma mensagem de erro apontada para o serviço de nomes**.** Foi feita uma vasta pesquisa bibliográfica além do manual de utilização do Orbacus, inclusive comunidades CORBA [Orbzone, Community for CORBA], e não foi encontrada solução para o problema. Observou-se que nessa versão do Orbacus há uma incompatibilidade entre o BC e o serviço de nomes. Foi feita a troca pela versão 4.3.3 do Orbacus [Orbacus 4.3.3, 2007], mas a mensagem de erro permaneceu. Até o momento não foram encontradas informações acerca deste problema, que ficou sem solução.

Desta forma, procurou-se novamente buscar um ORB que possuísse os recursos AMI e BC para serem adicionados ao software do GWS. Escolheu-se o Orbix, do mesmo fabricante do Orbacus, o IONA. Entretanto, notou-se desde o início que o serviço de mensagens só estava instalado na versçao para C++. Mesmo assim, desejava-se testar e verificar o serviço BC, o que foi efetivamente feito, como descrito a seguir.

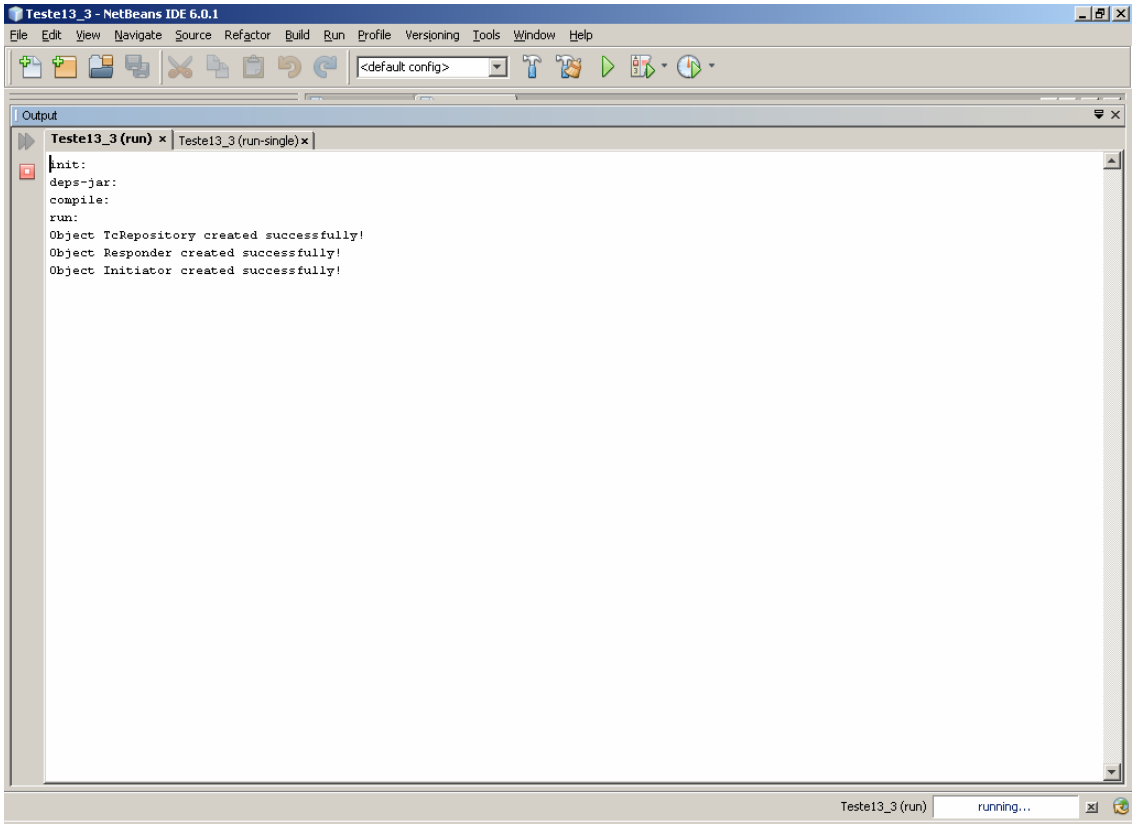

Figura 4.8. Execução do Servidor no Orbacus 4.3.0

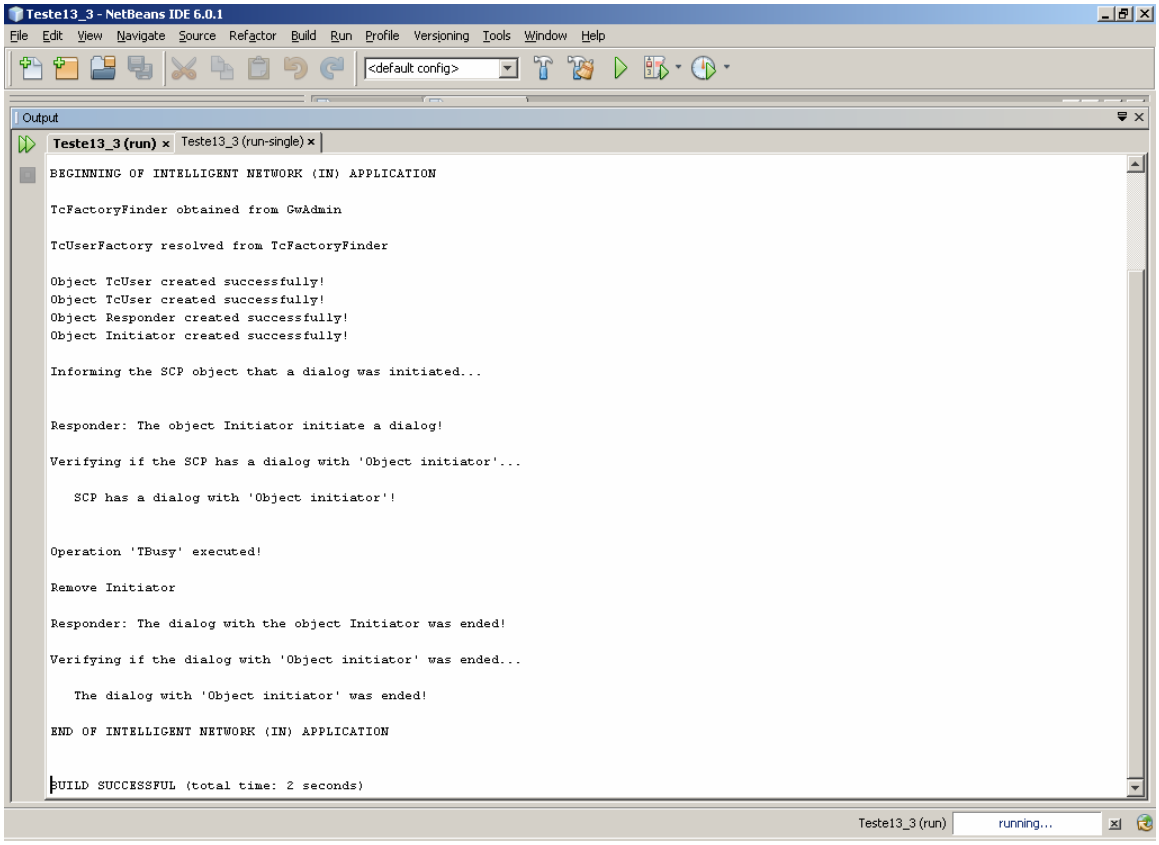

Figura 4.9. Execução do Cliente no Orbacus 4.3.0 com AMI

4.6 Orbix

Com o Orbix [Orbix, CORBA for the Enterprise] é possível desenvolver e implantar em larga escala e a nível empresarial sistemas CORBA em C++ e Java. Ele tem uma moderna arquitetura modular que permite sua configuração e mudança de funcionalidade sem modificar o código de aplicação, e possui também uma rica arquitetura de desenvolvimento que permite a configuração e gerenciamento de um complexo sistema distribuído [Orbix, 2005].

O serviço de nomes define um repositório de nomes que mapeia objetos que serão utilizados, sendo que um nome mapeia para um objeto somente. O Orbix estende o modelo do serviço de nomes, permitindo que um nome mapeie para um grupo de objetos. Esse grupo é uma coleção de objetos que pode aumentar ou diminuir de tamanho dinamicamente, dependendo da carga que esses objetos manipulam.

O Orbix suporta grupos de objetos através de suas próprias interfaces IDL. Essas interfaces permitem a criação e manipulação dos grupos de objetos, podendo adicionar e remover objetos dos grupos, e encontrando objetos que são membros de um determinado grupo. Grupos de objetos são transparentes para o cliente. Os módulos IDL *IT_LoadBalancing* e *IT_Naming*, definidos no *orbix/load_balancing.idl* e *orbix/naming.idl* respectivamente, definem operações que possibilitam o acesso ao BC no Orbix [Orbix, 2005].

As implementações do cliente (*Client)* e do servidor (*Server)* foram modificadas, de acordo com o manual do Orbix [Orbix, 2005], de forma a acessar os módulos IDL *IT_LoadBalancing* e *IT_Naming,* possibilitando a utilização BC no modelo do GWS. Essas modificações estão no Apêndice C.

Foram criados dois grupos de objetos: um contendo os iniciadores dos diálogos (SSPs proxys), e outro contendo os respondedores desses diálogos (SCPs CORBA). Esses são os objetos que serão gerenciados pelo BC, através de determinada estratégia, para distribuir de maneira equitativa a carga sobre eles.

Inicialmente adotou-se uma estratégia não-adaptativa de BC, utilizando o algoritmo *Round-Robin*. Foram criados dois clientes para que fosse possível verificar o comportamento do serviço. Os testes realizados demonstram o funcionamento correto do BC.

As Figuras 4.10 a 4.18 apresentam as capturas de tela da execução do servidor e dos clientes, que foi realizada no *prompt* de comandos do DOS.

Depois a estratégia utilizada foi modificada para adaptativa, que utiliza o algoritmo *Active load balancing.* Mais uma vez os testes realizados mostraram o funcionamento correto do serviço, isto é, as ligações entre os objetos iniciadores e respondedores ocorriam de acordo com suas cargas no momento da execução.

Para verificar a capacidade do modelo do GWS utilizando a estratégia adaptativa de BC, foi feito um teste onde foram gerados 5 objetos iniciadores dos diálogos (SSPProxy) e 5 objetos respondedores (SCP CORBA), no Servidor.

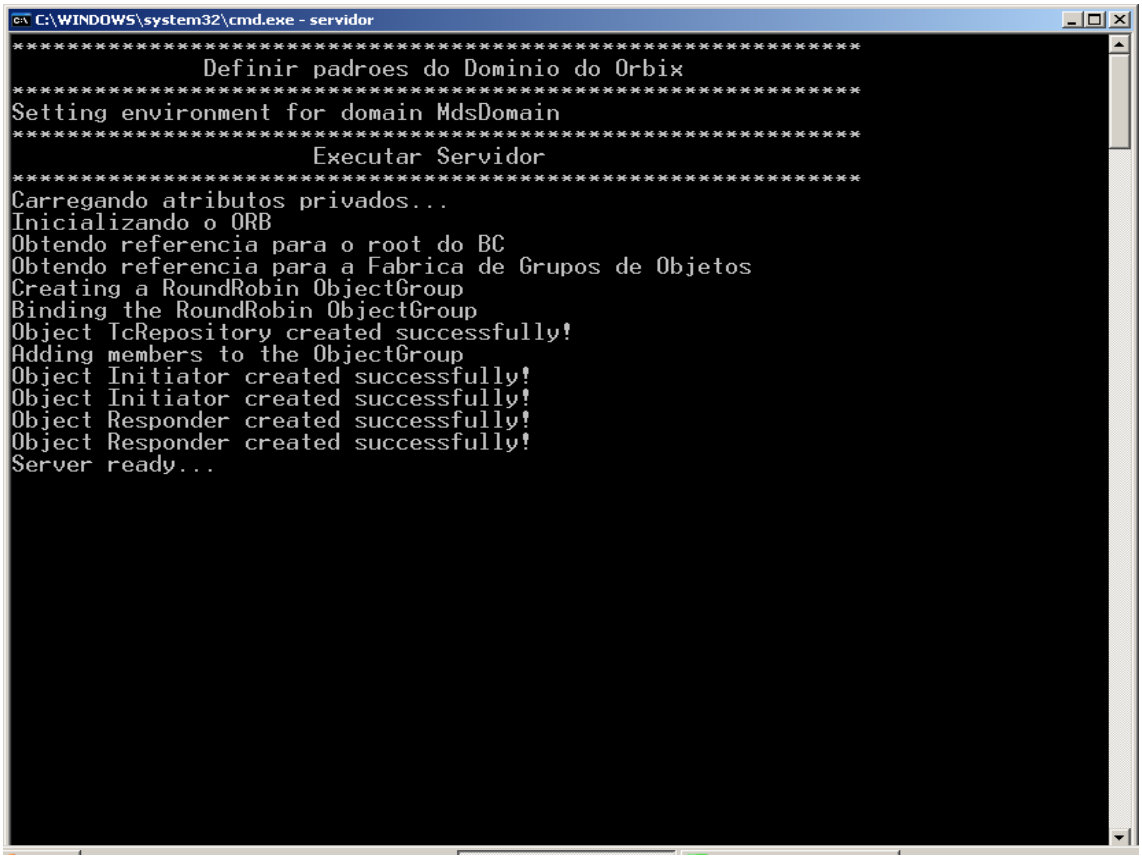

Figura 4.10. Execução do Servidor no Orbix – Aguardando clientes

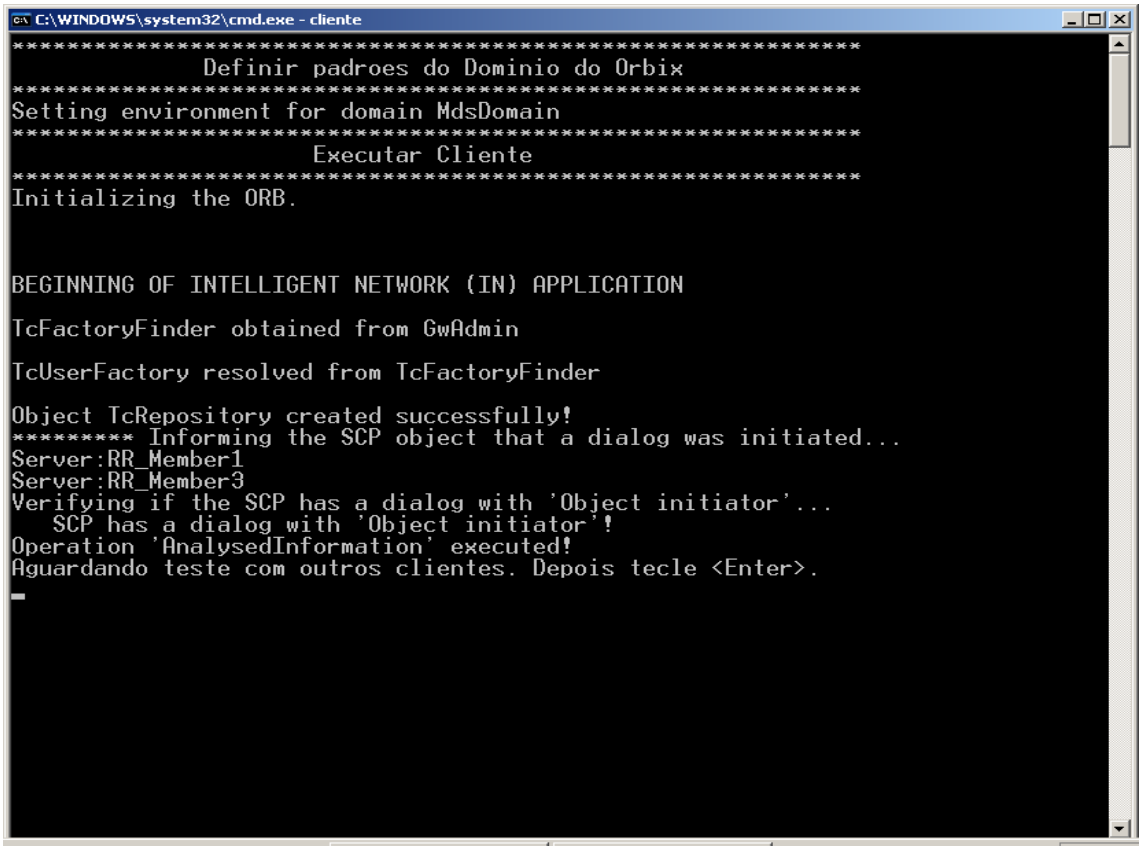

Figura 4.11. Execução do primeiro Cliente

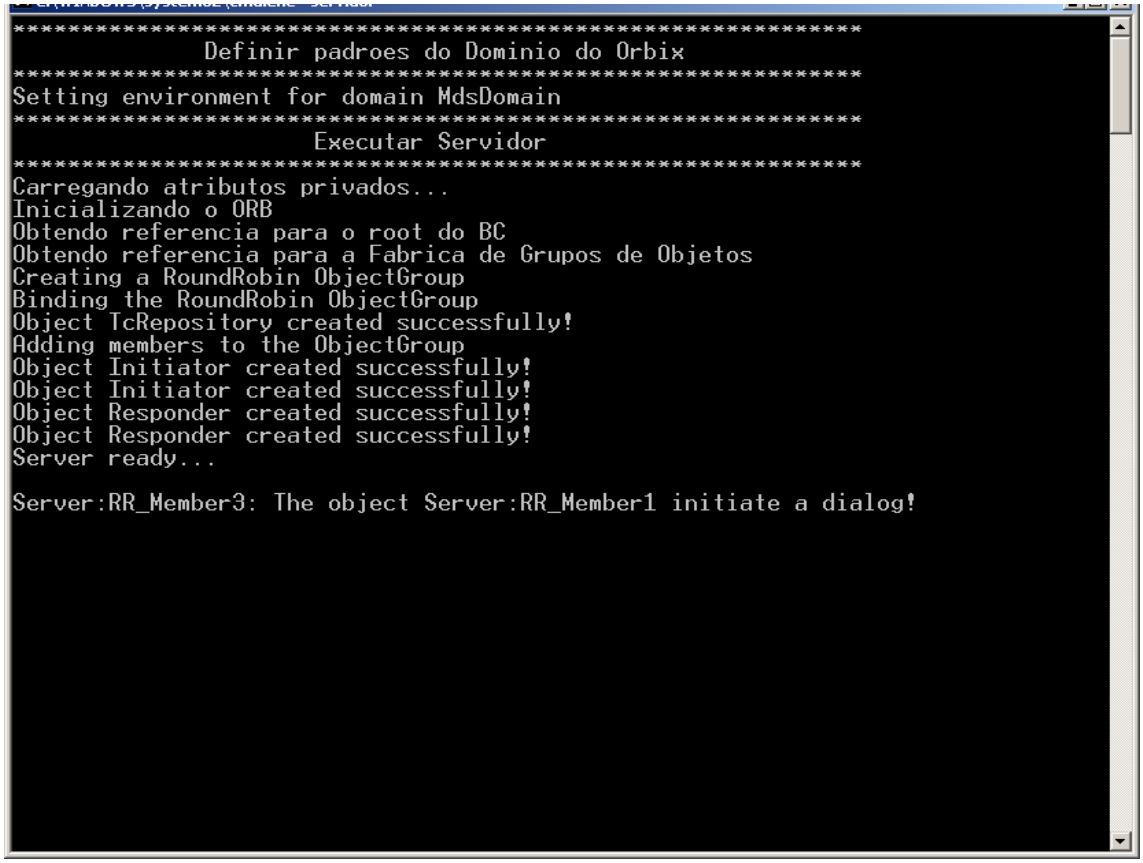

Figura 4.12 Execução do Servidor após o primeiro Cliente

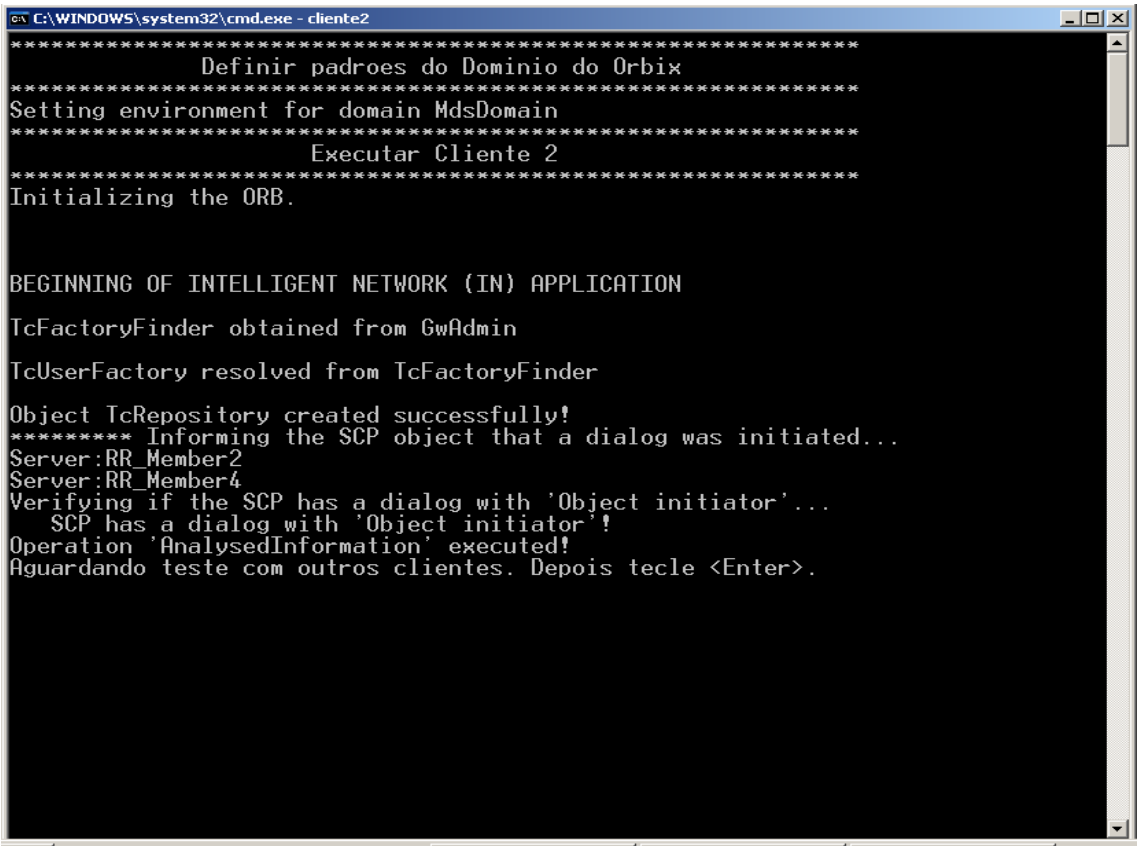

Figura 4.13. Execução do segundo Cliente

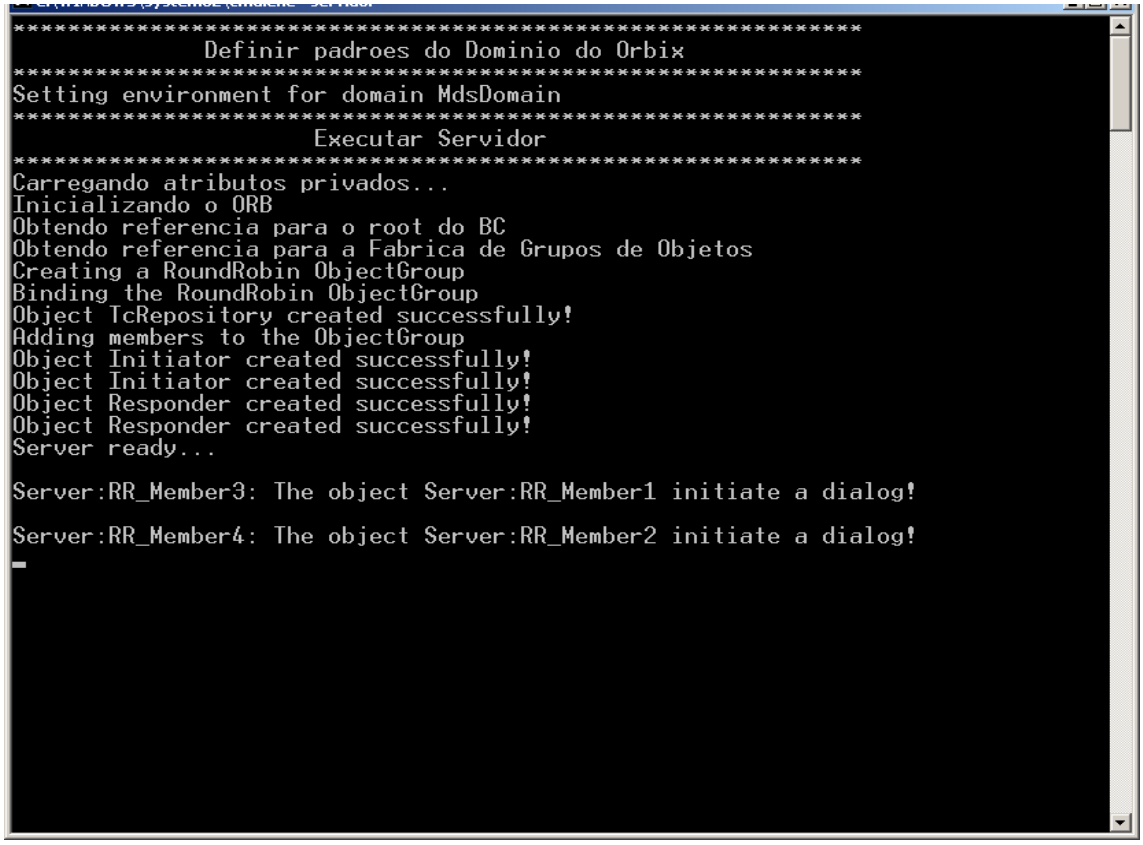

Figura 4.14. Execução do Servidor após o segundo Cliente

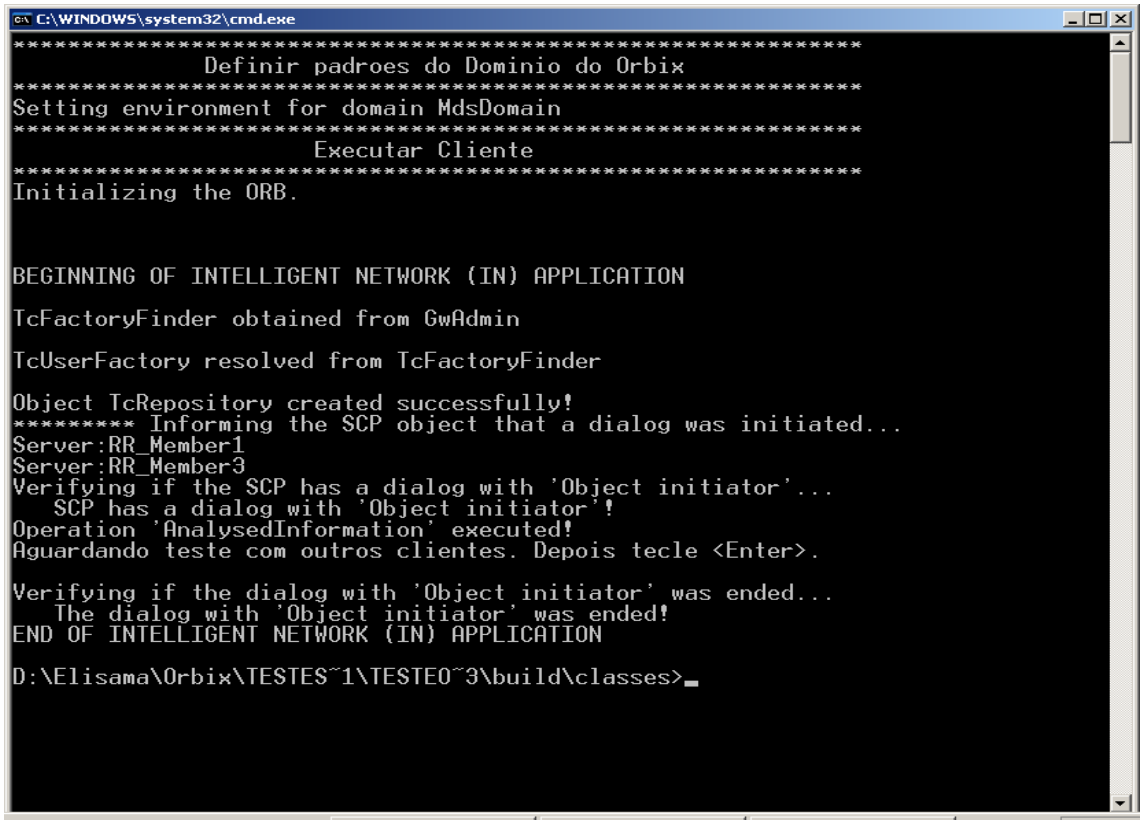

Figura 4.15. Finalizando primeiro Cliente

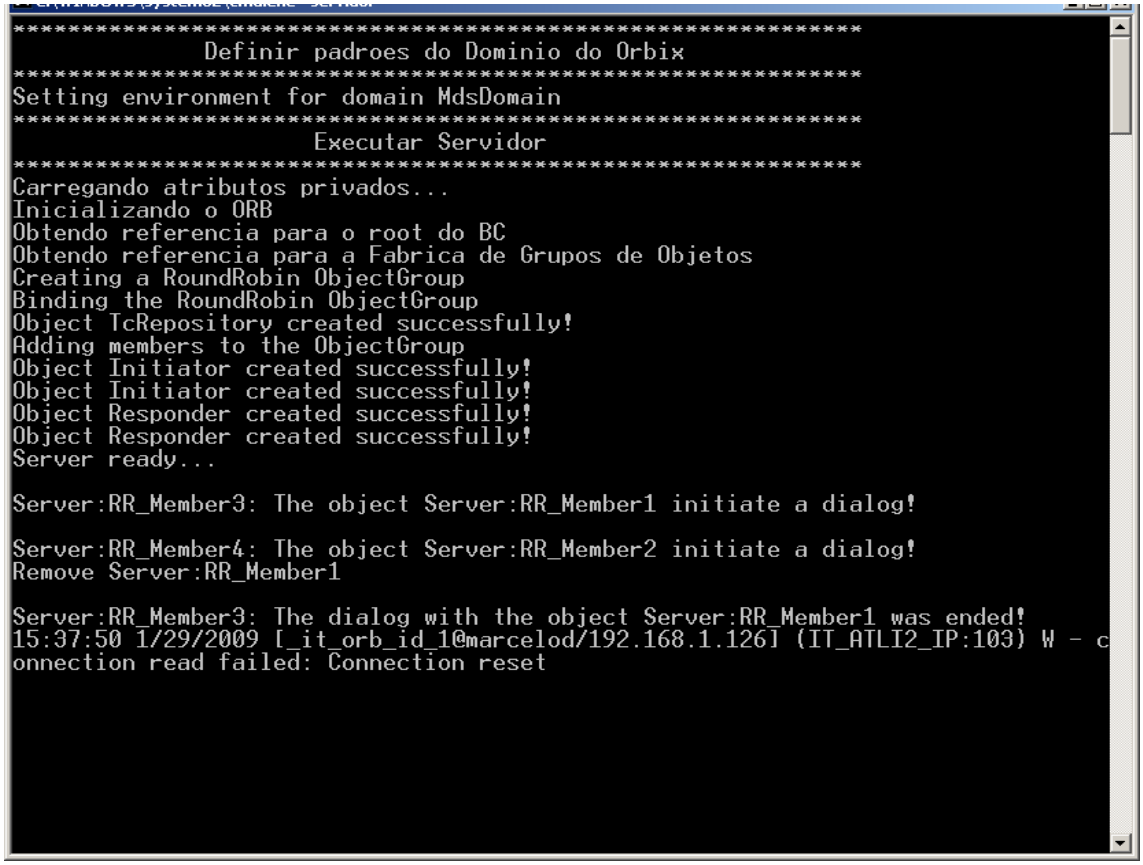

Figura 4.16. Resposta do Servidor após a finalização primeiro Cliente

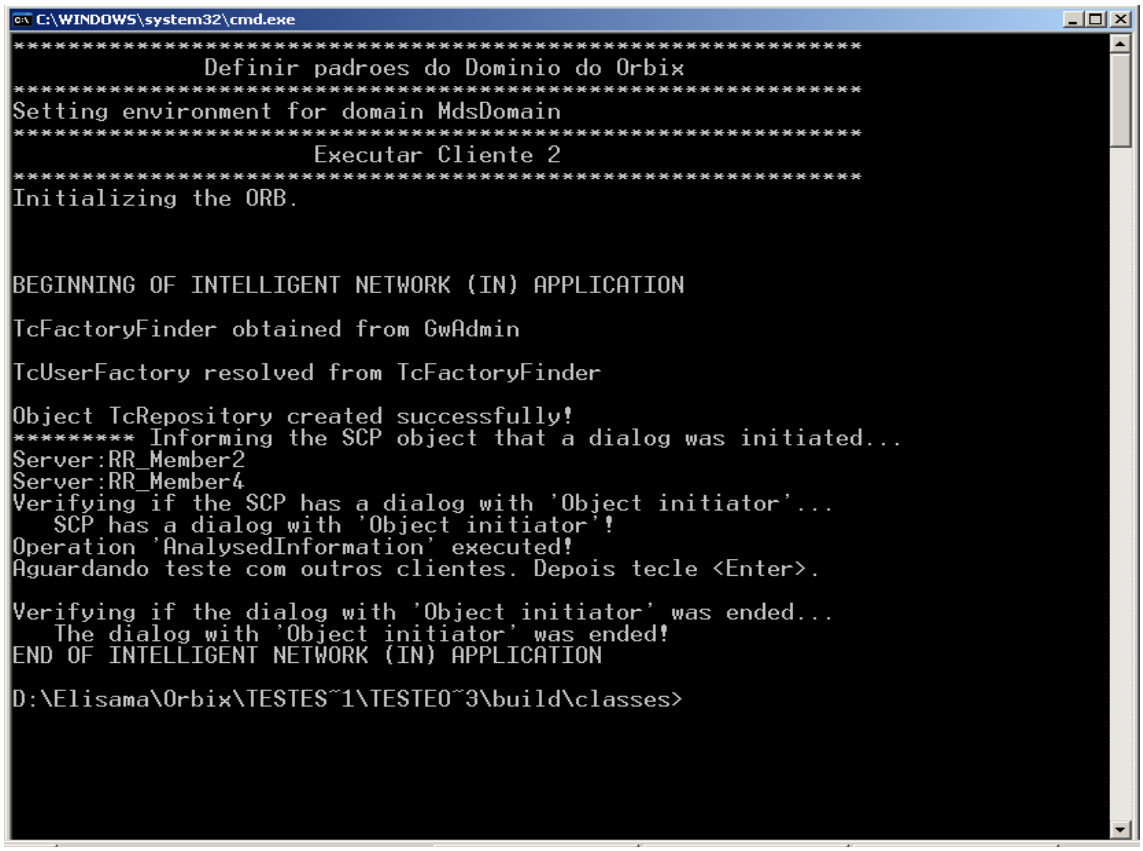

Figura 4.17.Finalizando segundo Cliente

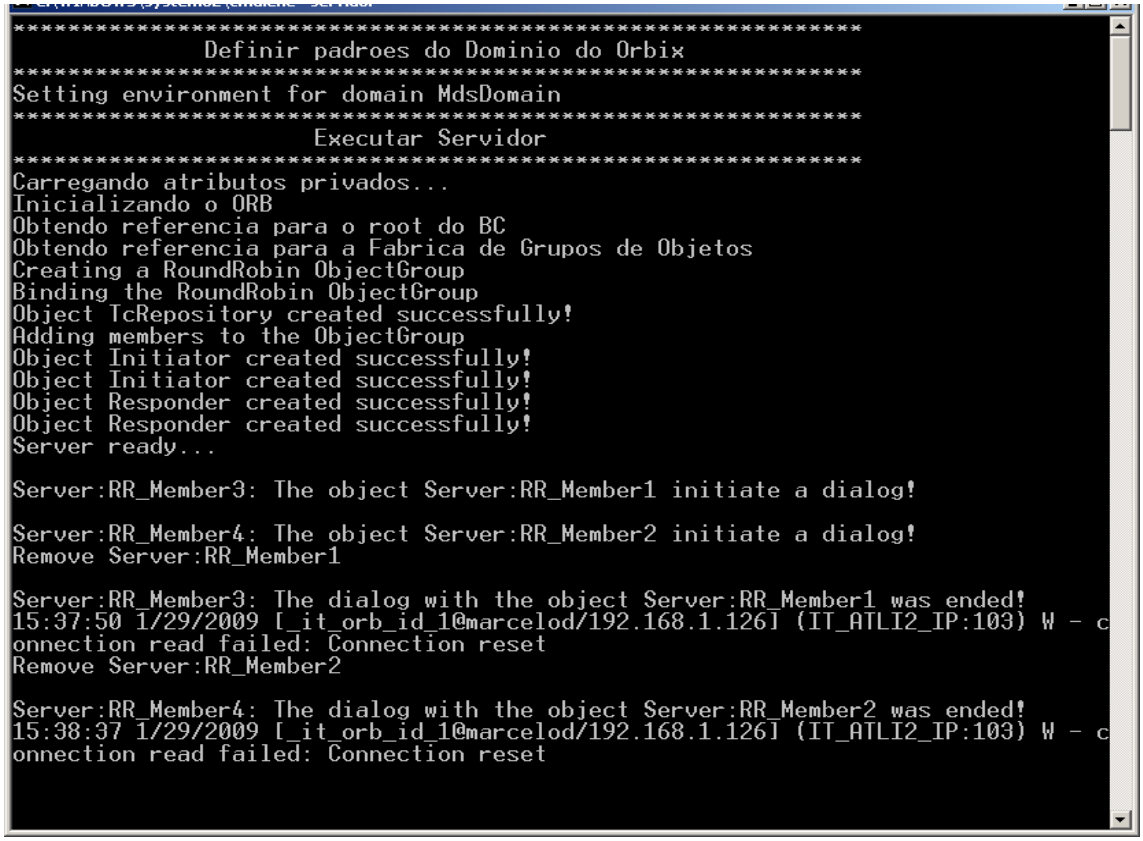

Figura 4.18 Resposta do Servidor após a finalização segundo Cliente

Foi criado um software simulador com a função de executar 50 *threads* Clientes, que faziam o papel de requisições vindas da RI convencional. Eles foram executados simultaneamente. O número de 50 *threads* foi obtido pelo motivo exposto a seguir.

Durante o teste, verificou-se que alguns objetos (iniciadores ou respondedores), que já estavam sendo utilizados, eram reutilizados pelo BC, ocasionando mensagem erro no Servidor e por essa razão a associação não era completada. Não foi possível interferir nos procedimentos do algoritmo *Active load balancing,* visto que são procedimentos internos do Orbix. Desta forma, não havia como identificar a forma usada pelo serviço de BC para verificar a carga desses elementos, e tentar fazer as associações, mesmo quando esses elementos já faziam parte de outra associação. Mas, objetos iniciadores e respondedores na forma como são definidos, permitem que se altere a quantidade de associações simultâneas que se pode realizar com cada par. Baseado nisso, para contornar o problema de geração de erros, no código do Servidor, foi alterada a quantidade de associações de 1 para 3. Depois dessa modificação, não ocorreram mais "falhas" e o GWS funcionou com 5 objetos iniciadores e 5 respondedores, podendo atender, desta forma, a 50 chamadas simultâneas provenientes de fora do ambiente CORBA, simuladas através dos 50 Clientes. A quantidade

de objetos iniciadores e respondedores podem ser alterados no software do Servidor, de acordo com a necessidade e verificadas as limitações de processamento do hardware usado. Um ponto importante a ser observado é que o aumento da capacidade de resposta do Servidor do GWS com a inclusão do BC depende da capacidade de processamento dos equipamentos (hardware).

Um programa na linguagem Delphi foi desenvolvido com o intuito de trabalhar com esses clientes na forma de *threads*. A Figura 4.19 ilustra a tela deste simulador, indicando os 50 clientes que foram executados simultaneamente.

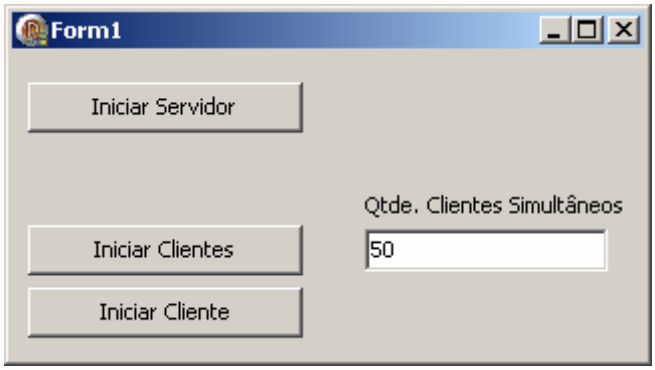

Figura 4.19. Tela do Simulador

A execução do Servidor foi armazenada num arquivo texto, e é mostrada na Figura 4.20.

```
Setting environment for domain MdsDomain
Carregando atributos privados...
Inicializando o ORB
Obtendo referencia para o root do BC
Obtendo referencia para a Fabrica de Grupos de Objetos
Criando Grupos de Objetos (MÚtodo Active)
Erro: IT_LoadBalancing.DuplicateGroup: IDL:iona.com/IT_LoadBalancing/DuplicateGr
oup:1.0
Removendo objeto id: Server:RR_Member1
Removendo objeto id: Server:RR_Member2
Removendo objeto id: Server:RR_Member3
Removendo objeto id: Server:RR_Member4
Removendo objeto id: Server:RR_Member5
Removendo objeto id: Server:RR_Member6
Removendo objeto id: Server:RR_Member7
Removendo objeto id: Server:RR_Member8
Removendo objeto id: Server:RR_Member9
Removendo objeto id: Server:RR_Member10
Executando 'Unbinding' nos Grupos de Objetos...
Destruindo Grupos de Objetos...
Executando 'Unbinding' em Grupo de Objeto...
Desttruindo Grupo...
Fechando ORB.
************************************************************
Executar Servidor
************************************************************
Carregando atributos privados...
Inicializando o ORB
Obtendo referencia para o root do BC
Obtendo referencia para a Fabrica de Grupos de Objetos
Criando Grupos de Objetos (MÚtodo Active)
Executando 'Binding' nos Grupos de Objetos
Object TcRepository created successfully!
```

```
Adicionando membros ao Grupo de Objetos
Object Initiator created successfully!
Object Initiator created successfully!
Object Initiator created successfully!
Object Initiator created successfully!
Object Initiator created successfully!
Object Responder created successfully!
Object Responder created successfully!
Object Responder created successfully!
Object Responder created successfully!
Object Responder created successfully!
Servidor pronto e aguardando...

Server:RR_Member6: The object Server:RR_Member5 initiate a dialog!
Remove Server:RR_Member5

Server:RR_Member6: The dialog with the object Server:RR_Member5 was ended!
21:50:37 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset

Server:RR_Member10: The object Server:RR_Member2 initiate a dialog!
Remove Server:RR_Member2

Server:RR_Member10: The dialog with the object Server:RR_Member2 was ended!

Server:RR_Member8: The object Server:RR_Member2 initiate a dialog!

Server:RR_Member9: The object Server:RR_Member5 initiate a dialog!
Remove Server:RR_Member2

Server:RR_Member8: The dialog with the object Server:RR_Member2 was ended!
Remove Server:RR_Member5

Server:RR_Member9: The dialog with the object Server:RR_Member5 was ended!
21:50:43 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:44 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:44 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset

Server:RR_Member9: The object Server:RR_Member1 initiate a dialog!
Remove Server:RR_Member1

Server:RR_Member9: The dialog with the object Server:RR_Member1 was ended!

Server:RR_Member9: The object Server:RR_Member1 initiate a dialog!
Remove Server:RR_Member1

Server:RR_Member9: The dialog with the object Server:RR_Member1 was ended!

Server:RR_Member8: The object Server:RR_Member3 initiate a dialog!

Server:RR_Member6: The object Server:RR_Member1 initiate a dialog!
Remove Server:RR_Member3

Server:RR_Member8: The dialog with the object Server:RR_Member3 was ended!
Remove Server:RR_Member1

Server:RR_Member6: The dialog with the object Server:RR_Member1 was ended!

Server:RR_Member7: The object Server:RR_Member5 initiate a dialog!

Server:RR_Member9: The object Server:RR_Member3 initiate a dialog!
21:50:47 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member5

Server:RR_Member7: The dialog with the object Server:RR_Member5 was ended!
Remove Server:RR_Member3

Server:RR_Member9: The dialog with the object Server:RR_Member3 was ended!

Server:RR_Member8: The object Server:RR_Member3 initiate a dialog!
21:50:48 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
```

```
Server:RR_Member7: The object Server:RR_Member2 initiate a dialog!
21:50:49 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:49 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member3

Server:RR_Member8: The dialog with the object Server:RR_Member3 was ended!
Remove Server:RR_Member2

Server:RR_Member7: The dialog with the object Server:RR_Member2 was ended!

Server:RR_Member10: The object Server:RR_Member1 initiate a dialog!
21:50:51 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset

Server:RR_Member10: The object Server:RR_Member2 initiate a dialog!
Remove Server:RR_Member1

Server:RR_Member10: The dialog with the object Server:RR_Member1 was ended!
Remove Server:RR_Member2

Server:RR_Member10: The dialog with the object Server:RR_Member2 was ended!

Server:RR_Member9: The object Server:RR_Member1 initiate a dialog!

Server:RR_Member8: The object Server:RR_Member1 initiate a dialog!
Remove Server:RR_Member1

Server:RR_Member9: The dialog with the object Server:RR_Member1 was ended!

Server:RR_Member10: The object Server:RR_Member3 initiate a dialog!
Remove Server:RR_Member1

Server:RR_Member8: The dialog with the object Server:RR_Member1 was ended!

Server:RR_Member9: The object Server:RR_Member3 initiate a dialog!
Remove Server:RR_Member3

Server:RR_Member10: The dialog with the object Server:RR_Member3 was ended!
Remove Server:RR_Member3

Server:RR_Member9: The dialog with the object Server:RR_Member3 was ended!
21:50:54 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset

Server:RR_Member10: The object Server:RR_Member4 initiate a dialog!
21:50:54 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:54 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member4

Server:RR_Member10: The dialog with the object Server:RR_Member4 was ended!
21:50:56 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:56 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:56 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:56 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:57 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:57 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:50:59 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset

Server:RR_Member6: The object Server:RR_Member1 initiate a dialog!

Server:RR_Member7: The object Server:RR_Member1 initiate a dialog!
Remove Server:RR_Member1

Server:RR_Member6: The dialog with the object Server:RR_Member1 was ended!
```

```
Server:RR_Member10: The object Server:RR_Member3 initiate a dialog!

Server:RR_Member9: The object Server:RR_Member3 initiate a dialog!
Remove Server:RR_Member1

Server:RR_Member9: The object Server:RR_Member1 initiate a dialog!

Server:RR_Member7: The dialog with the object Server:RR_Member1 was ended!

Server:RR_Member9: The object Server:RR_Member2 initiate a dialog!

Server:RR_Member8: The object Server:RR_Member5 initiate a dialog!

Server:RR_Member8: The object Server:RR_Member4 initiate a dialog!

Server:RR_Member10: The object Server:RR_Member1 initiate a dialog!

Server:RR_Member6: The object Server:RR_Member5 initiate a dialog!

Server:RR_Member6: The object Server:RR_Member5 initiate a dialog!
21:51:03 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member3

Server:RR_Member10: The dialog with the object Server:RR_Member3 was ended!
Remove Server:RR_Member1

Server:RR_Member9: The dialog with the object Server:RR_Member1 was ended!
Remove Server:RR_Member3

Server:RR_Member9: The dialog with the object Server:RR_Member3 was ended!
Remove Server:RR_Member5

Server:RR_Member8: The dialog with the object Server:RR_Member5 was ended!
Remove Server:RR_Member2

Server:RR_Member9: The dialog with the object Server:RR_Member2 was ended!
Remove Server:RR_Member1

Server:RR_Member10: The dialog with the object Server:RR_Member1 was ended!
Remove Server:RR_Member4

Server:RR_Member8: The dialog with the object Server:RR_Member4 was ended!
Remove Server:RR_Member5

Server:RR_Member6: The dialog with the object Server:RR_Member5 was ended!
Remove Server:RR_Member5

Server:RR_Member6: The dialog with the object Server:RR_Member5 was ended!

Server:RR_Member8: The object Server:RR_Member1 initiate a dialog!
21:51:06 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member1

Server:RR_Member8: The dialog with the object Server:RR_Member1 was ended!

Server:RR_Member9: The object Server:RR_Member1 initiate a dialog!
21:51:06 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset

Server:RR_Member8: The object Server:RR_Member3 initiate a dialog!
21:51:07 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member1

Server:RR_Member9: The dialog with the object Server:RR_Member1 was ended!
Remove Server:RR_Member3

Server:RR_Member8: The dialog with the object Server:RR_Member3 was ended!
21:51:07 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:07 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:07 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
```

ction read failed: Connection reset
21:51:07 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:07 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:07 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:07 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset

Server:RR_Member6: The object Server:RR_Member1 initiate a dialog!

Server:RR_Member9: The object Server:RR_Member5 initiate a dialog!

Server:RR_Member8: The object Server:RR_Member1 initiate a dialog!

Server:RR_Member7: The object Server:RR_Member4 initiate a dialog!
Remove Server:RR_Member5

Server:RR_Member9: The dialog with the object Server:RR_Member5 was ended!
Remove Server:RR_Member1

Server:RR_Member8: The dialog with the object Server:RR_Member1 was ended!

Server:RR_Member9: The object Server:RR_Member3 initiate a dialog!
21:51:08 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member1

Server:RR_Member6: The dialog with the object Server:RR_Member1 was ended!

Server:RR_Member10: The object Server:RR_Member4 initiate a dialog!

Server:RR_Member6: The object Server:RR_Member2 initiate a dialog!
Remove Server:RR_Member4

Server:RR_Member7: The dialog with the object Server:RR_Member4 was ended!

Server:RR_Member6: The object Server:RR_Member4 initiate a dialog!

Server:RR_Member8: The object Server:RR_Member5 initiate a dialog!

Server:RR_Member10: The object Server:RR_Member5 initiate a dialog!

Server:RR_Member8: The object Server:RR_Member2 initiate a dialog!

Server:RR_Member9: The object Server:RR_Member2 initiate a dialog!

Server:RR_Member10: The object Server:RR_Member3 initiate a dialog!

Server:RR_Member7: The object Server:RR_Member2 initiate a dialog!

Server:RR_Member6: The object Server:RR_Member5 initiate a dialog!
Remove Server:RR_Member3

Server:RR_Member9: The dialog with the object Server:RR_Member3 was ended!

Server:RR_Member7: The object Server:RR_Member1 initiate a dialog!
Remove Server:RR_Member4

Server:RR_Member10: The dialog with the object Server:RR_Member4 was ended!
Remove Server:RR_Member2

Server:RR_Member6: The dialog with the object Server:RR_Member2 was ended!

Server:RR_Member8: The object Server:RR_Member2 initiate a dialog!
21:51:09 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member4

Server:RR_Member6: The dialog with the object Server:RR_Member4 was ended!
21:51:09 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member5

Server:RR_Member8: The dialog with the object Server:RR_Member5 was ended!

```
Remove Server:RR_Member2

Server:RR_Member8: The dialog with the object Server:RR_Member2 was ended!
21:51:10 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member2

Server:RR_Member7: The dialog with the object Server:RR_Member2 was ended!
Remove Server:RR_Member5

Server:RR_Member10: The dialog with the object Server:RR_Member5 was ended!
Remove Server:RR_Member2

Server:RR_Member9: The dialog with the object Server:RR_Member2 was ended!
21:51:10 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
Remove Server:RR_Member3

Server:RR_Member10: The dialog with the object Server:RR_Member3 was ended!
Remove Server:RR_Member5

Server:RR_Member6: The dialog with the object Server:RR_Member5 was ended!
Remove Server:RR_Member2

Server:RR_Member8: The dialog with the object Server:RR_Member2 was ended!
Remove Server:RR_Member1

Server:RR_Member7: The dialog with the object Server:RR_Member1 was ended!
21:51:10 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:10 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:10 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
21:51:11 3/15/2009 [_it_orb_id_1@marcelod/127.0.0.1] (IT_ATLI2_IP:103) W - conne
ction read failed: Connection reset
```

Figura 4.20. Arquivo de *log* da execução do Servidor com 50 Clientes

A execução de um dos 50 clientes é apresentada na Figura 4.21, a seguir.

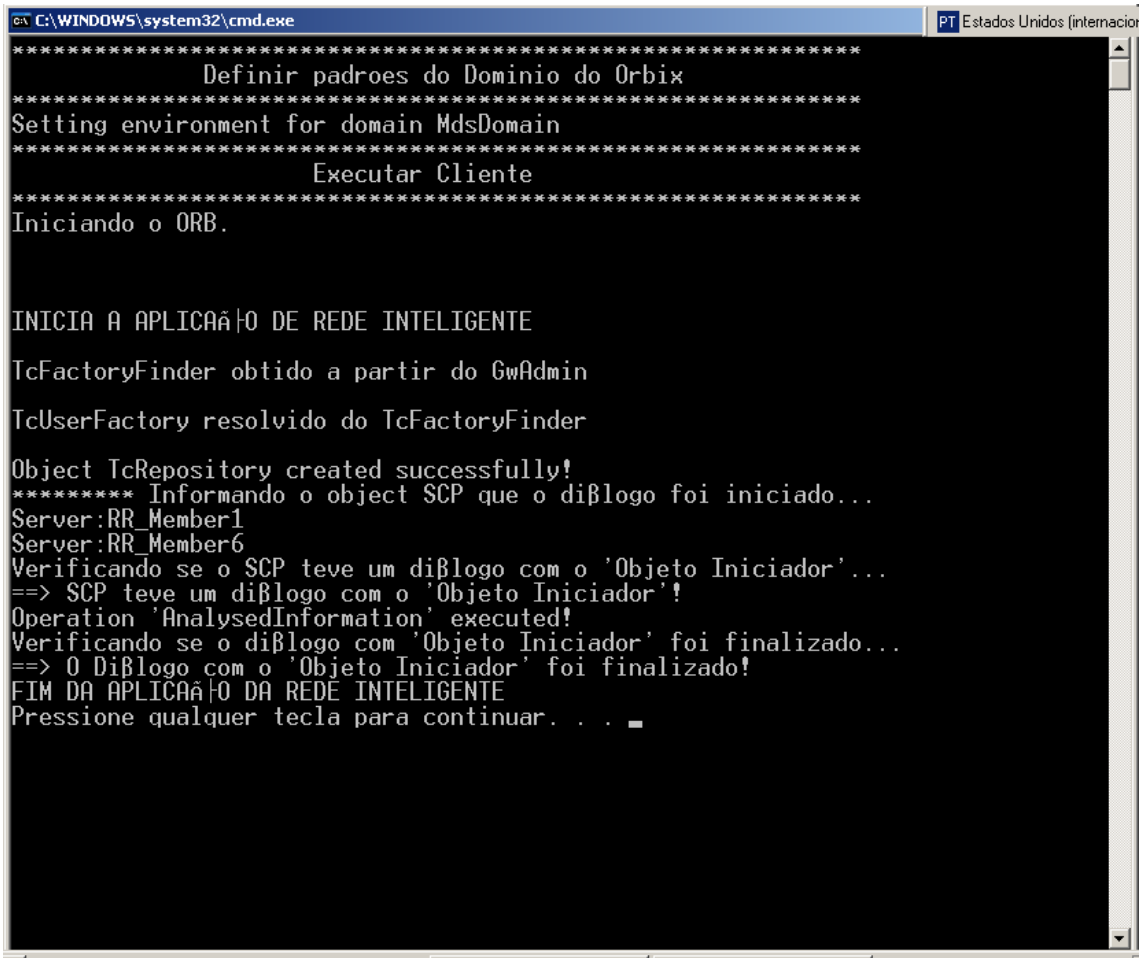

Figura 4.21. Execução de um Cliente

Como dito no Capítulo 3, a idéia era associar o AMI ao BC em uma mesma plataforma, de forma a melhorar ainda mais o desempenho da arquitetura do GWS. Porém, verificou-se que a versão do Orbix para Java não executa o AMI, somente a versão para C++. Portanto, esta foi a limitação encontrada ao programar no Orbix. Sendo assim, não foi possível incorporar os dois serviços juntos.

4.7 Conclusões

Para realizar a implementação do software do GWS e inserir os serviços AMI (no Orbacus) e BC (no Orbix) foi necessário um grande esforço de programação, ficando patente que a complexidade de implementar sistemas distribuídos baseados no CORBA é grande.

Trocar de ORB para executar os programas também foi um problema, pois demandou tempo excessivo, a instalação, a criação e configuração do ambiente de trabalho, funcionamento dos serviços CORBA, compilação das IDLs, entre outras coisas. Isto porque, os ORBs têm peculiaridades distintas, que dificultaram ainda mais o andamento do trabalho.

Em resumo, verificou-se que a idéia de juntar vários recursos do CORBA num mesmo ORB, dentre os escolhidos, ficou impraticável, pois cada um dispõe de recursos limitados. A Tabela 4.1 a seguir faz uma comparação entre os serviços disponíveis em cada ORB Java utilizado nesse trabalho.

Tabela 4.1. Comparação entre os serviços disponíveis em cada ORB Java

ORB	AMI	BC
Jacorb	**Sim**	**Não**
Orbacus	**Sim**	**Sim**
Orbix	**Não**	**Sim**

O ORB TAO, para C++, e a versão do Orbix para C++ possuem ambos os serviços. Porém como esse trabalho foi realizado em Java, não foi possível testá-los.

Apesar do método AMH ser uma ferramenta interessante a ser considerada, ele não foi implementado por ser de propriedade do TAO, um ORB para C++.

5. Conclusões

A proposta do presente trabalho foi a produção de um software para tratar num ambiente distribuído das requisições de serviços de valor agregado realizados atualmente pelas Redes Inteligentes. Esse software pode ser incorporado em um gateway de sinalização, cuja principal função é a de adaptar a sintaxe e semântica de execução de serviços da Rede Inteligente convencional (via SS7) às requisições de operações em servidores de aplicações baseados no CORBA.

Além da geração do software do GWS, percebeu-se em pesquisas anteriores [Soares, 2003], [Oliveira, 2004], a necessidade de incluir no modelo de GWS do OMG, recursos que proporcionassem mais escalabilidade aos objetos de execução dos serviços, tanto quanto melhorasse a vazão das mensagens no ambiente CORBA. Tais recursos podem ser conseguidos por meio de serviços especiais de ORB, que são padronizados. Portanto, considerando a vantagem de usar apenas recursos de ORB para esse fim, outra proposta deste trabalho foi a inserção de dois serviços de ORB na arquitetura do GWS: o Serviço de Mensagens (AMI) e o de Balanceamento de Carga, com a finalidade de melhorar a QoS do modelo.

Foi cogitada também a inclusão de uma técnica nova e eficiente, para permitir o tratamento de respostas de forma assíncrona por parte dos servidores de aplicação, chamada de *Asynchronous Method Handling* (AMH). No entanto, por ela ser proprietária de um Orb que não foi testado neste trabalho, o TAO, ela foi explicada, mas ficou fora do projeto.

O trabalho foi executado sobre três ORBs para Java nesta sequência: o JacOrb, o Orbacus e o Orbix.

Primeiramente o software do GWS foi construído sem considerar os serviços especiais de Orb. Ele foi bem executado nos três Orbs, sem apresentar qualquer problema.

Com o GWS funcionando no JacOrb, partiu-se para a programação do Serviço de Mensagens (AMI) nas chamadas do GWS para o servidor de aplicações. Porém a execução deste serviço apresentou falhas que não puderam ser resolvidas, devido a problemas internos

do JacOrb, verificados em sua própria documentação. O JacOrb foi o primeiro ORB a ser testado, devido à sua aplicação nos trabalhos usados como referências. Entretanto, por ele não conter o serviço de balanceamento de carga, a segunda opção foi o Orbacus.

No Orbacus, o software do GWS em conjunto com o AMI funcionou perfeitamente, como ficou demonstrado no Capítulo 4.

Em seguida foi testado no Orbacus o software do GWS separadamente com o BC, sem incluir o AMI. Ao executar o servidor, surgiu uma incompatibilidade entre o Serviço de Nomes, amplamente usado pelos objetos do GWS e o serviço de BC. A dificuldade foi sanada com a substituição do Orbacus pelo Orbix, o qual estende seu Serviço de Nomes de forma a permitir que um "nome" mapeie para um grupo de objetos, conforme a necessidade do balanceamento de cargas.

No Orbix, o software do GWS foi ampliado com o serviço de BC. Foram necessárias várias modificações nas implementações do cliente (*Client)* e do servidor (*Server),* para permitir o acesso deles aos módulos IDL *IT_LoadBalancing* e *IT_Naming*. Vários testes foram realizados para avaliar a capacidade do GWS em balancear as cargas dos elementos iniciadores e respondedores dos diálogos, o SSP Proxy e o SCP CORBA, respectivamente. Nesses testes ficou comprovada a capacidade de manipulação de várias requisições de serviço provenientes de diferentes clientes pelo balanceador de carga. Observou-se também que a capacidade de criação dos grupos de objetos ficou limitada à capacidade de processamento do hardware utilizado, porque tanto os clientes quanto o GWS foram implementados no mesmo equipamento. Para se ter uma idéia, com um grupo de cinco objetos iniciadores e um grupo de cinco objetos respondedores, o BC conseguiu manipular 50 requisições de serviços simultâneas.

No Orbix, não foi possível incluir o Serviço de Mensagens, simplesmente porque ele só existe na versão para C++, na versão Java ele ainda não foi instalado.

Em resumo, a proposta de juntar vários serviços de Orb na mesma plataforma do GWS não foi realizada a contento, devido às limitações das versões atuais do JacOrb, do Orbacus e do Orbix.

Apesar da facilidade de trabalhar com a linguagem Java em relação ao C++, no momento percebe-se que os ORBs para C++ são mais completos, pois disponibilizam mais

serviços especiais de ORB. Pelo menos em princípio, o Orbix para C++ e o TAO, por exemplo, atenderiam aos objetivos deste trabalho.

As maiores contribuições deste trabalho, apesar das limitações verifidas, relativamente aos ORBs, foram:

a) A construção de parte do GWS para a interligação de uma RI convencional com um ambiente CORBA;

b) Verificação de que apesar da disponibilidade de vários ORBs para a implementação da arquitetura do GWS, existe o problema da falta de portabilidade do código, devido a diferenças de implementação de serviços de ORB, além do fato de nem todos eles estarem disponíveis.

Como mais uma contribuição, acrescenta-se a publicação de dois trabalhos em simpósios de grande projeção nacional:

1) Um Mediador de Serviços entre as Redes de Próxima Geração e a RTPC, apresentado no MOMAG 2008, que foi realizado em Florianópolis, SC, no período de 07 a 10 de setembro de 2008;

2) Inserção da RTP na NGN: O Gateway de Sinalização, CBA 2008, Juiz de Fora, MG, que aconteceu no período de 14 a 17 de setembro de 2008.

Tendo em vista, que somente parte do gateway de sinalização foi implementada e, visando melhorar ainda mais o desempenho da sua arquitetura, acresecenta-se como sugestão para futuros trabalhos:

b) Para completar o software do GW especificado pelo OMG, é necessáio realizar o mapeamento do protocolo INAP – SS7 na linguagem IDL e, o mapeamento das operações escritas em IDL, em primitivas da SS7. Esses mapeamentos não são fáceis de realizar, pois são dois ambientes bem distintos de execução de serviços, a rede SS7 e a arquitetura cliente-servidor do CORBA. No entanto, uma base para a construção dessa parte do GWS pode ser encontrada em [OMG, 1998] e [Soares, 2003].

c) Implementar a arquitetura do GWS no TAO ou ORBIX para C++, para o cenário explorado neste trabalho, incluindo os mesmos serviços de ORB para melhorar o desempenho (escalabilidade e vazão). Usando o TAO, fica também como sugestão buscar uma forma de implementar o método AMH no servidor final (PCS CORBA), mesmo não sendo uma técnica ainda padronizada.

d) Implementar os PLSs por meio de objetos agentes móveis e distribuí-los em diversos pontos da rede (mais próximos dos usuários, por exemplo) para prover mais escalabilidade à arquitetura RI e reduzir o tráfego na SS7, melhorando os tempos de processamento dos serviços e consequentemente, melhorando o aprovisionamento deles. Uma idéia complementar seria introduzir o serviço de balanceamento de cargas como meio de gerenciar a carga entre os clones espalhados em diferentes hardwares.

Referências Bibliográficas

Alcatel. (2001). A Step-By-Step Migration Scenario from PSTN to NGN. *Technical Paper*.

Arulanthu, A. B. (2000). The Design and Performance of a scalable ORB Architecture for CORBA Asynchronous Messaging. Disponível em http://www.cs.wustl.edu/schimidt/PDF/ami2.pdf

Bhat, R. R.; Gupta, R. (2000). JAIN Protocol APIs. *IEEE Communications Magazine*, pp. 100-107.

Bodart, N. L.; Durães, E. R. C.; Soares, R. B.; Garcia, A. S. (2004). Balanceamento de Carga nas Redes Inteligentes Distribuídas. Quinto Workshop em Sistemas Computacionais de Alto Desempenho – WSCAD, Foz do Iguaçu, PR.

Business Wire (2000). TINA Standardization to Continue at the OMG, Ensures that TINA-C's Work is Continued and Strengthened by a International Standards Body, Disponível em http://www.thefreelibrary.com/TINA+Standardization+to+Continue+at+the+OMG%3B+Ensures+that+TINA-C's...-a065457101, Acessado em 15/08/2009.

CORBA 2.4 Specifications. The Common Object Request Broker: Architecture and Specification. Disponível em ftp://ftp.omg.org/pub/docs/formal/01-02-01.pdf>. Acessado em 16/03/2009.

Deshpande, M., Schmidt, D. C., O'Ryan, C., Brunsch, D. (2002). Design and Performance of Asynchronous Method Handling for CORBA. Disponível em http://www.cs.wustl.edu/~schmidt/PDF/DOA-02.pdf. Acessado em 16/03/2009.

Faynberg, I.; Gato, J.; Lu, H. (2002). (PSTN/IN) Requesting InTernet Service (SPIRITS) Protocol Requirements. IETF RFC 3298 (Informational), Internet Engineering Task Force.

Filho, H. B., Sanches, W. P. (2004). NGN. disponível em http://www.teleco.com.br/tutoriais/tutorialngn/Default.asp. Acessado em 16/03/2009.

Gobbi, D. M. (2005). Uma Implementação de Parte de um Gateway SS7/CORBA para prover Manutenção de Diálogos e Invocação de Operações. Projeto de Graduação, Vitória, ES.

Grech, M. L. F.; Mckinney, R. D.; Sharma, S; Stanaway, J. J.; Varney, D. W.; Vemuri, K. V. (2000). Delivering Seamless Services in Open Networks Using Intelligent Service Mediation. *Bell Labs Technical Journal*, pp. 186-202.

Gurbani, V.; Faynberg, I. (2004). The SPIRITS (Service in the PSTN/IN Requesting InTernet Service) Protocol. IETF Internet Draft, Internet Engineering Task Force.

Heinisch, A. M. C. (2006). NGN I: Histórico da Padronização da NGN pelo ITU-T. Disponível em http://www.teleco.com.br/tutoriais/tutorialngnI/Default.asp. Acessado em 16/03/2009.

Heinisch, A. M. C. (2006). NGN II: Visão Geral – Recomendação Y2001. Disponível em http://www.teleco.com.br/tutoriais/tutorialngnII/Default.asp. Acessado em 16/03/2009.

IETF document (2002). SPIRITS Protocol Requirements, Disponível em http://ietfreport.isoc.org/idref/draft-ietf-spirits-reqs/, Acessado em 15/08/2009.

IONA Technologies. Disponível em http://www.iona.com/. Acessado em 16/03/2009.

ITU-T – Recomendações série Q. Disponível em http://www.itu.int/rec/T-REC-Q/e. Acessado em 16/03/2009.

ITU-T – Recomendação Y.2001 (2004). Next Generation Networks – Frameworks and functional architecture models - General overview of NGN.

ITU-T – Recomendação Y.2011 (2004). Next Generation Networks – Frameworks and functional architecture models – General Principles and general reference model for Next Generation Networks.

JacORB. Disponível em http://www.jacorb.org/. Acessado em 16/03/2009.

JSR 32: JAIN™ SIP API Specification (2006). Disponível em http://jcp.org/en/jsr/detail?id=32#orig, Acessado em 15/08/2009.

Keijzer, J.; Tait, D.; Goedman, R. (2000). JAIN: A New Approach to Services in Communications Networks. *IEEE Communications Magazine*.

Malini, R. Q. (2004). Uma Proposta de Implementação de um Gateway de Comunicação entre a RI Tradicional e a RI Distribuída. Projeto de Graduação, Vitória, ES.

Nassif, A. T. (2004). Redes da Próxima Geração: Aspectos Econômicos, Técnicos e Cenários de Migração. Dissertação de Mestrado. Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF.

NetBeans 6.0.1 (2007) Disponível em http://www.netbeans.org/community/releases/index.html. Acessado em 16/03/2009.

Oliveira, N. L. (2004). Uma Metodologia de Balanceamento de Carga nas Redes Inteligentes Distribuídas. Dissertação de Mestrado em Informática, Universidade Federal do Espírito Santo, Vitória, ES.

OMG. (1998). Interworking Between CORBA and TC Systems. Document Number: telecom/98-10-0319. Disponível em < ftp://ftp.omg.org/pub/docs/telecom/98-10-03.pdf>. Acessado em 16/03/2009.

Orbacus 4.3.0. User Guide. Version 4.3. (Jun. 2005)

Orbacus 4.3.3. Using Orbacus. Version 4.3 SP2. (Feb. 2007).

Orbacus Technical Review. IONA Technologies. (Feb. 2007)

Orbix. CORBA for the Enterprise. Disponível em http://www.iona.com/products/orbix/welcome.htm. Acessado em 16/03/2009.

Orbix. CORBA Programmer's Guide, Java. Version 6.3. (Dec. 2005).

Orbzone. Community for CORBA. Disponível em http://www.orbzone.org. Acessado em 16/03/2009.

Othman, O.; O'ryan, C.; Schmidt, D. C. (2001-1). Strategies for CORBA Middleware-Based Load Balancing. *IEEE DS Online*, Volume 2, Number 3.

Othman, O.; O'Ryan, C.; Schmidt, D. C. (2001-2). The Design and Performance of an Adaptive CORBA Load Balancing Service. Disponível em http://www.cs.wustl.edu/~schmidt/PDF/load_balancing.pdf. Acessado em 16/03/2009.

Parlay Group. Disponível em http://www.parlay.org. Acessado em 16/03/2009.

Parlay X3.0 Specifications (2007). Disponível em http://portal.etsi.org/docbox/TISPAN/Open/OSA/ParlayX30.html, Acessado em 15/08/2009.

Project P508. (1997). EURESCOM. Evolution, Migration and Interworking to TINA. Disponível em: < http://www.eurescom.de/. > . Acessado em 16/03/2009.

Project P607. (1997). EURESCOM. Top-down Approch Applied to Multimidia Services. Disponível em: < http://www.eurescom.de/. > . Acessado em 16/03/2009.

Project P847. (1999). EURESCOM. What is TINA and is it Usefull for the TelCos? Disponível em: < http://www.eurescom.de/. > . Acessado em 16/03/2009.

Project P909. (1997). EURESCOM. Enabling Technologies for IN Evolution and IN-Internet Integration. Disponível em: < http://www.eurescom.de/. >. Acessado em 16/03/2009.

Q1205 (1993). ITU-T. Intelligent Network Physical Plane Architecture, Helsink.

Q1213 (1995). ITU-T. Global Functional Plane for Intelligent Network CS-1. Genebra.

Q1214 (1993). ITU-T. Distributed Functional Plane for Intelligent Network CS-1. Genebra.

Q1218 (1998). ITU-T. Interface Recommendation for Intelligent Network CS-1. Genebra.

Real-time Communications Concepts: PINT (2008), Disponível em http://msdn.microsoft.com/pt-br/library/aa923431.aspx, Acessado em 15/08/2009.

SIGTRAN – Signalling Transport. Disponível em http://www.ietf.org/html.charters/sigtran-charter.html. Acessado em 16/03/2009.

Soares, R. B. (2003). Estratégias de Introdução de Sistemas Distribuídos nas Redes Inteligentes. Tese de Doutorado, Universidade Federal do Espírito Santo, ES.

Soares, R. B., Durães, E. R. C. (2008-1). Um Mediador de Serviços entre as Redes de Próxima Geração e a RTPC, MOMAG 2008, Florianópolis, SC.

Soares, R. B., Durães, E. R. C. (2008-2). Inserção da RTP na NGN: O Gateway de Sinalização, CBA 2008, Juiz de Fora, MG.

Sun Microsystems Inc. (1999-1). JAIN INAP API Specification 1.2, Public Review Draft.

Sun Microsystems Inc. (1999-2). JAIN TCAP API Specification 1.0, Public Review Draft.

The JacOrb Team. (2007). JacORB 2.3 Programming Guide. Disponível em http://www.jacorb.org/documentation.html. Acessado em 16/03/2009.

TINA-C. TINA-IN Workgroup, IN Evolution Problem Statement. Disponível em: < http://www.tinac.com/specifications/specifications.htm> . Acessado em 16/03/2009.

Tronco, T. R. (2006). Redes de Nova Geração. 1ª Edição, Ed. Érica.

Apêndice 1: Códigos Java do Servidor e Cliente

A1.1 Código do Servidor

```
/*
 * Server.java
 *
 */

package impl;

import org.omg.CosNaming.*;
import java.io.*;
import orbFiles.TcSignaling.*;
import orbFiles.TC_User.*;

public class Server {


  public static void main( String[] args ) {

                        orbFiles.TcSignaling.TcUserGenericFactory tc_user_factory;
                        orbFiles.TcSignaling.TcFactoryFinder tc_factory_finder;
                        orbFiles.TcSignaling.GwAdmin gw_admin;
                        orbFiles.TcSignaling.TcServiceFinder tc_service_finder;
                        orbFiles.TcSignaling.TcRepository tc_repository;
                        NamingContextExt gt_root;
                        orbFiles.TcSignaling.TcUser tc_user_initiator;
                        orbFiles.TcSignaling.TcUser tc_user_responder;

                        // Não utilizado nesta fase do projeto. É apenas passado com parametro.
                        orbFiles.TcSignaling.TcPduProviderFactory tc_pdu_provider_factory;

    // Variaveis Auxiliares
    String addr = new String();
    String a_c  = new String();
    String resp_iface = new String();
    String init_iface = new String();

    try {
                        //********************************************************************
                        //              Inicializa os servicos
                        //********************************************************************

                        // Inicializa o ORB
                        org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);

                        // ObtÃ©m a referÃªncia para o POA
                        org.omg.PortableServer.POA poa =org.omg.PortableServer.POAHelper.narrow(
                        orb.resolve_initial_references("RootPOA"));

                        // Ativa o POA
                        poa.the_POAManager().activate();
```

```
                    // Obtém a referência para o Serviço de Nomes do CORBA
                    NamingContextExt nc_CORBA = NamingContextExtHelper.narrow(
                    orb.resolve_initial_references("NameService"));

                    //*********************************************************************
// Cria os Objeos do Gateway no Dominio Corba a partir dos Objetos Java
                    //*********************************************************************

          tc_pdu_provider_factory = orbFiles.TcSignaling.TcPduProviderFactoryHelper.narrow(
                              poa.servant_to_reference( new impl.TcPduProviderFactoryImpl() ));
                              // Não implementado

          gt_root = NamingContextExtHelper.narrow(
                              nc_CORBA.bind_new_context(nc_CORBA.to_name("GT.null")) );

          tc_factory_finder = orbFiles.TcSignaling.TcFactoryFinderHelper.narrow(
                              poa.servant_to_reference( new impl.TcFactoryFinderImpl(gt_root) ));

          gw_admin = orbFiles.TcSignaling.GwAdminHelper.narrow(
                              poa.servant_to_reference( new impl.GwAdminImpl(tc_factory_finder,
                              tc_pdu_provider_factory) ));

          tc_repository = orbFiles.TcSignaling.TcRepositoryHelper.narrow(
                              poa.servant_to_reference( new impl.TcRepositoryImpl() ));

           tc_service_finder = orbFiles.TcSignaling.TcServiceFinderHelper.narrow(
                              poa.servant_to_reference( new impl.TcServiceFinderImpl(gt_root,
                              gw_admin, tc_repository) ));



          impl.SSP_SCP_ResponderImpl ssp_scp_responder = new
                    impl.SSP_SCP_ResponderImpl(3);
          impl.SSP_SCP_InitiatorImpl ssp_scp_initiator = new impl.SSP_SCP_InitiatorImpl(3);

          org.omg.CORBA.Object obj2 = poa.servant_to_reference(ssp_scp_responder);
          org.omg.CORBA.Object obj3 = poa.servant_to_reference(ssp_scp_initiator);

          PrintWriter pw = new PrintWriter( new FileWriter( "Arquivo.ref" ));

          // print stringified objects references to file
          pw.println( orb.object_to_string( tc_repository ));
          pw.println( orb.object_to_string( obj2 ));
          pw.println( orb.object_to_string( obj3 ));

          pw.flush();
          pw.close();


          // Liga os objetos do Gateway ao Serviço de Nomes
          nc_CORBA.bind( nc_CORBA.to_name("GwAdmin.if"), gw_admin );
          nc_CORBA.bind( nc_CORBA.to_name("TcServiceFinder.if"), tc_service_finder );



                    // Liga a interface "TcUserFactory" ao "gt_root" do contexto de nomes

                    tc_user_factory = orbFiles.TcSignaling.TcUserGenericFactoryHelper.narrow(
```

```
                                poa.servant_to_reference( new impl.TcUserFactoryImpl(orb, poa) ));

                        addr = "GT_1";
                        a_c  = "DefAc";
                        resp_iface = "SCP_SSP";
                        init_iface = "SSP_SCP";
                        tc_factory_finder.bind(addr, a_c, resp_iface, init_iface, tc_user_factory);


                        // Executa o ORB
                        orb.run();
                    }
    catch ( Exception e ) {
      e.printStackTrace();
    }
  }
}
```

A1.2 Codigo do Cliente

```
/*
 * Client.java
 *
 */

package impl;

import org.omg.CosNaming.*;
import java.io.*;
import orbFiles.TcSignaling.*;
import orbFiles.TC_User.*;



public class Client {

  public static void main( String[] args ) {

    String iface;
                IdValue code = new IdValue();
                int param;
                String oper_name;
                String oper_result;
                iface = "SSF_SCF_responder";
                code.local_id(3);
                param = 10;

    try {

      File f = new File( "Arquivo.ref" );

                        //check if file exists
                        if( ! f.exists() ){
```

```
                System.out.println("File " + args[0] + " does not exist.");
                System.exit( -1 );
        }

        //check if args[0] points to a directory
        if( f.isDirectory() ){
                System.out.println("File " + args[0] + " is a directory.");
                System.exit( -1 );
        }

 // Objetos do Gateway
 orbFiles.TcSignaling.TcUserGenericFactory tc_user_factory;

orbFiles.TcSignaling.TcUser tc_user_initiator;

orbFiles.TcSignaling.TcUser tc_user_responder;

orbFiles.TC_User.SSP_SCP_Responder ssp_scp_responder;

orbFiles.TC_User.SSP_SCP_Initiator ssp_scp_initiator;

orbFiles.TcSignaling.TcRepository tc_repository;

orbFiles.TcSignaling.TcFactoryFinder tc_factory_finder;

orbFiles.TcSignaling.GwAdmin gw_admin;


 // Variáveis e Constantes
String addr = new String("GT_1");
String a_c  = new String("DefAc");

// Parâmetros de saída "out" da operação "tc_factory_finder.resolve()"
org.omg.CORBA.StringHolder init_iface = new org.omg.CORBA.StringHolder();
org.omg.CORBA.StringHolder resp_iface = new org.omg.CORBA.StringHolder();

// Identificação da associação ID para iniciar uma associação
String a_id = new String ("Initiator");

// Defines the TC context information to be used in the association
short tc_context_setting = orbFiles.TcSignaling.TC_CONTEXT_BASE.value;

// Parametetro para operações do TcUser
//int param;

// Inicializa o ORB
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);

// Obtém a referência para o POA
org.omg.PortableServer.POA poa = org.omg.PortableServer.POAHelper.narrow(
        orb.resolve_initial_references("RootPOA"));

// Ativa o POA
poa.the_POAManager().activate();



// Lendo arquivo TcRepository.txt
BufferedReader br = new BufferedReader( new FileReader( f ));
org.omg.CORBA.Object obj1 = orb.string_to_object( br.readLine() );
```

```
                br.close();

                // Obtém a referência para o Serviço de Nomes
                NamingContextExt nc_CORBA = NamingContextExtHelper.narrow(
                        orb.resolve_initial_references("NameService"));


// ************************************************** //
//                                                    //
//   BEGINNING OF INTELLIGENT NETWORK (IN) APPLICATION   //
//                                                    //
// ************************************************** //

System.out.println("\n\n\nBEGINNING OF INTELLIGENT NETWORK (IN) APPLICATION\n");

// Obtém a referencia para a interface GwAdmin do Serviço de Nomes
gw_admin = orbFiles.TcSignaling.GwAdminHelper.narrow(
                nc_CORBA.resolve( nc_CORBA.to_name("GwAdmin.if") ));

tc_factory_finder = gw_admin.tc_user_factory_naming_if();
                System.out.println("TcFactoryFinder obtained from GwAdmin\n");

tc_user_factory = tc_factory_finder.resolve(addr, a_c, resp_iface, init_iface);

System.out.println("TcUserFactory resolved from TcFactoryFinder\n");

tc_user_initiator = orbFiles.TcSignaling.TcUserHelper.narrow(
                poa.servant_to_reference( new impl.TcUserImpl(3)));

tc_user_responder = orbFiles.TcSignaling.TcUserHelper.narrow(
                poa.servant_to_reference( new impl.TcUserImpl(3)));

ssp_scp_responder = orbFiles.TC_User.SSP_SCP_ResponderHelper.narrow(
                poa.servant_to_reference( new impl.SSP_SCP_ResponderImpl(3)));

ssp_scp_initiator = orbFiles.TC_User.SSP_SCP_InitiatorHelper.narrow(
                poa.servant_to_reference( new impl.SSP_SCP_InitiatorImpl(3)));

tc_repository = TcRepositoryHelper.narrow( obj1 );

ssp_scp_initiator.scoped_name(a_id);

// Informa ao SCP que o diálogo foi iniciado
System.out.println("\nInforming the SCP object that a dialog was initiated...\n");
ssp_scp_responder.new_association(ssp_scp_initiator.scoped_name());

// Verifica se o SCP tem um diálogo com o "Object initiator"
    System.out.println("\nVerifying if the SCP has a dialog with 'Object initiator'...\n");
    if (ssp_scp_responder.find_a_id(ssp_scp_initiator.scoped_name()) == 1)
                System.out.println("   SCP has a dialog with 'Object initiator'!\n");

// Obtém o nome da operação
    oper_name = tc_repository.get_operation_name(iface,code);

    // Chama a operação
    oper_result = ssp_scp_responder.oper(oper_name, ssp_scp_initiator.scoped_name());

    // Imprime o resultado
    System.out.println("\n" + oper_result + "\n");
```

```
            // Finaliza o diálogo com o SCP
            ssp_scp_responder.end_association(ssp_scp_initiator.scoped_name());

            // Verifica se o diálogo com o "Object initiator"foi terminado
            System.out.println("\nVerifying if the dialog with 'Object initiator' was ended...\n");
            if (ssp_scp_responder.find_end(ssp_scp_initiator.scoped_name()) == 1)
                        System.out.println("   The dialog with 'Object initiator' was ended!\n");


        System.out.println("END OF INTELLIGENT NETWORK (IN) APPLICATION\n\n");
        // ****************************************************** //
        //                                                        //
        //     END OF INTELLIGENT NETWORK (IN) APPLICATION      //
        //                                                        //
        // ****************************************************** //

        // Finaliza o ORB
        orb.shutdown(true);
      }
      catch ( Exception e ) {
        e.printStackTrace();
      }
    }

}
```

Apêndice 2: Código Java do Cliente modificado com o AMI para execução no Orbacus

A2.1 Codigo do Cliente

```
/*
 * Client.java
 *
 */

package impl;

import org.omg.CosNaming.*;
import java.io.*;
import orbFiles.TcSignaling.*;
import orbFiles.TC_User.*;



public class Client {

  public static void main( String[] args ) {

    String iface;
                IdValue code = new IdValue();
                int param;
                String oper_name;
                String oper_result;
                iface = "SSF_SCF_responder";
                code.local_id(3);
                param = 10;

    try {

      File f = new File( "Arquivo.ref" );

                        //check if file exists
                        if( ! f.exists() ){
                                System.out.println("File " + args[0] + " does not exist.");
                                System.exit( -1 );
                        }

                        //check if args[0] points to a directory
                        if( f.isDirectory() ){
                                System.out.println("File " + args[0] + " is a directory.");
                                System.exit( -1 );
                        }

                // Objetos do Gateway
                orbFiles.TcSignaling.TcUserGenericFactory tc_user_factory;
```

```
orbFiles.TcSignaling.TcUser tc_user_initiator;

orbFiles.TcSignaling.TcUser tc_user_responder;

orbFiles.TC_User.SSP_SCP_Responder ssp_scp_responder;

orbFiles.TC_User.SSP_SCP_Initiator ssp_scp_initiator;

orbFiles.TcSignaling.TcRepository tc_repository;

orbFiles.TcSignaling.TcFactoryFinder tc_factory_finder;

orbFiles.TcSignaling.GwAdmin gw_admin;


 // Variáveis e Constantes
String addr = new String("GT_1");
String a_c  = new String("DefAc");

// Parâmetros de saída "out" da operação "tc_factory_finder.resolve()"
org.omg.CORBA.StringHolder init_iface = new org.omg.CORBA.StringHolder();
org.omg.CORBA.StringHolder resp_iface = new org.omg.CORBA.StringHolder();

// Identificação da associação ID para iniciar uma associação
String a_id = new String ("Initiator");

// Defines the TC context information to be used in the association
short tc_context_setting = orbFiles.TcSignaling.TC_CONTEXT_BASE.value;

// Parametetro para operações do TcUser
//int param;

// Inicializa o ORB
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);

// Obtém a referência para o POA
org.omg.PortableServer.POA poa = org.omg.PortableServer.POAHelper.narrow(
        orb.resolve_initial_references("RootPOA"));

// Ativa o POA
poa.the_POAManager().activate();



// Lendo arquivo TcRepository.txt
BufferedReader br = new BufferedReader( new FileReader( f ));
org.omg.CORBA.Object obj1 = orb.string_to_object( br.readLine() );
br.close();

 // Obtém a referência para o Serviço de Nomes
 NamingContextExt nc_CORBA = NamingContextExtHelper.narrow(
        orb.resolve_initial_references("NameService"));


// ************************************************** //
//                                                    //
//   BEGINNING OF INTELLIGENT NETWORK (IN) APPLICATION   //
//                                                    //
// ************************************************** //
```

```
System.out.println("\n\n\nBEGINNING OF INTELLIGENT NETWORK (IN) APPLICATION\n");

    // Obtém a referencia para a interface GwAdmin do Serviço de Nomes
    gw_admin = orbFiles.TcSignaling.GwAdminHelper.narrow(
            nc_CORBA.resolve( nc_CORBA.to_name("GwAdmin.if") ));

    tc_factory_finder = gw_admin.tc_user_factory_naming_if();
            System.out.println("TcFactoryFinder obtained from GwAdmin\n");

    tc_user_factory = tc_factory_finder.resolve(addr, a_c, resp_iface, init_iface);

    System.out.println("TcUserFactory resolved from TcFactoryFinder\n");

    tc_user_initiator = orbFiles.TcSignaling.TcUserHelper.narrow(
            poa.servant_to_reference( new impl.TcUserImpl(3)));

    tc_user_responder = orbFiles.TcSignaling.TcUserHelper.narrow(
            poa.servant_to_reference( new impl.TcUserImpl(3)));

    ssp_scp_responder = orbFiles.TC_User.SSP_SCP_ResponderHelper.narrow(
             poa.servant_to_reference( new impl.SSP_SCP_ResponderImpl(3)));

    ssp_scp_initiator = orbFiles.TC_User.SSP_SCP_InitiatorHelper.narrow(
             poa.servant_to_reference( new impl.SSP_SCP_InitiatorImpl(3)));

    tc_repository = TcRepositoryHelper.narrow( obj1 );

    ssp_scp_initiator.scoped_name(a_id);

    // Informa ao SCP que o diálogo foi iniciado
    System.out.println("\nInforming the SCP object that a dialog was initiated...\n");
    ssp_scp_responder.new_association(ssp_scp_initiator.scoped_name());

    // Verifica se o SCP tem um diálogo com o "Object initiator"
    System.out.println("\nVerifying if the SCP has a dialog with 'Object initiator'...\n");
    if (ssp_scp_responder.find_a_id(ssp_scp_initiator.scoped_name()) == 1)
            System.out.println("   SCP has a dialog with 'Object initiator'!\n");

    // Obtém o nome da operação
    oper_name = tc_repository.get_operation_name(iface,code);

    //Modificação para requisições assíncronas
    AsyncHandler asyncHandler = new AsyncHandler(poa);
    orbFiles.TC_User.AMI_InitiatorHandler handler = asyncHandler._this(orb);

    // Chama a operação
    oper_result = ssp_scp_responder. sendc_oper (handler, oper_name,
                    ssp_scp_initiator.scoped_name());

    // Imprime o resultado
    System.out.println("\n" + oper_result + "\n");

     // Finaliza o diálogo com o SCP
    ssp_scp_responder.end_association(ssp_scp_initiator.scoped_name());

    // Verifica se o diálogo com o "Object initiator”foi terminado
    System.out.println("\nVerifying if the dialog with 'Object initiator' was ended...\n");
    if (ssp_scp_responder.find_end(ssp_scp_initiator.scoped_name()) == 1)
            System.out.println("   The dialog with 'Object initiator' was ended!\n");
```

```
        System.out.println("END OF INTELLIGENT NETWORK (IN) APPLICATION\n\n");
        // ************************************************** //
        //                                                    //
        //     END OF INTELLIGENT NETWORK (IN) APPLICATION     //
        //                                                    //
        // ************************************************** //

        // Finaliza o ORB
        orb.shutdown(true);
      }
      catch ( Exception e ) {
        e.printStackTrace();
      }
    }

}
```

Apêndice 3: Código Java do Servidor e Cliente modificados com o BC para execução no Orbix

A3.1 Codigo do Servidor

```
/*
 * Server.java
 *
 */

package impl;

import org.omg.CosNaming.*;
import java.io.*;
import TcSignaling.*;
import TC_User.*;
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.PortableServer.POAManagerPackage.*;

import IT_LoadBalancing.*;
import IT_Naming.*;
import com.iona.corba.trading.console.util.Parse;


/**
 * This class emulates the action of a TC/CORBA Gateway
 */
public class Server {

//************************************************************************
//
//                                    P R O C E D U R E  P R I N C I P A L
//
//************************************************************************

        public static void main( String args[] )
        {

//************************************************************************
//                                    INICIANDO VARIAVEIS E GRUPOS DO BC
//************************************************************************

                int maxMembers = 5;

                String[] _id = new String[ maxMembers * 2 ];
                Member[] _member = new Member[ maxMembers * 2 ];
                String _server_name = args[0];

                org.omg.CORBA.Object[] objssp = new org.omg.CORBA.Object[ maxMembers ];
```

```
                SSP_SCP_InitiatorImpl[] _ssp = new SSP_SCP_InitiatorImpl[maxMembers] ;

                org.omg.CORBA.Object[] objscp = new org.omg.CORBA.Object[ maxMembers ];
                SSP_SCP_ResponderImpl[] _scp = new SSP_SCP_ResponderImpl[maxMembers] ;


                int j = 0;

                for (int i=0; i<=((maxMembers*2)-1) ; i++)
                {
                        _id[i] = new String (_server_name + ":RR_Member" + (i+1) );
                }

                for (int i=0; i<=(maxMembers-1) ; i++)
                {
                        objssp[i] = null;
                        _ssp[i] = null;

                        objscp[i] = null;
                        _scp[i] = null;
                }

//*************************************************************************
//                                      CARREGANDO ATRIBUTOS PRIVADOS
//*************************************************************************

                System.out.println("Carregando atributos privados...");

                try
                {
                        java.util.Properties props = System.getProperties();

                        props.put("org.omg.CORBA.ORBClass", "com.iona.corba.art.artimpl.ORBImpl");
                        props.put("org.omg.CORBA.ORBSingletonClass",
"com.iona.corba.art.artimpl.ORBSingleton");

                        org.omg.CORBA.ORB global_orb = null;

                        try
                        {
                                System.out.println("Inicializando o ORB");

                                global_orb = org.omg.CORBA.ORB.init(args, props);

                        }
                        catch (Exception ex)
                        {
                                System.out.println("Não foi possível inicializar o ORB.");
                                System.out.println("Erro: " + ex.toString());
                                System.exit(1);
                        }


                        NamingContextExt gt_root = null;
                        NamingContextExt _nameService = null;
                        IT_NamingContextExt _nameServiceReferenceBC1 = null;
                        IT_NamingContextExt _nameServiceReferenceBC2 = null;
                        ObjectGroupFactory _objectGroupFactory1 =null;
                        ObjectGroupFactory _objectGroupFactory2 =null;
                        CosNaming.NameComponent[] nm1 = new CosNaming.NameComponent[1];
```

```
                        CosNaming.NameComponent[] nm2 = new CosNaming.NameComponent[1];

                        try
                        {
                                org.omg.CORBA.Object _rootNameServiceObject =
global_orb.resolve_initial_references("NameService");

                                _nameService = org.omg.CosNaming.NamingContextExtHelper.narrow(
_rootNameServiceObject );

                                try
                                {
                                        gt_root = NamingContextExtHelper.narrow(

        _nameService.bind_new_context(_nameService.to_name("GT")) );
                                }
                                catch (Exception E)
                                {
                                        _nameService.unbind(_nameService.to_name("GT"));
                                        gt_root = NamingContextExtHelper.narrow(
                                        _nameService.bind_new_context(_nameService.to_name("GT")) );
                                }

                                System.out.println("Obtendo referencia para o root do BC");

                                _nameServiceReferenceBC1 =
IT_Naming.IT_NamingContextExtHelper.narrow(_rootNameServiceObject);
                                _nameServiceReferenceBC2 =
IT_Naming.IT_NamingContextExtHelper.narrow(_rootNameServiceObject);

                                System.out.println("Obtendo referencia para a Fabrica de Grupos de
Objetos");

                                _objectGroupFactory1 = _nameServiceReferenceBC1.og_factory();
                                _objectGroupFactory2 = _nameServiceReferenceBC2.og_factory();

                                nm1 = _nameServiceReferenceBC1.to_name("GW1");
                                nm2 = _nameServiceReferenceBC2.to_name("GW2");
                        }
                        catch (Exception ex)
                        {
                                System.out.println("Não foi possivel obter ou referenciar O Servico de
Nomes para IT_Naming." +
                                                "interface IT_NamingContextExt. Verifique se o Serico de
Nomes está rodando?");
                                System.out.println("Erro: " + ex.toString());
                                System.exit(1);
                        }

//*************************************************************************
//                                CRIANDO GRUPOS DE OBJETOS PARA O BC
//*************************************************************************

                        ObjectGroup _objectGroup1 = null;
                        String _objectGroupName1 = null;

                        ObjectGroup _objectGroup2 = null;
                        String _objectGroupName2 = null;

                        try
```

```
                        {
                                System.out.println("Criando Grupos de Objetos (Método Active)");
                                _objectGroupName1 = new String("GW_Group1");
                                _objectGroupName2 = new String("GW_Group2");

                                _objectGroup1 = _objectGroupFactory1.create_active( _objectGroupName1
);
                                _objectGroup2 = _objectGroupFactory2.create_active( _objectGroupName2
);

                                System.out.println("Executando 'Binding' nos Grupos de Objetos");
                                _nameServiceReferenceBC1.bind_object_group(nm1, _objectGroup1);
                                _nameServiceReferenceBC2.bind_object_group(nm2, _objectGroup2);
                        }
                        catch(Exception ex)
                        {
                                System.out.println("Erro: " + ex.toString());

                                try
                                {
                                        _objectGroup1 =
_objectGroupFactory1.find_group(_objectGroupName1);
                                        _objectGroup2 =
_objectGroupFactory2.find_group(_objectGroupName2);
                                }
                                catch (NoSuchGroup nsg)
                                {
                                        System.out.println("Erro ao tentar localizar Grupo de Objetos");
                                        System.out.println("Erro: " + nsg.toString());
//                                      System.exit(1);
                                }
                        }

//**************************************************************************
//                      O SERVIDOR PODE SER EXECUTADO PARA LIMPAR OS GRUPOS
//                      CRIADOS EM OUTRA EXECUÇÃO. UTILIZANDO O PARAMETRO "clean"
//                      NO CHAMADO DO SERVIDOR
//**************************************************************************

                        if( args[1].equals("clean") )
                        {
                                try
                                {
                                        _objectGroup1 = _objectGroupFactory1.find_group(
_objectGroupName1 );
                                        for (int i=0; i<=(maxMembers-1) ; i++)
                                        {
                                                System.out.println("Removendo objeto id: " + _id[i]);
                                                _objectGroup1.remove_member( _id[i] );
                                                j = i;
                                        }
                                }
                                catch (Exception ex)
                                {
                                        System.out.println("Não pode remover membros do Grupo1.
"+ex.toString());
                                }

                                try
                                {
```

```
                    _objectGroup2 = _objectGroupFactory2.find_group(
_objectGroupName2 );
                    for (int i=j+1; i<=((maxMembers*2)-1) ; i++)
                    {
                        System.out.println("Removendo objeto id: " + _id[i]);
                        _objectGroup2.remove_member( _id[i] );
                    }
                }
                catch (Exception ex)
                {
                    System.out.println("Não pode remover membros do Grupo2.
"+ex.toString());
                }

                if (_objectGroup1 != null)
                {
                    String[] members = _objectGroup1.members();
                    if (members.length == 0)  // Last one to remove members
                    {
                        try
                        {
                            System.out.println("Executando 'Unbinding' nos
Grupos de Objetos...");
                            _nameServiceReferenceBC1.unbind( nm1 );
                            System.out.println("Destruindo Grupos de
Objetos...");
                            _objectGroup1.destroy();
                        }
                        catch (Exception ex)
                        {
                            System.out.println("Falha ao Executar
'Unbind/destroy'.");
                        }
                    }
                }

                if (_objectGroup2 != null)
                {
                    String[] members = _objectGroup2.members();
                    if (members.length == 0)  // Last one to remove members
                    {
                        try
                        {
                            System.out.println("Executando 'Unbinding' em
Grupo de Objeto...");
                            _nameServiceReferenceBC2.unbind(nm2);
                            System.out.println("Desttruindo Grupo...");
                            _objectGroup2.destroy();
                        }
                        catch (Exception ex)
                        {
                            System.out.println("Falha ao executar
'Unbind/destroy'.");
                        }
                    }
                }

                System.out.println("Fechando ORB.");
                global_orb.shutdown(false);
                System.exit(0);
```

```
                        }


//*************************************************************************
//                      IMPLEMENTANDO O Portable Object Active
//*************************************************************************

                        org.omg.PortableServer.POA _poa = null;
                        try
                        {
                                _poa = org.omg.PortableServer.POAHelper.narrow(
                                        global_orb.resolve_initial_references("RootPOA"));

                                _poa.the_POAManager().activate();
                        }
                        catch (Exception ex)
                        {
                                System.out.println("Não foi possível obter o 'RootPOA' ou o
'POAManager'");
                                System.out.println("Erro: " + ex.toString());
                                System.exit(1);
                        }
                        // Fim InicalizarPOA



//*************************************************************************
// CRIANDO OS OBJETOS DO GATEWAY NO DOMINIO CORBA A PARTIR DOS OBJETOS JAVA
//*************************************************************************

                        TcSignaling.TcPduProviderFactory tc_pdu_provider_factory;

                        tc_pdu_provider_factory = TcSignaling.TcPduProviderFactoryHelper.narrow(
                                _poa.servant_to_reference(new impl.TcPduProviderFactoryImpl()) );

                        TcSignaling.TcFactoryFinder tc_factory_finder;

                        tc_factory_finder = TcSignaling.TcFactoryFinderHelper.narrow(
                                _poa.servant_to_reference( new impl.TcFactoryFinderImpl(gt_root) ));

                        TcSignaling.TcRepository tc_repository;

                        tc_repository = TcSignaling.TcRepositoryHelper.narrow(
                                _poa.servant_to_reference( new impl.TcRepositoryImpl() ));

                        TcSignaling.GwAdmin gw_admin;

                        gw_admin = TcSignaling.GwAdminHelper.narrow(
                                _poa.servant_to_reference( new impl.GwAdminImpl(tc_factory_finder,
tc_pdu_provider_factory) ));

                        TcSignaling.TcServiceFinder tc_service_finder;

                        tc_service_finder = TcSignaling.TcServiceFinderHelper.narrow(
                                _poa.servant_to_reference( new impl.TcServiceFinderImpl(gt_root,
gw_admin, tc_repository) ));

//*************************************************************************
// "BINDS" OBJETOS DO GATEWAY PARA O SERVICO DE NOMES
//*************************************************************************
```

```
                        gt_root.bind( gt_root.to_name("GwAdmin.if"), gw_admin );

                        gt_root.bind( gt_root.to_name("TcServiceFinder.if"), tc_service_finder );

                        TcSignaling.TcUserGenericFactory tc_user_factory;

                        tc_user_factory = TcSignaling.TcUserGenericFactoryHelper.narrow(
                                _poa.servant_to_reference( new impl.TcUserFactoryImpl(global_orb, _poa)
));

                        if (tc_user_factory==null) {System.out.println("tc_user_factory vazio");}

                        String addr = new String();
                        String a_c  = new String();
                        String resp_iface = new String();
                        String init_iface = new String();

                        addr = "GT_1";          // Holds the Global Title of the Factory
                        a_c  = "DefAc";         // CONSTANT for this Gateway implementation
                        resp_iface = "SSP_SCP"; // Type I definition: the values of both
                        init_iface = "SSP_SSP"; // "resp_iface" and "init_iface" are the same
                        tc_factory_finder.bind(addr, a_c, resp_iface, init_iface, tc_user_factory);

        // 2nd Global Title: same TcUserFactory as before, but with
                        // different ScopedNames

                        addr = "GT_2";              // Holds the Global Title of the Factory
        a_c  = "DefAc";             // CONSTANT for this Gateway implementation
        resp_iface = "SSP_SCP_Extra"; // Type I definition: the values of both
        init_iface = "SSP_SSP_Extra"; // "resp_iface" and "init_iface" are the same
        tc_factory_finder.bind(addr, a_c, resp_iface, init_iface, tc_user_factory);

                        PrintWriter pw = new PrintWriter( new FileWriter( ".\\Arquivo.ref" ));

                        pw.println( global_orb.object_to_string( tc_repository ));
                        pw.flush();
                        pw.close();


//*************************************************************************
// ADICIONANDO MEMBROS AOS GRUPOS DE OBJETOS DO BC
//*************************************************************************

                        try
                        {
                                System.out.println("Adicionando membros ao Grupo de Objetos");

                                for (int i=0; i<=(maxMembers - 1) ; i++)
                                {
                                        _ssp[i] = new SSP_SCP_InitiatorImpl( 3 );
                                        _ssp[i].scoped_name( _id[i] );
                                        objssp[i] = null;

                                        try
                                        {
                                                objssp[i] = _poa.servant_to_reference( _ssp[i] );
                                        }
                                        catch (org.omg.PortableServer.POAPackage.WrongPolicy ex)
                                        {
```

```
                                        System.out.println("Não pode ativar o servant_poa.");
                                        System.out.println("E: " + ex.toString());
                                        System.exit(1);
                                }
                                _member[i] = new Member(_ssp[i]._this_object(), _id[i] );
                                _objectGroup1.add_member( _member[i] );
                        }

                        for (int i=0; i<=(maxMembers - 1) ; i++)
                        {
                                j = (i + maxMembers);
                                _scp[i] = new SSP_SCP_ResponderImpl( 3 );
                                _scp[i].scoped_name( _id[j] );
                                objscp[i] = null;

                                try
                                {
                                        objscp[i] = _poa.servant_to_reference( _scp[i] );
                                }
                                catch (org.omg.PortableServer.POAPackage.WrongPolicy ex)
                                {
                                        System.out.println("Não pode ativar o servant_poa.");
                                        System.out.println("E: " + ex.toString());
                                        System.exit(1);
                                }

                                _member[i] = new Member(_scp[i]._this_object(), _id[j] );
                                _objectGroup2.add_member( _member[i] );
                        }
                }
                catch (IT_LoadBalancing.DuplicateMember ex)
                {
                        System.out.println("Um Servidor com este nome já foi criado.");
                        System.out.println("Se o Servidor ainda estiver sendo executado então utilize
um nome diferente.");
                        System.out.println("Se o servidor não estiver sendo executado então remova
a Referencia no Servico de Nomes.");
                        System.out.println("Para remover utilize: ");
                        System.out.println("   java -classpath ./java/classes:$CLASSPATH
load_balancing.Server <server_name> clean");
                        System.exit(1);
                }
                catch (Exception ex1)
                {
                        System.out.println("E:" + ex1.toString());
                        System.exit(1);
                }

//*************************************************************************
// EXECUTAR O SERVIDOR E AGUARDAR AS REQUISIÇÕES DOS CLIENTES
//*************************************************************************
                try
                {
                        System.out.println("Servidor pronto e aguardando...");
                        global_orb.run();
                }
                catch (Exception ex)
                {
                        System.out.println("Não foi possível ativar POAManager, ou falha no
orb.run().");
```

```
                                System.out.println("Erro: " + ex.toString());
                                System.exit(1);
                        }

                        System.exit(0);
                }

                catch ( Exception e )
                {
                        e.printStackTrace();
                }
        }
}
```

A3.2 Codigo do Cliente

```
package impl;

import org.omg.CosNaming.*;
import java.io.*;
import TcSignaling.*;
import TC_User.*;
import java.util.*;
import org.omg.CORBA.*;
import CosNaming.NamingContextPackage.*;

public class Client {

        static ORB global_orb;

  public static void main( String args[] )
                throws org.omg.CosNaming.NamingContextPackage.InvalidName,
org.omg.CosNaming.NamingContextPackage.NotFound,
org.omg.CosNaming.NamingContextPackage.CannotProceed
        {

                if (args.length < 1)
                {
                        System.out.println("Usage: java  impl.Client <stock_symbol>");
                        return;
                }

                java.util.Properties props = System.getProperties();
                props.put("org.omg.CORBA.ORBClass", "com.iona.corba.art.artimpl.ORBImpl");
                props.put("org.omg.CORBA.ORBSingletonClass",
"com.iona.corba.art.artimpl.ORBSingleton");

                try
                {
                        System.out.println("Iniciando o ORB.");
                        global_orb = ORB.init(args, props);
                }
                catch (SystemException ex)
                {
```

```
                    System.out.println("Exceção Inesperada durando a Inicialização do ORB" );
                    System.out.println(ex);
                    System.exit(1);
                }

org.omg.CORBA.Object _rootNameServiceObject = null;
                NamingContextExt _nameService = null;
                NamingContextExt gt_root = null;

                org.omg.CosNaming.NameComponent[] nm;

                try
                {
                    _rootNameServiceObject = global_orb.resolve_initial_references("NameService");
                    _nameService = NamingContextExtHelper.narrow( _rootNameServiceObject );

                    nm = _nameService.to_name("GT");

                    gt_root = NamingContextExtHelper.narrow( _nameService.resolve_str("GT") );
}
catch (org.omg.CORBA.ORBPackage.InvalidName ex)
{
  System.out.println("Exceção inesperada durante 'resolve_initial_references()':");
  System.out.println(ex);
  System.exit(1);
}
catch (SystemException ex)
{
  System.out.println("Exceção inesperada durante 'resolve_initial_references()':");
  System.out.println(ex);
  System.exit(1);
}

String iface;
                IdValue code = new IdValue();
                int param;
                String oper_name;
                String oper_result;
                iface = "SSF_SCF_responder";
                code.l_id(3);
                param = 10;

try {

  TcSignaling.TcUserGenericFactory tc_user_factory;

  TcSignaling.TcUser tc_user_initiator;

  TcSignaling.TcUser tc_user_responder;

                    TC_User.SSP_SCP_Responder ssp_scp_responder;

  TC_User.SSP_SCP_Initiator ssp_scp_initiator;

                    TcSignaling.TcRepository tc_repository;

  TcSignaling.TcFactoryFinder tc_factory_finder;

  TcSignaling.GwAdmin gw_admin;
```

```
        String addr = new String("GT_1");
        String a_c  = new String("DefAc");

                        org.omg.CORBA.StringHolder init_iface = new org.omg.CORBA.StringHolder();
        org.omg.CORBA.StringHolder resp_iface = new org.omg.CORBA.StringHolder();

        short tc_context_setting = TcSignaling.TC_CONTEXT_BASE.value;

        org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);

                        org.omg.PortableServer.POA poa = org.omg.PortableServer.POAHelper.narrow(
                        orb.resolve_initial_references("RootPOA"));

                        poa.the_POAManager().activate();

         File f = new File( ".\\Arquivo.ref" );
        BufferedReader br = new BufferedReader( new FileReader( f ));
                        org.omg.CORBA.Object objTcRepository = orb.string_to_object( br.readLine() );

                        br.close();

// ********************************************************************************
//   INICIA A REDE INTELIGENTE (NA APLICAÇÃO)
// ********************************************************************************
        System.out.println("\n\n\nINICIA A APLICAÇÃO DE REDE INTELIGENTE \n");

        gw_admin = TcSignaling.GwAdminHelper.narrow(
          gt_root.resolve( gt_root.to_name("GwAdmin.if") ));

        tc_factory_finder = gw_admin.tc_user_factory_naming_if();
        System.out.println("TcFactoryFinder obtido a partir do GwAdmin\n");

        tc_user_factory = tc_factory_finder.resolve(addr, a_c, resp_iface, init_iface);
        System.out.println("TcUserFactory resolvido do TcFactoryFinder\n");

                        tc_repository = TcSignaling.TcRepositoryHelper.narrow(
                                poa.servant_to_reference( new impl.TcRepositoryImpl() ));

                        org.omg.CORBA.Object obj_ssp = null;
                        org.omg.CORBA.Object obj_scp = null;
                        try
                        {
                                obj_ssp = _nameService.resolve_str( "GW1" );
                                obj_scp = _nameService.resolve_str( "GW2" );
                        }
                        catch (SystemException ex)
                        {
                                System.out.println("Não foi possível resolver os grupos do BC.");
                                System.out.println(ex);
                                System.exit(1);
                        }
                        catch (Exception ex)
                        {
                                System.out.println("Não foi possível resolver os grupos do BC.");
                                System.out.println("Exceção desconhecida pela aplicação.");
                                System.out.println(ex);
                                System.exit(1);
                        }

                        ssp_scp_initiator = TC_User.SSP_SCP_InitiatorHelper.narrow( obj_ssp );
```

```
                        ssp_scp_responder = TC_User.SSP_SCP_ResponderHelper.narrow( obj_scp );


                        System.out.println("********* Informando o object SCP que o diálogo foi
iniciado...");
                        System.out.println(ssp_scp_initiator.scoped_name());
                        System.out.println(ssp_scp_responder.scoped_name());

                        ssp_scp_responder.new_association( ssp_scp_initiator.scoped_name() );

                        System.out.println("Verificando se o SCP teve um diálogo com o 'Objeto
Iniciador'...");

                        if (ssp_scp_responder.find_a_id(ssp_scp_initiator.scoped_name()) == 1)
                                System.out.println("==> SCP teve um diálogo com o 'Objeto Iniciador'!");

                        oper_name = tc_repository.get_operation_name(iface,code);

                        oper_result = ssp_scp_responder.oper(oper_name, ssp_scp_initiator.scoped_name());

                        System.out.println( oper_result );


                        ssp_scp_responder.end_association( ssp_scp_initiator.scoped_name() );

                        System.out.println("Verificando se o diálogo com 'Objeto Iniciador' foi finalizado...");

                        if (ssp_scp_responder.find_end(ssp_scp_initiator.scoped_name()) == 1)
                                System.out.println("==> O Diálogo com o 'Objeto Iniciador' foi
finalizado!");


//                      System.out.println("Aguardando teste com outros clientes. Depois tecle <Enter>.");
//                      try
//                      {
//                              System.in.read();
//                      }
//                      catch (java.io.IOException e)
//                      {
//                              e.printStackTrace();
//                      }

                        System.out.println("FIM DA APLICAÇÃO DA REDE INTELIGENTE");
// ***************************************************************************
//
//     FIM DA APLICAÇÃO DA REDE INTELIGENTE
//
// ***************************************************************************

        orb.shutdown(true);
    }
    catch ( Exception e ) {
      e.printStackTrace();
    }
  }
}
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo