Gerenciamento e Monitoração de Recursos Distribuídos para um Portal de Voz

Denis Hipólito de Araujo

Dissertação de Mestrado submetida à Coordenação dos Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Mestre no domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação - Engenharia da Computação

Angelo Perkusich, D.Sc.

Campina Grande, Paraíba, Brasil

A663g 2003

Araújo Denis Hipólito de

Gerenciamento e monitoração de recursos distribuídos para um portal de vozdelo / Denis Hipólito de Araujo - Campina Grande - UFCG, 2003

90 p.: il.

Inclui Bibliografia

Dissertação (Mestradoem Engenharia Elétrica)UFCG/CCT/DEE.

1.Sistemas Distribuidos 2. Sistemas Embutidos

3. Portal de Voz

I. Titula

CDU: 681.3.o66

GERENCIAMENTO E MONITORAÇÃO DE RECURSOS DISTRIBUÍDOS PARA UM PORTAL DE VOZ

DENIS HIPÓLITO DE ARAÚJO

Dissertação Aprovada em 29.12.2003

ANGELO PERKUSICH, D.Sc., UFCG
Orientador

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Componente da Banca

JOSÉ SÉRGIO DA ROCHA NETO, D.Sc., UFCG
Componente da Banca

CAMPINA GRANDE - PB
Dezembro - 2003

Dedicatória

Dedico este trabalho a meus pais Diniz e Dalva que durante todos estes anos sempre me apoiaram e incentivaram.

Agradecimentos

- A Deus, por tudo;
- Aos meus pais José Diniz de Araújo e Dalva Pereira de Araújo, pelo infinito amor, incentivo e compreensão em todos os momentos;
- Aos meus irmãos Décio, Fábio e Flávio e a minha irmã Stephanie, que sempre me incentivaram;
- Ao professor Angelo Perkusich, pela orientação, amizade, incentivo e ensinamentos;
- Aos amigos da graduação e do mestrado Danielly, Felipe, Júnior, Edmar, Alfranque, José Alves e Rex, pela amizade, companheirismo e momentos de descontração;
- Aos demais professores do DEE-UFCG, pelos ensinamentos e amizade;
- A todos os funcionários do DEE-UFCG, em especial a Adail, Ângela, Eleonora, Marcos Morais e Luis Carlos, pela amizade;
- Aos amigos e colegas de trabalho do Genius Instituto de Tecnologia de Manaus-AM, em especial a Silvio Mano Maeta;
- Ao CNPq, pelo importante apoio financeiro no primeiro ano do mestrado;
- Aos professores integrantes da banca de avaliação;
- Aos amigos Jimmy Cury, Luis Uebel, Álvaro Motta, Hallyson de Melo e Miguel Rodrigues.

Resumo

Neste trabalho apresenta-se uma solução distribuída para o gerenciamento de um portal de voz, baseado numa infra-estrutura distribuída denominada de GRP (Gerenciador de Recursos e Processos). No GRP é possível ajustar padrões de qualidade de serviço, possibilitar o trânsito de arquivos de configuração entre os componentes da infra-estrutura e a requisição de recursos por um gerenciador de aplicativos. O GRP foi projetado, implementado e testado no contexto de uma aplicação comercial real.

Abstract

In this work we present a solution to manage a voice portal based on an infrastructure named GRP (Resource and Process Manager). For it is possible to adjust quality of service requirements, allow the exchange of configuration files among the components of the infrastructure, and the request of resources by an application manager. The GRP has been designed, implemented in the context of a real commercial application

Lista de Símbolos e Abreviaturas

GRP - Gerenciador de Recursos e Processos
G – GRP - Gerenciador de Recursos e Processos Global
L – GRP - Gerenciador de Recursos e Processos Local
EDF - Earliest Deadline First
R – EDF - Reservation-based preemptive EDF
ASR - Automatic Speech Recognition
TTS - Text-To-Speech
CORBA - Common Object Request Broker Architecture
DSRT - Dynamic Soft Real Time Scheduler
QoS - Quality of Service
TS - Time Sharing
HTML - HyperText Markup Language
XML - Extensible Markup Language
VXML - Voice Extensible Markup Language
UCP - Unidade Central de Processamento
API - Application Program Interface
DTMF - Dual Tone Multi Frequency
TCP - Transmission Control Protocol
IP - Internet Protocol
GRM - Global Resource Manager
LRM - Local Resource Manager
URI - Uniform Resource Identifiers
SRT - Soft Real-Time
UML - Unified Modeling Language

Lista de Figuras

Lista de Tabelas

Conteúdo

Capítulo 1

Introdução

O desenvolvimento das redes de computadores, a melhoria do desempenho e o barateamento dos microprocessadores, são fatores que têm contribuído efetivamente para a disseminação do desenvolvimento de aplicações que utilizam processamento paralelo em sistemas distribuídos. Tal cenário promove o desenvolvimento de soluções com menor custo que aquelas baseadas em máquinas multiprocessadas corporativas. Com a utilização dos sistemas distribuídos pode-se também agregar diversos tipos de *hardware* e utilizar algoritmos específicos para gerenciamento e escalonamento dos recursos disponíveis numa rede de computadores, promovendo a escalabilidade e tolerância a faltas. Tornando as aplicações mais adaptáveis e também mais complexas [Cir02].

A aplicação de portal de voz concentra múltiplas atividades e atendem diversos clientes concorrentemente, necessitando alto poder de processamento de voz [Con01], concentrando aplicações de reconhecimento (*ASR, Automatic Speech Recognition*) e síntese (*TTS, Text-To-Speech*) de voz, navegadores *Voice-XML* [For00], além da infraestrutura de telefonia. Na Figura 1.1 apresentamos a estrutura de um portal de voz, com os servidores de síntese e reconhecimento, as instâncias do navegador por voz, interface para canais de voz e um servidor de gerenciamento.

A idéia é agregar as aplicações de um portal de voz como recursos disponíveis em uma rede de computadores, onde esses recursos estão conectados a uma plataforma de gerenciamento distribuída para disponibilizar um ambiente para execução de atividades distribuídas e paralelas, tornando-os mais interdependentes e colaborativos. Para isto, torna-se necessário definir uma política de gerenciamento para os recursos computacionais fisicamente dispersos em uma rede local, alocando de forma distribuída tarefas para diferentes recursos disponíveis em computadores que possam garantir as

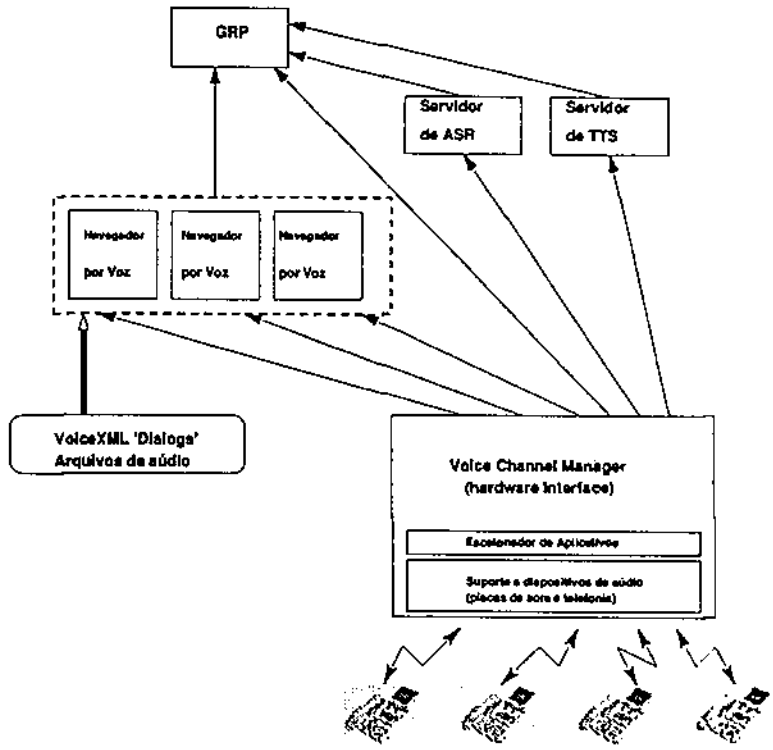


Figura 1.1: Estrutura de um portal de voz

condições e os requisitos para alocação destes recursos. Na Figura 1.2 apresentamos um exemplo de uma estrutura de portal de voz distribuída em três computadores, os elementos que compõem está figura serão discutidos nos próximos capítulos.

A política de gerenciamento baseia-se no registro dos recursos locais disponibilizados para um gerenciador local, com uma configuração que transita pela rede e é disponibilizada para todas as máquinas distribuídas no agrupamento. Cada agrupamento é composto por um gerenciador global e diversos gerenciadores locais. Qualquer agrupamento pode estar contido em um agrupamento maior, possuindo também um gerenciador global. Os gerenciadores locais executam os componentes do portal de voz.

1.1 Problema

No contexto desta dissertação o problema central abordado é estabelecer e implementar um mecanismo para garantir as requisições de serviços, e previsões de carga de processador para um portal de voz em uma arquitetura com diversos servidores em uma rede TCP/IP.

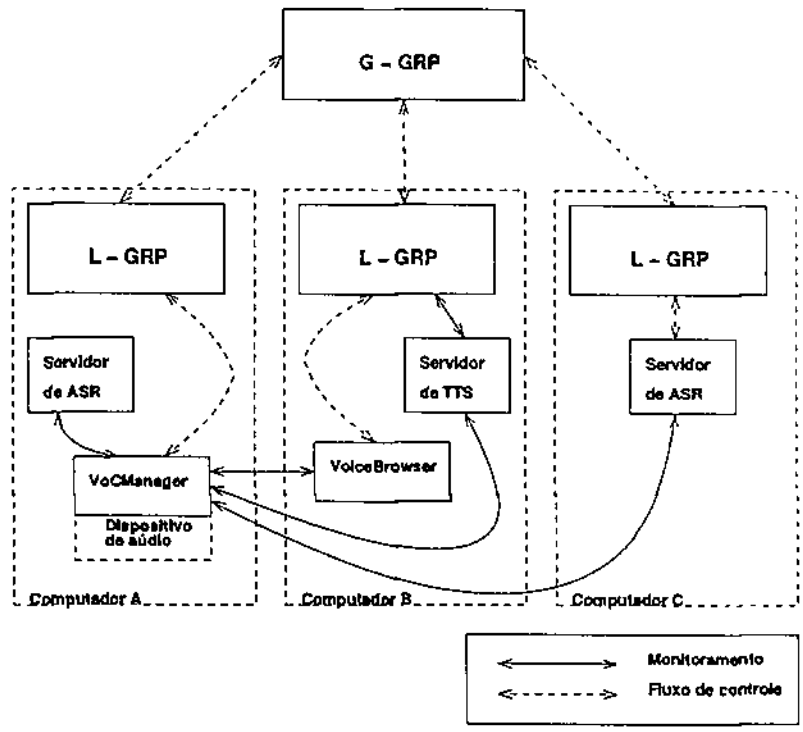


Figura 1.2: Estrutura distribuída de um portal de voz

1.2 Solução

A solução implementada neste trabalho é o desenvolvimento de um gerenciador de recursos e processos denominado de GRP (Gerenciador de Recursos e Processos) para atendimento das requisições de reconhecimento e síntese de voz.

O objetivo do GRP é implemementar os mecanismos necessários para o gerenciamento e escalonamento das requisições relacionadas com atividades de reconhecimento e síntese de voz. As informações necessárias para tanto são a carga de processamento em cada servidor e do número de instâncias já executadas daquele recurso.

O GRP é uma aplicação para gerenciamento dinâmico de recursos, e é instalado em todos os nós que compõem o sistema do portal de voz, como discutido no contexto do trabalho de Marques e Kon [MK02]. O GRP tem as informações de como disponibilizar o recurso alocado na máquina ou grupo de máquinas que ele está gerenciando, e também como deve proceder no atendimento de uma requisição de um cliente para um recurso. Sua política de escalonamento é prover o melhor uso de todos os recursos disponíveis, garantir tolerância à falhas e qualidade de serviço para os recursos registrados.

No Capítulo 2 apresentamos uma discussão sobre outros gerenciadores diponíveis e

estudados. Eles destinam-se a rede de grande porte e gerenciam e escalonam recursos computacionais heterogêneos. O escalonamento é baseado no fornecimento do recurso que melhor se aproxime das especificações requeridas.

O *Qualman* [NhCN97] é o que mais se aproxima da nossa infra-estrutura, incorporando a admissão e reserva de recursos. Permitindo reserva de recursos de processamento e memória.

O gerenciador de recursos proposto por Marques e Kon [MK02] é um arcabouço com estrutura hierárquica baseada em um padrão da indústria denominado de CORBA (*Common Object Request Broker Architecture*) [Käh00] para o gerenciamento dinâmico de recursos em sistemas distribuídos. Admitindo a reserva de recursos de processamento de acordo com especificações dos clientes. Este serviço é fornecido por uma biblioteca denominada de *DSRT (Dynamic Soft Real Time Scheduler)*, citado em [NhCN97] [Gup99].

O gerenciador de recursos do projeto *Globus* [Sie99] também baseia-se em uma hierarquia de gerenciadores. A grande desvantagem é que ele é baseado em um sistema de comunicação proprietário, e não possui suporte a reserva de recursos. O *Legion* [CKKG99] e o *Prospero* [NR94] também possuem uma estrutura semelhante, entretanto também não oferecem suporte a reserva de recursos.

1.3 Organização do trabalho

Esta dissertação está organizada como segue:

No Capítulo 2 apresenta-se trabalhos relacionados com o tema dessa dissertação, como o *Prospero Resource Manager* [NR94], o *Legion* [CKKG99], o gerenciador de recursos do projeto *Globus* [Sie99], o *QualMan* da Universidade de Illinois em Urbana-Champaign [NhCN97] e o *Middleware* para gerenciamento de recursos desenvolvido na Universidade de São Paulo - USP [MK02].

No Capítulo 3 descreve-se o gerenciador de recursos, detalhando os estudos envolvidos e soluções para os problemas apresentados.

No Capítulo 4 apresenta-se a implementação do gerenciador de recursos como definido no Capítulo 3. A linguagem utilizada na implementação foi C++.

No Capítulo 5 apresenta-se os resultados obtidos com a aplicação do GRP ao portal de voz implementado no Genius Instituto de Tecnologia em Manaus-AM.

No Capítulo 6 apresenta-se as conclusões e sugestões para trabalhos futuros. E por fim a bibliografia utilizada para este trabalho.

Capítulo 2

Trabalhos Relacionados

O objetivo deste capítulo é apresentar uma visão de outras soluções disponíveis na literatura no contexto de gerenciamento de recursos e processos em ambientes distribuídos.

2.1 Gerenciador de Recursos Prospero

O PRM (*Prospero Resource Manager*) [NR94] é um sistema de alocação de recursos escalável para alocação de recursos em grandes redes e em sistemas multiprocessados. O PRM é parte do projeto SCOPE (*Scalable Computing Infrastructure*) [NR93] do *Information Sciences Institute* da *University of Southern California.* O *SCOPE* foi desenvolvido e projetado para alocação de serviços distribuídos para um grande número de recursos computacionais heterogêneos. Esses serviços estão localizados em diversas instituições e organizações que cooperam entre si para suprir capacidades computacionais, e também podem ser usados por usuários mediante o pagamento por processamento aos provedores dos serviços, para terem acesso a serviços sobre demanda de capacidade. O *SCOPE* integra elementos de alocação de recursos, autenticação e pagamento *on-line.*

O PRM implementa mecanismos para administrar os recursos em dois níveis:

1. alocando recursos de sistemas por demanda de capacidade computacional;

2. adminstração em separado dos recursos assinalados por cada trabalho.

O PRM é utilizado em redes de grande escala. Sendo a sua hierarquia composta por múltiplos gerenciadores, cada um controlando uma pequena porção da rede, podendo ser uma pequena rede local, ou uma rede de uma empresa ou até mesmo uma rede composta por várias instituições separadas geograficamente.

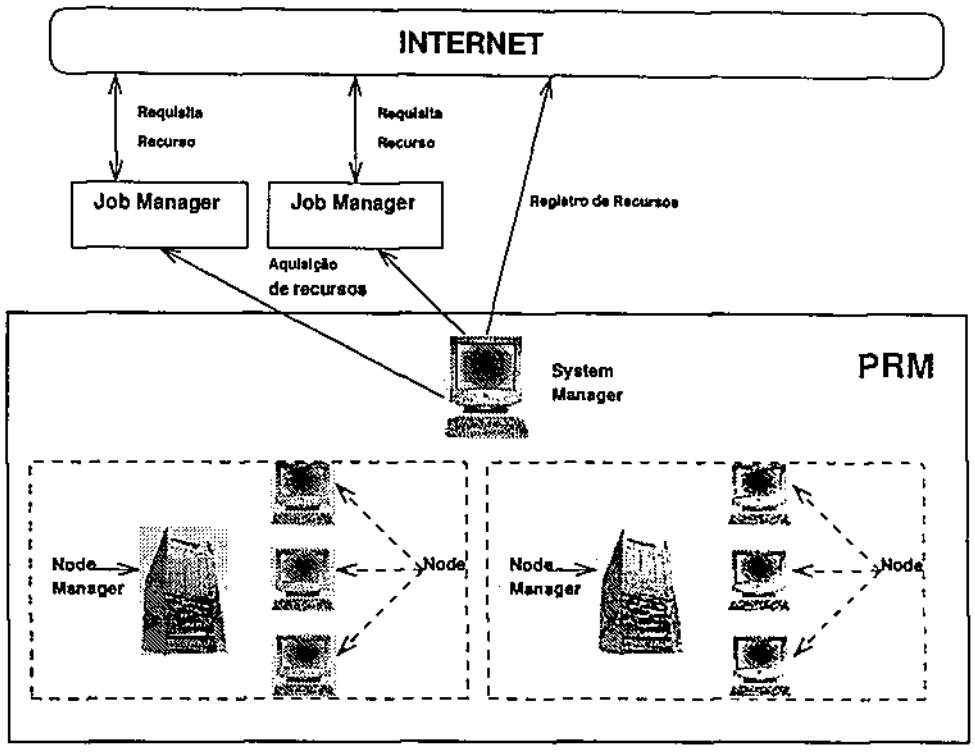


Figura 2.1: Diagrama do *Prospero Resource Manager*

As funcionalidades do PRM são distribuídas entre três tipos de gerenciadores, veja Figura 2.1: *System Manager, Job Managers e Node Managers.*

O *System Manager* é o administrador geral dos recursos. Onde diversos *System Managers* podem administrar recursos diferentes em um agrupamento hierárquico. Quanto mais alto o nível na hierarquia, maior é a responsabilidade pelo gerenciamento de um número maior de recursos. O controle dos recursos pode ser transferidos de um *sytem manager* para outro de um nível maior na hierarquia. O *System Manager* mantém informações sobre as características e a disponibilidade de cada recurso, e a que trabalho cada recurso está associado.

O *Job Manager* atua como uma abstração de um sistema virtual, ou seja, como uma interface em que as tarefas podem requisitar os recursos. O *Job Manager* administra os recursos que foram alocados para o trabalho pelos *system managers* responsáveis por cada recurso.

O *Node Manager* é executado localmente em cada nó da rede. Ele aceita mensagens do *System Manager*, identificando o *Job Manager* que irá ser carregado e executa componentes de serviços (programas). Quando autorizado pelo *Job Manager*, ele carrega e

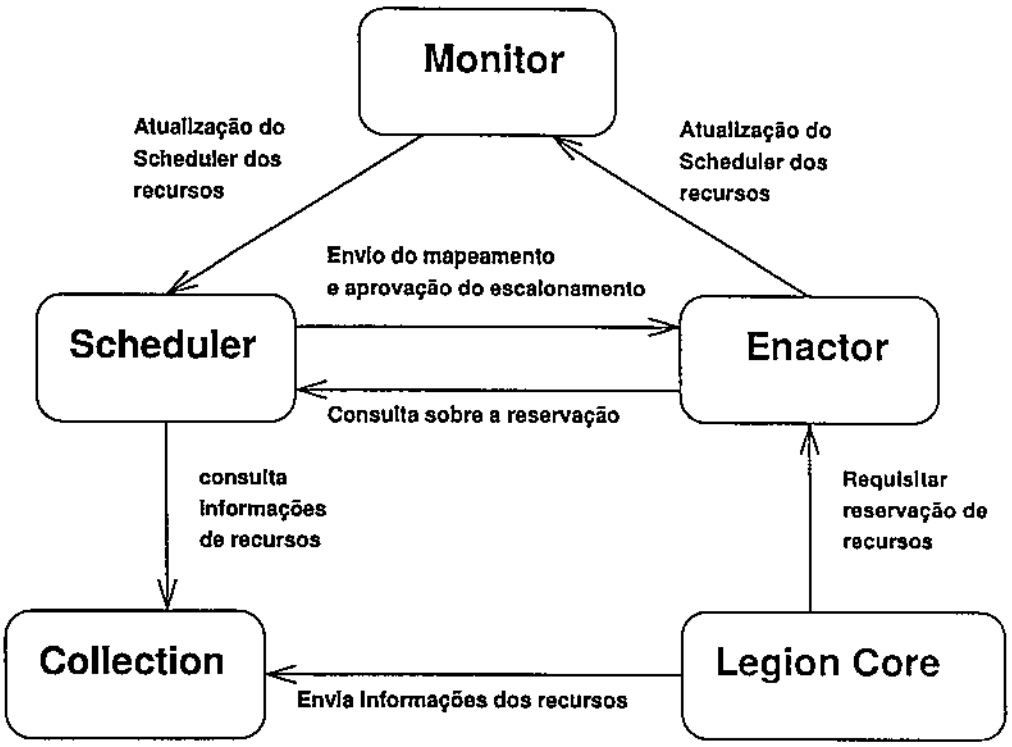


Figura 2.2: Componentes lógicos do *Legion*

executa componentes de serviços, e notifica o *Job Manager* sobre eventuais falhas nas tarefas.

2.2 Gerenciamento de Recursos no *Legion*

O *Legion* [CKKG99] pode conectar centenas e ou até milhões de computadores geograficamente separados para utilização massiva de supercomputação paralela através de *links* de alta velocidade. Estes recursos computacionais podem ser administrados por diversas organizações.

A reserva de recursos é feita através da negociação entre os provedores dos recursos e os consumidores. O *Legion* consiste de quatro componentes: *Collection, Scheduler, Enactor e Monitor.* Na Figura 2.2 tem-se os componentes lógicos do *Legion* e como estes são relacionados para o escalonamento e reserva de processos.

Collection é um repositório de informações descrevendo o estado dos recursos contidos no sistema. O *Scheduler* interage com o *collection* requisitando informações das instâncias dos recursos, e então as mapeias de instâncias de objetos para recursos. Este

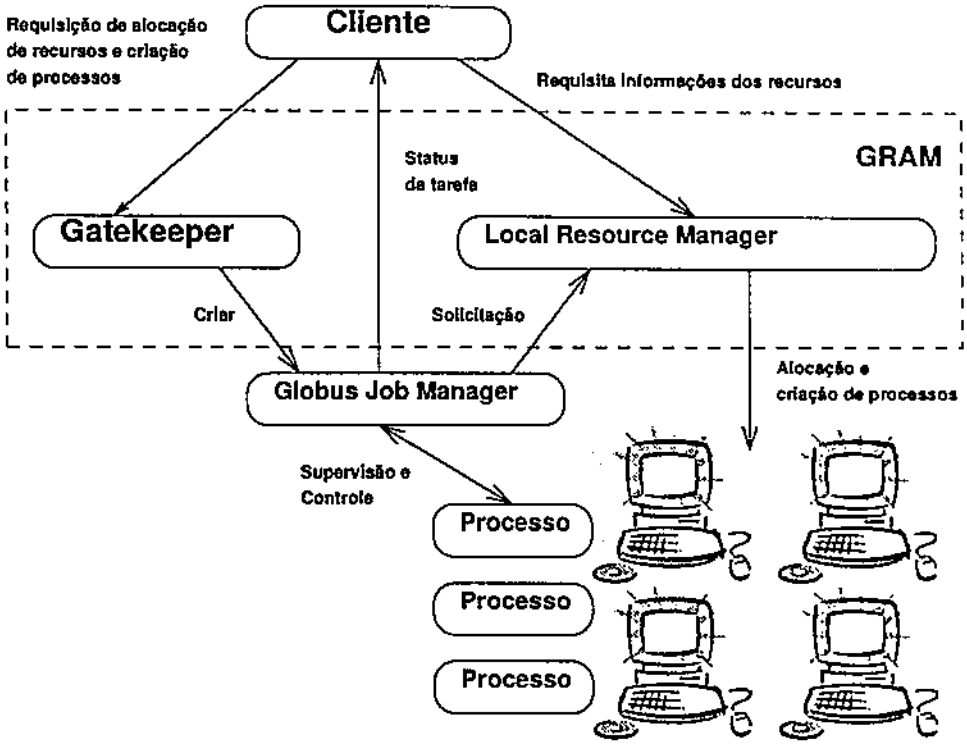


Figura 2.3: *Globus Resource Allocation Manager*

mapeamento é passado para o *Enactor* para implementação, e o resultado é reportado para o *Scheduler*. O *Monitor* monitora os estados dos recursos através de sistemas dirigidos a eventos.

2.3 Gerenciador de Recursos Globus

O projeto *Globus* [Sie99] é um esforço desenvolvido no *Argonne National Laboratory* e da *University of Southern California Information Sciences Institute*. O *Globus* é um conjunto de ferramentas para prover serviços básicos de um *Grid Computacional* [1] [Cir02], incluindo gerenciamento de recursos.

O *Globus Resource Allocation Manager*-GRAM provê uma ferramenta para administrar recursos. GRAM implementa mecanismos para traduzir requisição de recursos em comandos que são especificados para sistemas locais particulares. A interface com

[1] Um *Grid Computacional* é uma rede de computadores que agrega recursos computacionais que podem ser usados de forma transparente pelos usuários, escondendo do usuário a complexidade que o sistema envolve para o fornecimento do serviço computacional.

o usuário do GRAM é chamada de *gatekeeper*. Na Figura 2.3 tem-se todos os componentes do *GRAM* e o *Job Manager*, e como eles se relacionam. O *gatekeeper* recebe as requisições na máquina em que ele está instalado, autentica o usuário, e executa um processo de *job manager* com a autenticação do usuário. O *job manager* implementa a comunicação com o usuário local, alocação de recursos, execução de trabalhos e a liberação dos recursos após a finalização do trabalho executado. Para coordenar a requisição de recursos para múltiplas máquinas, é utilizado o *co-allocator*, que sincroniza o processo de alocação.

2.4 *QualMan* com reserva de tempo de processamento

A arquitetura do *QualMan* atende as garantias de QoS (*Quality of Service*) [NS95] para aplicações de multimídia, veja artigo do grupo da Universidade de Illinois em Urbana-Champaign [NhCN97]. Ele consiste de um conjunto de servidores de recursos de processador, memória e comunicação. Incluindo um mecanismo de administração de variação de QoS. A função deste mecanismo de QoS é traduzir as condições desejáveis de qualidade de serviço para parâmetros escalonáveis do sistema dos servidores de recursos de processador, memória e comunicação.

O *DSRT* (*Dynamic Soft Real Time Scheduler*), veja seção 3.3.2, é o servidor de recurso de processador na arquitetura do *QualMan*, ele utiliza algoritmos de escalonamento baseados em EDF (*Earliest Deadline First*) para escalonar diversos processos para tempo real não-críticos e melhor esforço [FFO00].

2.5 Gerenciador de Recursos *RM*

O RM (*Resource Manager*) é um serviço de *middleware* baseado em objetos distribuídos que dá suporte ao gerenciamento dinâmico de recursos em sistemas distribuídos [MK02]. Sendo esses serviços baseados em CORBA (*Common Object Request Broker Architecture*) [Käh00], podendo desta forma interagir com outros serviços e aplicações CORBA em um ambiente heterogêneo.

O Gerenciador de Recursos permite controlar recursos presentes em uma hierarquia de agrupamentos conectados a Internet. Através do RM pode-se também especificar a

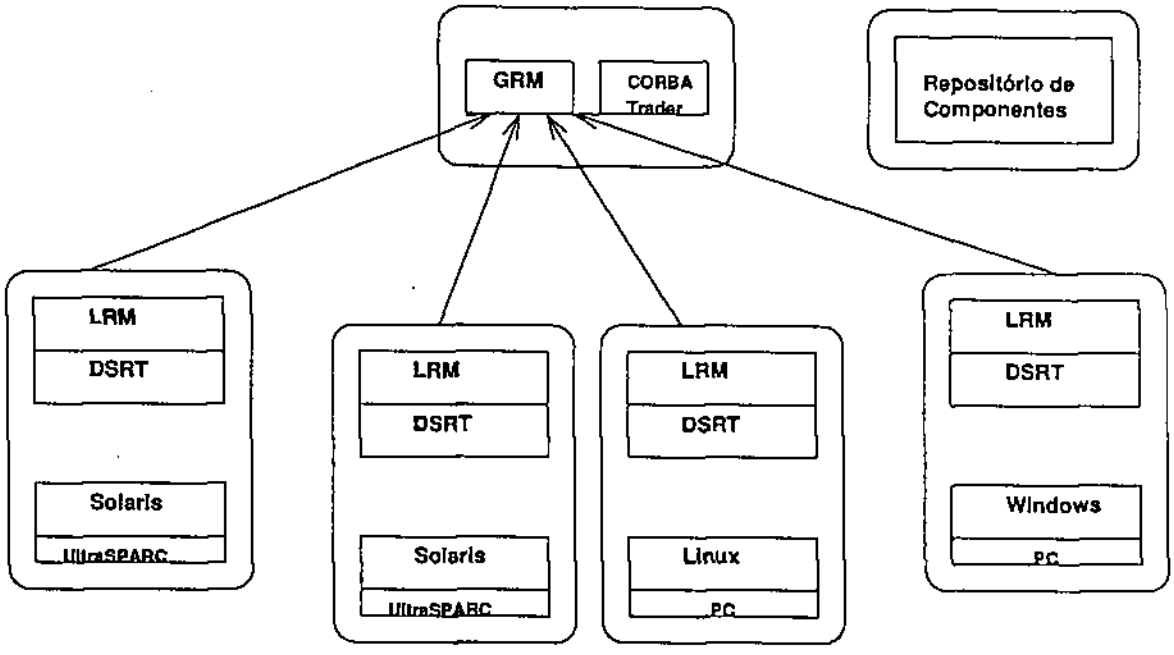


Figura 2.4: Organização de um agrupamento com GRM e LRM

QoS através de reserva de recursos, utilizando a biblioteca do DSRT, veja Seção 3.3.2.

O serviço é formado pelo LRM (*Local Resource Manager*), GRM (*Global Resource Manager*), e um repositório de componentes, veja Figura 2.4. O primeiro gerencia os recursos de uma única máquina, o segundo possui uma visão global de todo um agrupamento e gerencia os recursos do agrupamento como um todo, e o último armazena o código executável dos componentes de software juntamente com a informação sobre os requisitos para a execução daquele componente.

2.6 Conclusão

Neste capítulo podemos observar duas linhas de gerenciadores: os gerenciadores implementados para escalonar recursos computacionais de diferentes instituições conectadas a uma infra-estrutura de computação escalável, e os gerenciadores que escalonam serviços de multimídia incorporando a admissão e reserva de recursos.

Através do estudo desses gerenciadores, concluiu-se que uma solução distribuída

Gerenciadores de Recursos	Estrutura Hierárquica	Reserva de Recursos	Sistema Prioritário	Sistema Operacional
GRP	Sim	Sim	Não	Windows
Qualman	Não	Sim	Não	Windows, UNIX
RM	Sim	Sim	Não	Windows, UNIX e Solaris
Globus	Sim	Não	Sim	UNIX
Legion	Sim	Sim	Sim	Windows, UNIX
Prospero	Sim	Não	Sim	SunOS, HP-UX

Tabela 2.1: Comparação entre os Gerenciadores de Recursos

para o portal de voz seria em disponibilizá-lo em uma infra-estrutura baseada em agrupamentos hierárquicos, e realizar o gerenciamento e escalonamento dos recursos com admissão e reserva de recursos. Na Tabela 2.1 apresenta-se uma comparação entre o GRP e os outros gerenciadores estudados.

Capítulo 3

Gerenciador de Recursos e Processos

O GRP (Gerenciador de Recursos e Processos) é uma infra-estrutura de software para administrar os recursos disponíveis em uma rede distribuída. Ele recebe solicitações e registra recursos, verificando o seu comportamento ao longo da vida útil de cada recurso. Interagindo também com o sistema operacional [SGG02] [Tan92] para verificar parâmetros locais da máquina.

O papel principal do GRP é promover o melhor uso dos recursos disponíveis, garantir tolerância à falhas e qualidade de serviço para os recursos registrados. Priorizando a execução de aplicações paralelas em uma plataforma adequada às características que a aplicação exige, como quantidade adequada de processamento livre e condições de tempo-real exigidas para a execução de tarefas processadas na aplicação. O GRP gerencia recursos que podem estar fisicamente separados. Entende-se por recursos: máquinas, memória e aplicativos.

O GRP administra recursos em dois níveis: alocar recursos para um determinado trabalho[1], e administrar separadamente cada recurso destinado a uma determinada tarefa.

[1]Entende-se por trabalho como uma coleção de tarefas sendo executadas para a realização de um serviço.

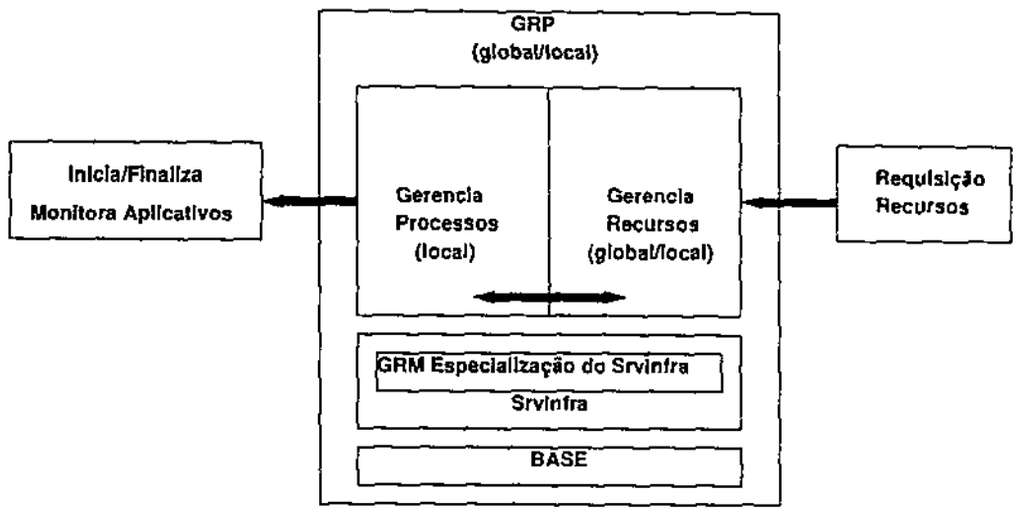


Figura 3.1: Gerenciador de recursos global/local

3.1 Arquitetura

O Gerenciador de Recursos e Processos (GRP) pode ser denominado de G-GRP (Global GRP) ou L-GRP (Local GRP) [KYH+01] [MK02], veja Figura 3.1. O primeiro, é um gerenciador de recursos para um agrupamento de gerenciadores locais. O segundo, é um escalonador de recursos que monitora os recursos locais a ele registrados, no caso de um portal de voz, os recursos são os reconhecedores e sintetizadores de voz, o navegador VXML e a interface de telefonia. Na Figura 3.2 tem-se os blocos funcionais de um portal de voz, o GRP implementa a coordenação e configuração, o *ASRServer* e o *TTSServer* implementa o processamento do áudio, o *VoiceBrowser* implementa o controle da execução da infra-estrutura e o *VoCManager* pela interface com os dispositivos de áudio (placa de som ou placa de telefonia). Esses recursos são aplicações que são continuamente requisitadas para a execução de trabalhos não críticos.

Em único nó da rede pode-se ter o mesmo GRP nos dois modos de operação: G-GRP e L-GRP. Entende-se por nó um elemento físico de uma rede, com memória e processamento local.

O GRP implementa as seguintes funcionalidades:

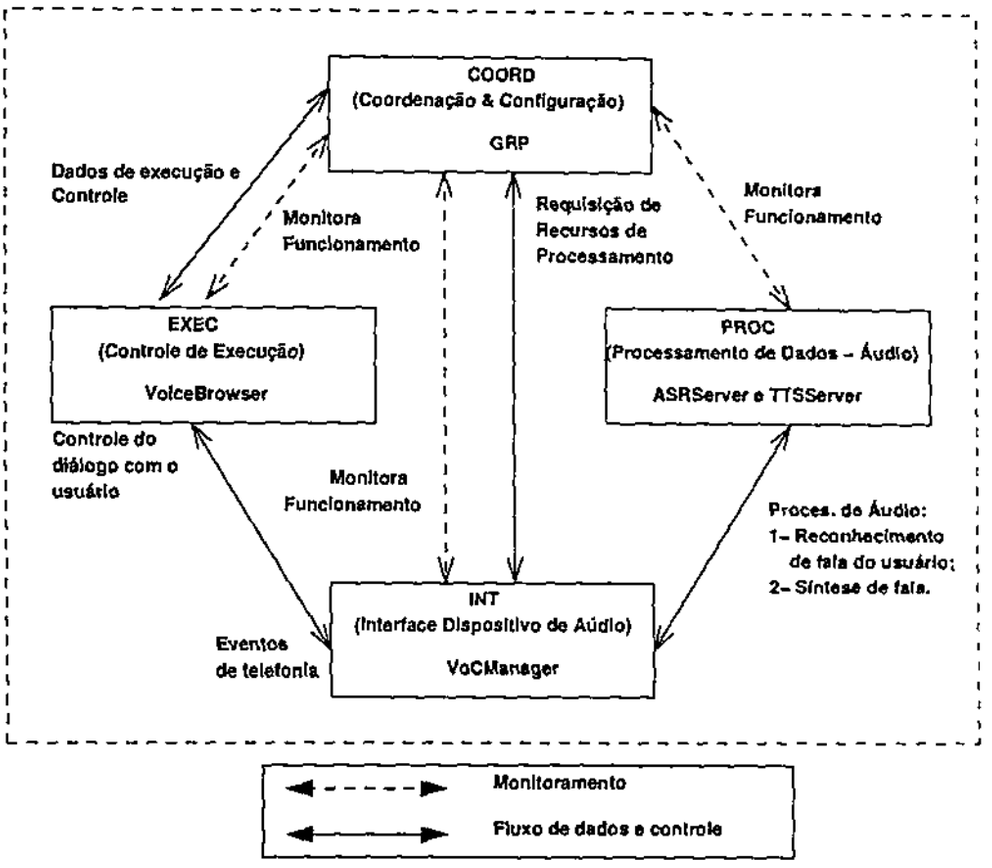


Figura 3.2: Blocos Funcionais de um Portal de Voz

- Administração de recursos;
- Monitoração de processos associados aos recursos;
- Executação dos aplicativos que disponibilizam um determinado recurso.

Além de administrar recursos o GRP pode também monitorar os processos associados aos recursos, com a execução dos aplicativos que provêem o recurso e a monitoração dos seus estados.

Uma característica importante desses escalonadores é que eles recebem continuamente solicitações de vários escalonadores de aplicativos [Gup99] e, portanto, têm que arbitrar entre esses vários escalonadores de aplicativos, o uso dos recursos que controlam.

3.1.1 Escalonadores de Aplicativos

Os escalonadores de aplicativos utilizam os recursos controlados por vários escalonadores de recursos distintos, conhecendo detalhes da aplicação que escalonam. Escalonadores de aplicativos são construídos com uma classe de aplicações necessárias predefinidas. Eles são também recursos que estão disponíveis em determinados nós da rede, onde esses recursos são iniciados e gerenciados pelo L-GRP.

No nosso caso, o gerenciador de aplicativos está acoplado à interface de telefonia, pois esta estrutura prove mecanismos para possibilitar que o usuário do sistema tenha acesso aos diversos serviços do Portal de Voz, tais como: reconhecimento e síntese de voz, e as páginas VoiceXML interpretados pelo sintetizador de voz.

Os escalonadores de aplicativos:

1. escolhem quais recursos serão utilizados na execução de uma determinada atividade;

2. estabelecem quais tarefas cada um destes recursos realizará;

3. submetem solicitações aos escalonadores de recursos locais (L-GRP) para que as tarefas sejam executadas.

Os escalonadores de aplicativos não controlam os recursos que usam. Eles obtém acesso a tais recursos submetendo solicitações para o seu L-GRP. Caso o L-GRP não consiga atender a solicitação de recurso do escalonador de aplicativos, então o L-GRP envia a requisição para o seu G-GRP, o qual decide se na rede existe algum L-GRP que possa disponibilizar o recurso. A disponibilidade desse recurso depende das características do nó, e se o mesmo tem condições de atender o QoS requisitado para o determinado recurso.

3.1.2 Quadro Negro

Num estilo de repositório existem dois tipos distintos de componentes: uma estrutura de dados centrais, que representa o estado atual; e uma coleção de componentes independentes que opera sobre o armazenamento central de dados. Quando o estado atual da estrutura de dados central é a principal causa da seleção de processo a ser executado, o repositório pode ser um quadro negro (*blackboard*) [SG96] [PdAdA03]. Na Figura 3.3 é apresentado o modelo do quadro negro, onde *ksi* são as estruturas computacionais

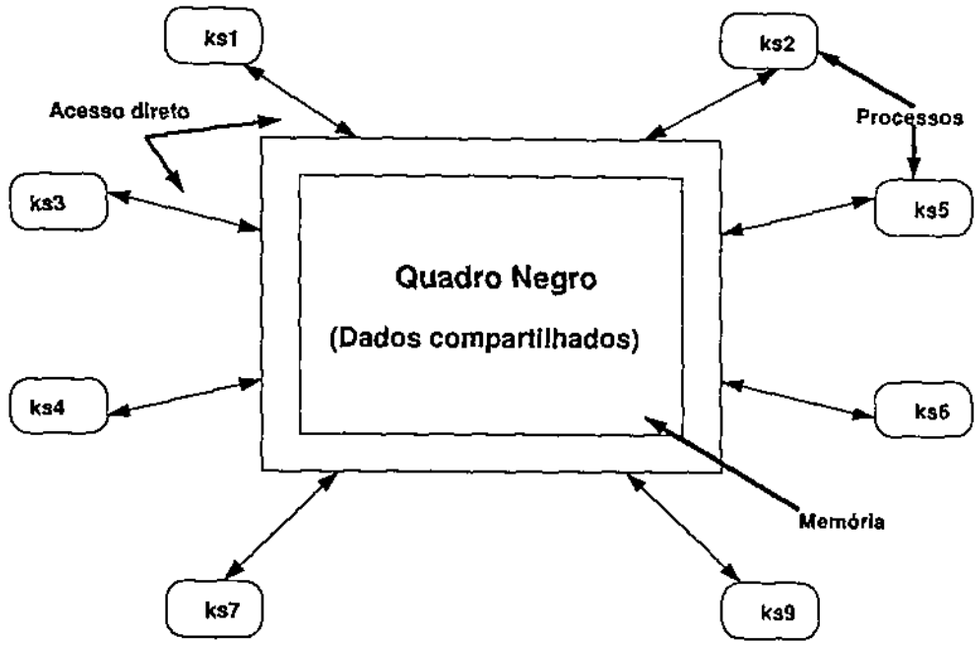


Figura 3.3: *Blackboard*

que fazem acesso direto à estrutura de dados ou memória compartilhada representada pelo quadro negro.

Através da utilização de uma estrutura compartilhada, uma forma de comunicação indireta pode ser estabelecida entre os componentes que armazenam e recuperam dados da estrutura.

No GRP o quadro negro é uma estrutura de dados representando os recursos gerenciados. Nesta estrutura estão presentes os recursos gerenciados com os seus respectivos parâmetros de configuração, e variáveis de controle.

3.1.3 Agrupamento

Na Figura 3.4 temos a nossa infra-estrutura de gerenciadores organizada em agrupamentos, onde cada G-GRP tem acesso a todas as informações necessárias para o gerenciamento dos L-GRP conectados ao seu agrupamento, mantendo uma visão global da árvore ou sub-árvore do agrupamento. Sendo os GRPs organizados de forma hierárquica pelos administradores do sistema e o algoritmo de atualização é estendido de forma que os GRPs em um nível hierárquico enviem de tempos em tempos uma

atualização do estado do seu agrupamento para o GRP do nível superior.

Na estrutura de árvore o L-GRP é responsável por gerenciar os recursos em uma única máquina. O GRP recebe solicitações por recursos de gerenciadores de aplicativos locais que interagem diretamente com o L-GRP, ou por serviços remotos cujas solicitações vem através do G-GRP. Quando requisitado por um determinado L-GRP ou outro G-GRP, o G-GRP envia através da rede todos as informações necessárias para o gerenciamento de um agrupamento ou de um nó.

Algumas regras devem ser observadas na construção dos agrupamentos:

- cada agrupamento é composto por:
  - um gerenciador global;
  - gerenciadores locais;
  - recursos gerenciados localmente em cada máquina do agrupamento;
  - agrupamentos podem originar outras sub-árvores com G-GRP e L-GRP associados
- qualquer agrupamento pode pertencer a um agrupamento maior, sendo sub-árvore deste agrupamento;
- os componentes do portal de voz são executados pelos gerenciadores locais;
- cada L-GRP é associado a uma única máquina;
- o fluxo de informação de monitoramento, dados e controle, obedecem a uma hierárquia na sub-árvore e árvore do agrupamento;
- a sub-divisão de um agrupamento em sub-árvores, depende:
  - da posição geográfica;
  - do número de GRP conectados;
  - das regras de gerenciamento estabelecidas pelos administradores dos sistemas.

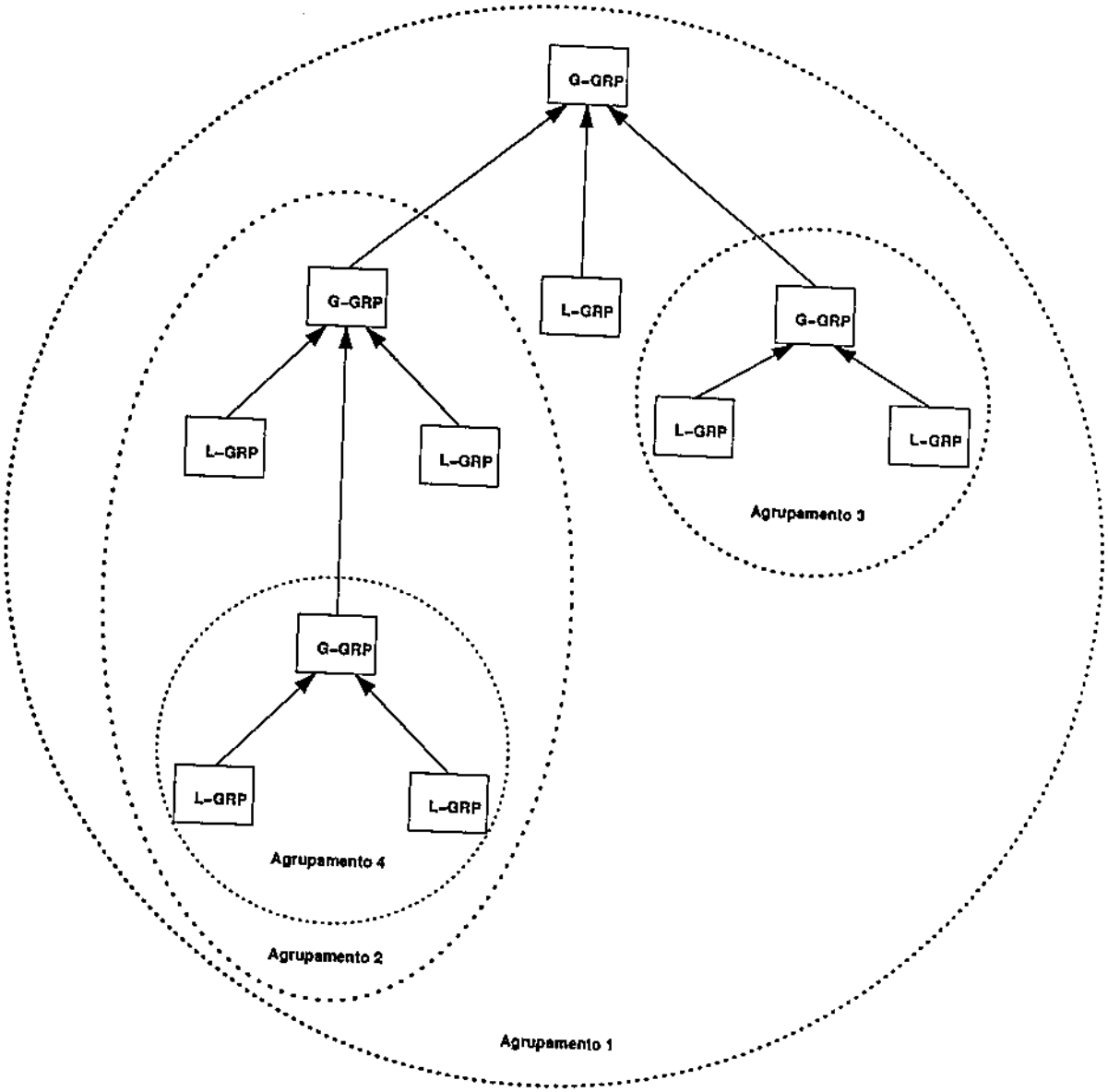


Figura 3.4: Agrupamento de uma rede com GRPs

3.1.4 Módulos do GRP

O GRP é composto por seis módulos, enumerados a seguir:

1. **Módulo de Suporte de Aplicação** - Este módulo é o núcleo de funcionamento central do GRP, implementa os mecanismos para tratar todas as requisições provenientes dos recursos ou escalonadores de aplicativos. Além disso, efetua a passagem de parâmetros para outros GRPs ou recursos a ele registrados;

2. **Monitor Local** - Este adquire as informações sobre a máquina local, como a carga de processamento da UCP ou memória total utilizada por cada recurso registrado;

3. **Módulo QoS** - Este módulo implementa a administração da QoS na máquina local, manipulação dinâmica do nível de satisfação da QoS e o escalonamento em tempo-real das tarefas. Este módulo utiliza-se da biblioteca DSRT/NT (*Dynamic Soft Real-Time*). O DSRT é um escalonador dinâmico a nível do usuário para escalonar tarefas periódicas e aperiódicas de tempo real suave [FFO00] [KWP01], desenvolvido pelo departamento de Ciência da Computação da Universidade de Illinois em Urbana-Champaign, veja Seção 3.3.2.

4. **Módulo TCP/IP** - Administra todas as conexões TCP/IP;

5. **Módulo de Administração de Processos** - Responsável pela inicialização dos recursos em uma máquina local e o monitoramento desses recursos durante toda a sua vida útil;

6. **Gerenciador de Licenças** - O gerenciador de licenças é uma biblioteca desenvolvida pela equipe do Genius para controlar os *software* enviados para os clientes externos e garantir a sua utilização legal. É baseado em uma entrada no registro do *Microsoft Windows*, com informações coletadas do aplicativo sob controle e da máquina hospedeira.

   O objetivo é criar uma combinação de dados que possa dificultar a quebra da segurança, definida por alguma política do Genius.

   A sequência de operações definidas neste modelo é a seguinte:

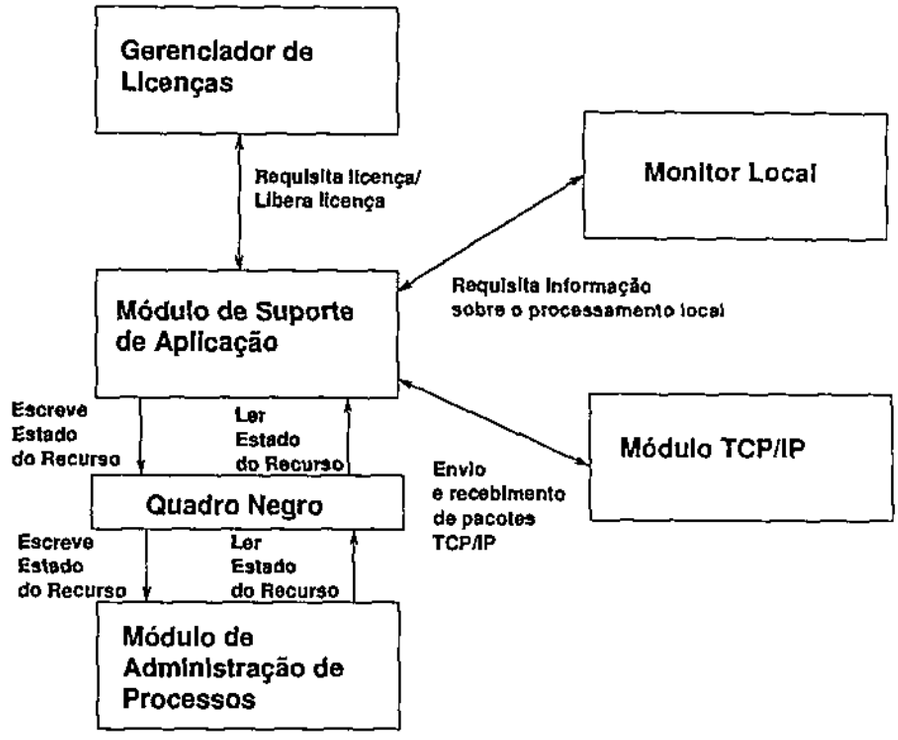


Figura 3.5: Módulos do Gerenciador de Recursos e Processos

(a) definir um conjunto de informações necessárias para identificar o aplicativo;

(b) coletar estas informações da máquina em que o *software* será instalado;

(c) criptografar estes dados;

(d) armazenar estas informações na máquina, basicamente como uma entrada no registro do *Microsoft Windows*;

(e) ler posteriormente esta entrada;

(f) descriptografar;

(g) validar a licença do aplicativo para cada vez que ele for executado, ou quando for necessário;

(h) informar quando não for possível ter acesso a uma licença válida.

Estes módulos compõem a estrutura do Gerenciador de Recursos e Processos (Global e Local). Na Figura 3.5 temos um diagrama de blocos dos módulos e que tipos de informações são trocados entre eles.

3.2 Portal de Voz

Um portal de voz é uma forma de acessar informações da *Web* por usuários de aparelhos telefônicos, através da linguagem natural como interface para acessar conteúdo [Con01].

Um serviço de portal de voz provê acesso através do telefone para informações como notícias, meteorologia, rotas urbanas, cinema, restaurante e etc.

Em uma aplicação de portal de voz, deve-se garantir que:

- a confiabilidade no reconhecimento de voz deve ser alta, evitando o reenvio de amostras para processamento e reconhecimento, bem como a consistência dos dados, garantindo a causalidade das amostras;
- a tolerância a faltas e suporte à comunicação de grupo devem ser atendidas;
- as requisições aos gerenciadores do sistema distribuído devem ser rapidamente atendidas, ou enviadas para outros gerenciadores que possam atender a tal solicitação;
- deve-se garantir a distribuição das licenças para o reconhecedor de voz de forma segura.

3.2.1 *VoiceXML*

De forma análoga ao padrão *HTML (HyperText Markup Language)*, há um padrão para escrever o documento que será carregado e interpretado pelo navegador do portal de voz. Um dos padrões mais conhecidos para escrever este tipo de documento é o *VoiceXML* [For00], sendo este uma extensão do *XML*[2].

Os documentos *VoiceXML* descrevem:

- Textos falados (fala sintetizada artificialmente);
- Reprodução de gravações contidas em arquivos de áudio;
- Reconhecimento de palavras e sentenças. Esse reconhecimento é baseado em gramáticas, que delimitam quais sentenças ou palavras são reconhecidas;
- Reconhecimento de tons *DTMF (Dual Tone Multi Frequency)*;

[2] O *XML* é um padrão desenvolvido especialmente para controle de fluxo de "diálogos".

- Gravação de fala do usuário;
- Controle do fluxo de um diálogo;
- Controle da telefonia (finalização de uma chamada, transferência de chamadas).

O principal objetivo do uso do *VoiceXML* é permitir que os desenvolvedores possam abstrair detalhes específicos de implementação dos sistemas, e possam se concentrar no desenvolvimento de aplicações. Aliando as vantagens do serviço *web* às aplicações de voz.

Exemplo

O exemplo a seguir solicita que o usuário escolha umas das opções: notícias, tempo ou esportes. A resposta do usuário é aceita, e então passada para um outro documento (um *script* chamado *select.jsp*) que irá fornecer o serviço que o usuário selecionou.

```
<?xml version="1.0" ?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml" xml:lang="pt-BR">

    <form>
        <field name="selection" >
            <grammar type="application/srgs">
                main = Not&amp;iacute;cias | Tempo | Esportes;
            </grammar>
            <prompt>
Escolha uma das seguintes op&amp;ccedil;&amp;otilde;es:
     Not&amp;iacute;cias, Tempo ou Esportes.
            </prompt>
        </field>
        <block>
            <submit next="select.jsp" />
        </block>
    </form>
</vxml>
```

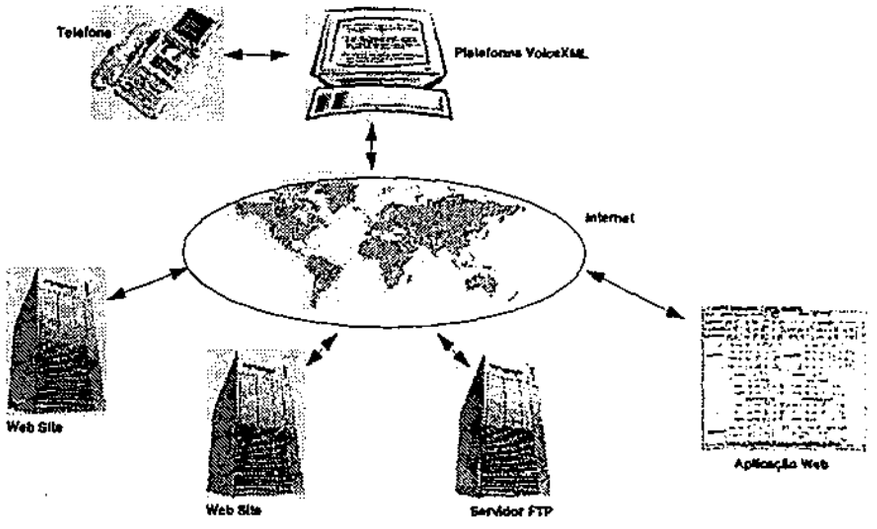


Figura 3.6: Execução de uma aplicação VoiceXML

Execução de uma aplicação VoiceXML

Um usuário conecta-se à aplicação discando um número de telefone apropriado. O interpretador *VoiceXML* responde a chamada e inicia a execução de um documento *VoiceXML*. Na Figura 3.6 apresenta-se como é estruturado o ambiente de execução de uma aplicação *VoiceXML*, diferenciando das aplicações *web* tradicionais pela inserção do telefone. Sob o controle do documento, o interpretador deve executar as seguintes ações:

1. enviar áudio de voz sintetizada, mensagens, ou outros conteúdos de áudio (tais como música ou outros efeitos sonoros) para o usuário;
2. aceitar entradas numéricas que o usuário envia por meio de sinais *DTMF*;
3. aceitar entradas de voz e reconhecer as palavras;
4. aceitar entradas de voz e simplesmente gravá-las, sem tentar reconhecer qualquer palavra;
5. enviar informações do usuário para o servidor *web*;
6. receber informações da Internet e repassá-las ao usuário.

Documentos *VoiceXML* podem executar funções de programação, como processamento aritmético e de texto. Isto permite a validação das entradas enviadas pelo

usuário. Além disso, a sessão do usuário não precisa ser uma seqüência simples de operação, que é executada do mesmo modo sempre, para isso o documento deve incluir lógica de decisão e outras estruturas mais complexas.

3.2.2 Gramáticas de Voz

A utilização de gramáticas é uma poderosa característica da linguagem *VoiceXML* [Ani01b], pois permite a criação de diálogos flexíveis que podem responder às entradas dos usuários na forma de linguagem natural [Ani01a].

Os sistemas de reconhecimento de fala fazem uso de gramáticas para a realização do reconhecimento. O objetivo principal de uma gramática é definir e limitar o espaço de busca para o sistema de reconhecimento. Uma gramática define quais são as palavras e quais são as seqüências que podem ser reconhecidas. Qualquer palavra ou seqüência de palavras que não tenha sido contemplada pela gramática não será reconhecida pelo sistema.

Por exemplo, se uma gramática define três palavras, "quadrado", "triângulo"e "círculo", o reconhecedor só será capaz de interpretar corretamente essas palavras quando faladas isoladamente. Se o usuário falar a palavra "pentágono", esta não será reconhecida, pois não faz parte da gramática. Se o usuário falar "quero quadrado", isto também não será reconhecido, pois a palavra "quero"não foi definida, nem a seqüência de palavras "quero quadrado"foi definida como sendo uma seqüência válida.

A definição correta de uma gramática é muito importante para garantir que o sistema de reconhecimento seja capaz de compreender os comandos do usuário.

O portal de voz suporta o *Augmented BNF Form do W3C Speech Recognition Grammar Format.*

As gramáticas podem ser definidas de duas formas diferentes:

1. no corpo de um documento *VoiceXML*, ou seja, uma gramática *inline*;
2. em um documento externo que é referenciado através de uma *URI (Uniform Resource Identifiers)* - uma gramática externa.

Uma gramática *inline* é definida da seguinte forma:

```
<grammar type="application/srgs">
    main = vermelho | azul | laranja;
</grammar>
```

Uma gramática externa é referenciada da seguinte forma:

```
<grammar type="application/srgs" src="colors.gram"/>
```

O elemento *<grammar>* é usado para fornecer uma gramática de voz que:

- especifica uma série de sentenças que o usuário deve falar para executar uma ação ou prover uma informação;
- fornece uma *string* correspondente ou uma série de valores para descrever a informação ou ação.

O campo *src* define a *URI* do arquivo da gramática. No exemplo apresentado acima, a gramática está definida no arquivo "colors.gram".

3.2.3 Definição de Regras para uma Gramática

Uma regra é composta basicamente pela definição do "nome da regra"e a "expansão da regra". Por exemplo:

```
$frutas = banana | laranja | morango | cereja | graviola;
```

Neste exemplo, o nome da regra é fruta e a expansão corresponde à lista de nomes de frutas. O nome de uma regra pode ser qualquer seqüência de caracteres que sirva para identificar aquela regra. O nome de uma regra segue as seguintes convenções:

- deve começar com o caractere $;
- deve ser constituída por letras e números somente;
- é *case-sensitive*, ou seja, maiúsculas e minúsculas são considerados caracteres diferentes.

Tokens

Um *token* é uma parte da expansão de uma regra que corresponde a uma palavra que pode ser pronunciada por um usuário. Por exemplo, na regra:

```
$frutas = banana | laranja | morango | cereja | graviola;
```

As palavras "banana", "laranja", "morango", "cereja"e "graviola"correspondem as *tokens* da regra $frutas.

Referência a uma Regra

Atualmente somente regras definidas no escopo local podem ser referenciadas. Referências para regras definidas externamente não são suportadas. Por exemplo, a gramática:

```
$main = $action $object; $action = mover | arrumar | limpar;
$object = cadeira | mesa | cama;
```

Define uma regra, $main, a partir de referências para duas outras regras que foram definidas localmente: $action e $object.

Combinações

Tokens e referências para regras podem ser combinadas de forma a gerar expressões mais complexas. A seguir são apresentados os possíveis tipos de combinações que podem ser utilizados.

- Alternativas - um conjunto de elementos alternativos. Por exemplo:

  ```
  $frutas = banana | laranja | morango;
  ```

  Pode-se selecionar "banana", "laranja"ou "morango".

- Seqüências - um conjunto de elementos que devem ser ditos em uma determinada ordem. Exemplos:

  ```
  $prato = sanduíche de presunto e queijo;     //seqüência de tokens

  $work = $action $object;                     //seqüência de regras

  $remove = levar $object para fora da casa;  //seqüência de regras
  e tokens
  ```

- Agrupamentos - uma forma de tratar um conjunto de elementos como sendo um único termo. Agrupamentos são definidos através do uso de parêntesis: ( ). Por exemplo:

```
$bebida = pepsi | coca cola;
```

Onde são válidos "pepsi"ou "coca cola". A mesma interpretação pode ser obtida através de:

```
$bebida = pepsi | (coca cola);
```

Se os parêntesis fossem colocados da seguinte forma:

```
$bebida = (pepsi | coca) cola;
```

A interpretação é diferente, pois as seqüências válidas agora são "pepsi cola"ou "coca cola".

- Opcional - define referências para regras ou *tokens* como itens opcionais em uma determinada seqüência. Itens opcionais são delimitados por colchetes: [ ]. Por exemplo:

```
$pedido = [por favor] quero uma bebida;
```

Nesse caso, a seqüência "por favor"é opcional, ou seja, nessa regra tanto "por favor quero uma bebida"quanto "quero uma bebida"são válidos.

3.2.4 Interpretação Semântica

A interpretação semântica corresponde a uma *string* que pode ser incluída na seqüência de qualquer regra de expansão. É possível incluir várias interpretações semânticas em uma única sentença. O resultado da interpretação semântica é delimitado por um par de chaves { }. Exemplo:

```
$yesno = $yes {YES} | $no {NO};
$yes = sim | afirmativo | certo |
correto;
$no = não | negativo | errado | incorreto;
```

No caso apresentado acima temos a interpretação semântica YES associada à regra $yes e NO para a regra $no. Desta forma, se a entrada de fala for "sim", "afirmativo", "certo"ou "correto", teremos uma única interpretação semântica, a *string* YES, sendo retornada para todas essas possibilidades.

Um outro exemplo de utilização da interpretação semântica é o seguinte:

```
$objetocolorido = $cor $objeto; $cor =
       (vermelho | rosa)    {cor="vermelho"}
     | (amarelo | laranja)  {cor="amarelo"}
     | (azul | celeste)     {cor="azul"};

$objeto =
         (carro | carreta) {objeto="carro"}
       | (bola | boneca)   {objeto="brinquedo"}
       | (saia | blusa)    {cor="roupa"};
```

Esta gramática reconhece frases como "saia amarela"ou "blusa laranja". Para ambas as frases será retornada a interpretação semântica:

```
{
    cor: amarelo;
    objeto: roupa;
}
```

Este exemplo mostra como é possível fazer com que a gramática aceite sinônimos e retorne um resultado canônico mais fácil de ser tratado.

O uso da interpretação semântica permite que os resultados de reconhecimento possam ser tratados de uma forma mais simples pela aplicação *VoiceXML*.

3.2.5 Sistemas de síntese de fala

Sistemas de *TTS* conseguem sintetizar qualquer texto de entrada. Isso é feito através de um processamento que converte os grafemas [3] em fonemas [4]. Em seguida, esses fonemas são concatenados, gerando um resultado audível.

[3] Cada um dos símbolos gráficos que representam as unidades da palavra escrita.

[4] A mais pequena unidade sonora de uma língua ou dialecto susceptível de ser articulada pelo aparelho fonador.

Não devemos confundir sistema de *TTS* com sistemas que trabalham com a junção e a reprodução utilizando uma base de dados de palavras pré-gravadas. Os sistemas que usam gravações conseguem reproduzir apenas as palavras contempladas por sua base de dados.

O processo de conversão dos grafemas para fonemas é baseado em um conjunto de regras de pronúncia e em um dicionário para as exceções. Entretanto, o conjunto de regras e o dicionário não conseguem contemplar todas as pronúncias existentes em uma língua, podendo gerar, em alguns casos, pronúncias estranhas e até mesmo incorretas.

3.3 Gerenciamento de Recursos

A arquitetura cliente/servidor utilizada na maioria das aplicações distribuídas, torna-se ineficiente quando os servidores são quase sempre fixos em um nó da rede. Sendo a maioria das tarefas executadas localmente na máquina do cliente, com isto pode-se ter uma degradação da qualidade de serviço, pois a máquina do cliente pode não ser a mais apropriada, e também pode existir nós na rede onde essas tarefas possam ser executadas mais eficientemente [MK02].

As alternativas ao gerenciamento de recursos buscam combinar uma coleção de recursos distribuídos de forma totalmente transparente ao cliente, como se esses serviços estivessem disponíveis em uma máquina local. Com essas alternativas podem-se balancear melhor os recursos na rede distribuída, definindo regras de balanceamento que atendam as requisições da *QoS* definidas para cada recurso.

O objetivo principal da utilização de gerenciamento de recursos é possibilitar a execução remota de tarefas em uma coleção de computadores em uma rede local. Para atender esse objetivo, tem-se que implementar um esquema de gerenciamento dinâmico de recursos em um sistema distribuído. Sendo a distribuição desses recursos feita de uma forma transparente aos clientes dos recursos. Na distribuição dos recursos leva-se em consideração as características estáticas e dinâmicas de *hardware* e *software* disponíveis no sistema distribuído.

Os usuários podem especificar os requisitos da utilização desses recursos de *software*. A partir dessas especificações, o sistema encarrega-se em localizar uma máquina apropriada com o recurso disponível e em garantir a QoS adequada para determinada tarefa.

São funções do gerenciamento de recursos:

1. registrar os recursos a ele conectados;
2. enviar parâmetros de configuração para os seus recursos;
3. verificar se as conexões remotas estão ativas e os recursos estão disponíveis, reiniciando os recursos quando necessário;
4. atender requisições por recursos;
5. execução de aplicativos;
6. garantir tolerâncias a faltas;
7. atender aos parâmetros de *QoS*.

3.3.1 Tolerância a Falhas

O conjunto de técnicas para garantir tolerância a falhas tem por objetivo fornecer um serviço que atenda as especificações toleradas para a ocorrência de erros no sistema, através da redundância [Gar99]. As redundâncias agregadas ao sistema, para garantir tolerância a falhas, dizem respeito a redundâncias de *hardware, software* ou através da duplicação de estados sendo executados em processadores distintos [Sch90] [TP00]. Como ponto inicial, tem-se que identificar algumas formas para a detecção de defeitos no sistema. Um defeito é quando o serviço fornecido encontra-se em um estado inconsistente ao especificado, o erro é alguma alteração indesejada a qual pode levar a um defeito e a falha é a causa principal do defeito. Uma forma bastante simples para detecção de erro é aplicar estímulos na entrada do sistema e verificar o resultado obtido na sua saída e compará-lo com o resultado esperado para este mesmo sistema.

O objetivo principal da detecção de erros é abstrair a causa do defeito para o cliente, visando fornecer maior confiabilidade ao sistema.

A aplicação de estratégias de tolerância a falhas, resulta em perda do desempenho do sistema, devido ao tempo gasto para a realização de testes, comparação de resultados e/ou sincronização das tarefas.

3.3.2 Qualidade de Serviço

Qualidade de Serviço (QoS) trata-se de uma política para dividir mais racionalmente a banda disponível e garantir para certos serviços a latência e largura de banda necessária,

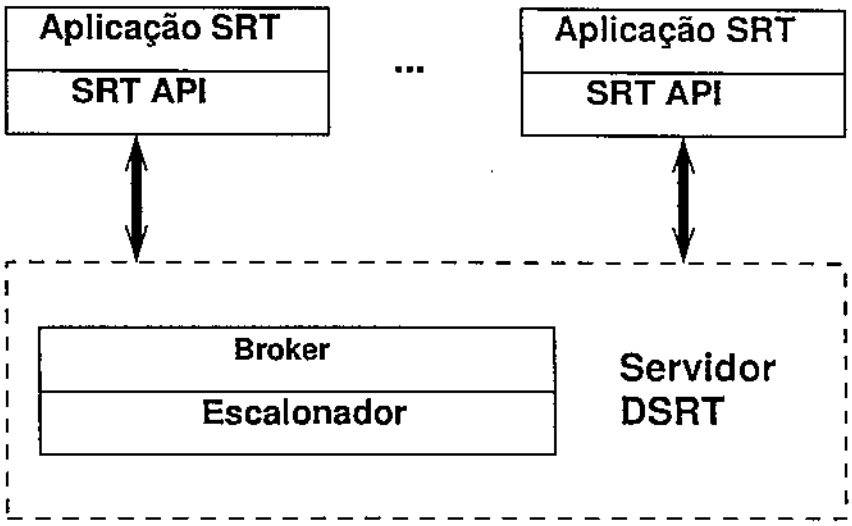


Figura 3.7: Arquitetura do DSRT

mesmo em sobrecarga do sistema, pois aqueles serviços definidos como prioritários deverão continuar funcionando perfeitamente, mesmo que isto prejudique o desempenho de outros de menor importância.

A qualidade de serviço é garantida através da alocação de recursos do sistema, esses recursos devem ser alocados nos pontos terminais e intermediários da conexão aonde a informação irá passará. Podemos ter três formas distintas de abordar a qualidade de serviço:

1. O valor máximo de um determinado tipo de recurso é reservado de forma permanente para a conexão durante todo o seu período de atividade, garantia determinística;

2. Um valor médio de um determinado tipo de recurso é reservado para a conexão durante todo o seu período de atividade. Esta aproximação usa o conceito de multiplexação estatística de modo a aumentar a taxa de utilização dos recursos, garantia estatística;

3. Nenhum valor de recurso é reservado para as conexões. Este esquema é conhecido como melhor esforço e é o esquema adotado pelos protocolos mais comuns como o TCP.

Dynamic Soft Real Time CPU Scheduler

O DSRT possui as seguintes funcionalidades:

- Garante alocação de compartilhamento de tempo (*TS - Time Sharing*);
- Implementa estratégias de adaptação para ajuste automático da reserva de recursos;
- Disponibiliza mecanismos para renegociação da reserva de recursos.

A versão do DSRT/NT é um arcabouço *middleware* para capacitar o *Microsoft Windows NT* em tempo real suave. O DSRT habilita o sistema para testes de requisições de reserva de processador e também adaptação de reserva. Para fazer o escalonamento de várias tarefas em tempo real suave e de tarefas de melhor esforço, o DSRT utiliza um algoritmo baseado no EDF, o R-EDF (*Reservation-based preemptive EDF*) [WY01].

A arquitetura do DSRT é mostrada na Figura 3.7. O DSRT utiliza uma API (*Application Program Interface*) para requisitar reserva, consultar estado da reserva dos processos, investigar reserva de recursos, fazer reserva de recursos e adaptação. O *broker* interage com a aplicação em tempo real suave, recebendo requisições e retorna se é possível ou não a requisição. O escalonador faz controle de admissão na reserva e escalona aplicações em tempo real suave e melhor esforço.

3.3.3 Funcionamento da Infra-Estrutura do Portal de Voz

Nesta seção apresenta-se como os componentes interagem para prover as funcionalidades necessárias.

A interação entre os componentes é dividida em duas partes: uma voltada para o processo de inicialização do sistema, e outra para os processos envolvidos no atendimento de uma requisição de serviço.

Processo de inicialização dos aplicativos

A inicialização da estrutura do portal de voz ocorre em duas etapas: a primeira envolve a passagem de parâmetros de configuração para todos os computadores que fazem parte da plataforma. A segunda inicia a execução e configura os aplicativos que serão executados em cada um dos computadores.

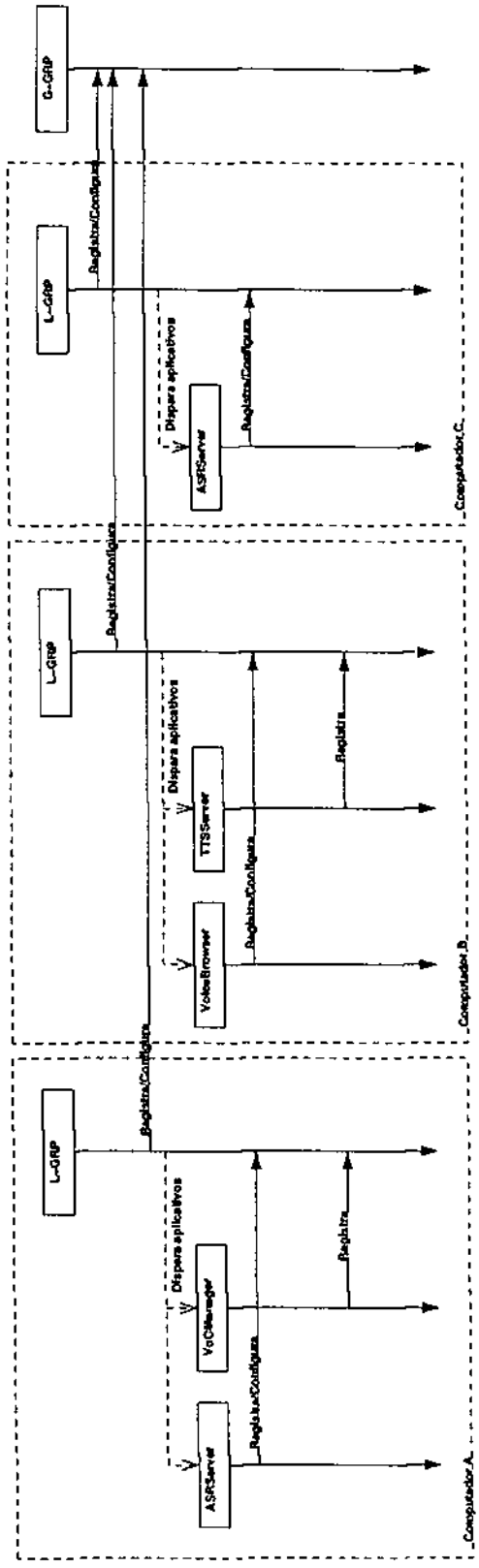


Figura 3.8: Inicialização da estrutura do portal de voz distribuído

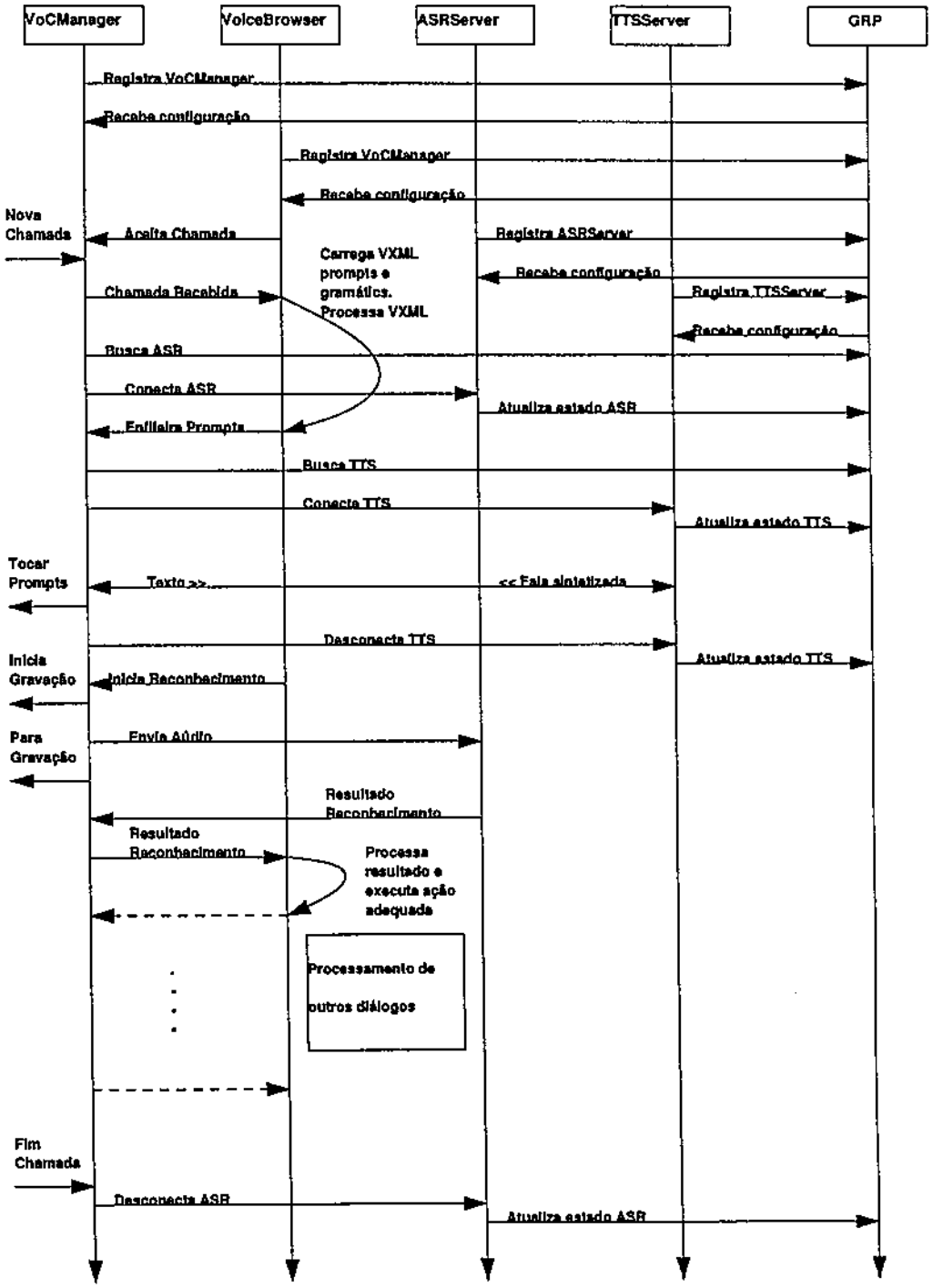


Figura 3.9: Seqüência de ações executadas para o atendimento de uma chamada

A primeira etapa da inicialização entre os gerenciadores locais e o global é definida quando o G-GRP inicia a execução, este carrega um arquivo de configuração contendo os parâmetros de inicialização de todos os aplicativos que serão executados para dar o suporte ao portal de voz, veja Figura 3.8. Quando cada L-GRP inicia a execução, este se conecta ao G-GRP para obter as informações de como configurar os aplicativos que serão executados localmente.

A segunda etapa da inicialização entre o gerenciador local e os aplicativos locais ocorre quando cada L-GRP utiliza as configurações recebidas do G-GRP para identificar quais são os aplicativos que devem ser executados localmente. Após cada aplicativo ter sido iniciado localmente, este deve se conectar ao seu gerenciador local presente naquele computador e obter os parâmetros de configuração específicos, veja Figura 3.9. Assim que cada aplicativo termina o processo de configuração, eles devem sinalizar ao L-GRP que estão prontos para atenderem novas requisições, especialmente os servidores de *ASR (ASRServer)* e servidores de *TTS (TTSServer)*, através da atualização do seu estado. Por fim, o G-GRP recebe informações sobre todos os aplicativos que estão sendo executados.

Quando um *VoCManager*[5] (Seção 3.1.1) precisa de um *ASRServer* ou um *TTSServer*, este consulta o L-GRP. Se existir um servidor disponível localmente para atender essa requisição, então o L-GRP entrega esse recurso para o *VoCManager*. Caso não exista esse recurso localmente, o L-GRP redireciona esta requisição para o G-GRP que consulta uma lista com todos os recursos disponíveis. Se não existir um recurso disponível, isso resultará em um erro de execução no *VoCManager*.

Durante a execução, o L-GRP monitora o funcionamento dos aplicativos locais. Se um dos aplicativos parar de funcionar, este evento é detectado. O gerenciador local notificará a ocorrência desse evento ao gerenciador global, e buscará finalizar o aplicativo com problemas e, em seguida, se possível, reinicia a execução da aplicação. O aplicativo executará todo o processo para obter os parâmetros de configuração da mesma forma como foi apresentado anteriormente.

Processo de atendimento de uma chamada

Para entender a interação entre os componentes utiliza-se, como exemplo, um processo de atendimento e estabelecimento de um diálogo.

[5]O gerenciador de aplicativos está acoplado ao *software* da infra-estrutura de telefônia.

Na Figura 3.9, apresenta-se as seqüências das ações executadas para o atendimento de uma chamada, envolvendo as etapas de síntese de fala, reconhecimento de fala, processamento de diálogos, e finalização da chamada.

Uma determinada chamada só pode ser atendida depois que o *VoiceBrowser* [6] sinaliza para o *VoCManager* essa possibilidade. Antes disso, mesmo que o *VoCManager* esteja ativo, as chamadas realizadas não são atendidas.

Quando o *VoCManager* recebe uma nova chamada, este evento é sinalizado para o *VoiceBrowser* que dispara uma seqüência de ações, a saber:

- O *VoiceBrowser* inicia o processo de carregamento e interpretação das páginas *VoiceXML*. Como resultado dessa interpretação, arquivos de áudio e gramáticas são carregadas iniciando-se o processo de execução do conteúdo da página;
- O *VoCManager* consulta o seu L-GRP em busca de um *ASRServer* para atender a chamada. Em seguida ele se conecta ao *ASRServer*, e com ele os comandos de fala do usuário serão interpretados;
- O *VoCManager* recebe do *VoiceBrowser* uma série de arquivos de áudio para serem tocados. Estes arquivos de áudio são enfileirados e tocados em seqüência;
- O processo de reconhecimento é disparado pelo *VoiceBrowser* que sinaliza para o *VoCManager* que este deve começar o processo de aquisição de áudio;
- O resultado do reconhecimento inicia uma série de ações que podem ser desde tocar um arquivo de áudio para o usuário, como realizar o carregamento de uma outra página *VoiceXML*;
- Uma ligação pode ser finalizada de duas formas: pelo usuário ou pelo sistema. Caso o usuário desligue, esse evento é capturado pelo *VoCManager*. Caso o sistema finalize a ligação, o *VoiceBrowser* sinaliza para o *VoCManager* que a ligação deve ser finalizada;
- Quando uma ligação é finalizada, o *VoCManager* finaliza a execução corrente, libera o canal de *ASRServer* que estava sendo ocupado e entra no modo de espera por uma nova chamada. Nesse estado de espera, nenhum recurso de *ASR* ou de *TTS* é utilizado.

[6] Este aplicativo representa o navegador de *VoiceXML*.

Processo de atualização de estado

O GRP deve ter informações atualizadas sobre a utilização dos recursos. Essas informações são enviadas dos recursos ao gerenciador local, e repassados posteriormente ao gerenciador global. Entretanto, para evitar a sobrecarga e um fraco desempenho da infra-estrutura que comprometa a escalabilidade do sistema, adotou-se uma estratégia de modo que os gerenciadores não mantivessem informações precisas do estado atual do agrupamento, mas sim uma idéia aproximada do estado global do sistema, e da carga de processamento alocados nas máquinas [MK02].

Para verificar se os recursos estão disponíveis, o gerenciador aguarda mensagens de atualização de estados por um período de tempo definido pelo programador. Esta mensagem é utilizada pelo gerenciador para obter o estado global do sistema, veja Figura 3.9. Caso o recurso não envie uma mensagem de atualização no tempo determinado, o Módulo de Administração de Processos (veja Seção 3.1.4) é instruído a averiguar o estado atual do processo que provê o recurso. Para mais esclarecimentos sobre os estados dos recursos, veja a Seção 4.4. Caso o processo esteja respondendo, o gerenciador aguarda a sua mensagem de atualização de seu estado. Caso este atraso prolongue-se por muito tempo, o recurso é reiniciado. Caso o processo não esteja respondendo, o processo é reiniciado.

Para o gerenciador global, os gerenciadores locais também enviam mensagens de atualização dos seus estados. Caso essa mensagem atrase, o gerenciador global coloca o estado do gerenciador local como indisponível para o sistema até que o gerenciador global receba uma mensagem de atualização.

Capítulo 4

Implementação do GRP

O GRP foi desenvolvido inicialmente para prover toda a infra-estrutura de gerenciamento e controle do portal de voz do Genius Instituto de Tecnologia. O GRP foi desenvolvido em linguagem de programação C++ para ambiente operacional *Microsoft Windows*. Além disso, desenvolveram-se todas as aplicações do portal de voz. Neste capítulo são apresentadas e discutidas as classes principais que são utilizadas para fornecer controle de tarefas e sincronização, troca de mensagens usando TCP/IP e outras funções de uso geral pela estrutura do GRP.

4.1 Estrutura de Base

Para o armazenamento de dados dos recursos como o seu estado atual (veja Figura 4.11) ou informações de configuração e de controle dos gerenciadores e dos componentes do portal de voz, utilizamos uma estrutura denominada de quadro negro, como detalhado na Seção 3.1.2. Deve-se observar que a estrutura apresentada é compartilhada por diversos fluxos de execução no gerenciador, veja na Figura 4.1 que o Módulo de Suporte a Aplicação (*ASModule*) e o Módulo de Administração de Processos (*ProcessManager*) acessam as informações do quadro negro.

O portal de voz compartilha uma biblioteca que disponibiliza as classes para o sistema. Esta biblioteca contém ferramentas para leitura, escrita e registro de áudio, registro do funcionamento do sistema, geração e controle de fluxo TCP/IP e encapsulamento de informações. Esta biblioteca é denominada de *VoPBase*, pois é a biblioteca que fornece todas as funcionalidades básicas para o Portal de Voz.

Para a comunicação entre os módulos do sistema utilizamos o mapeamento de

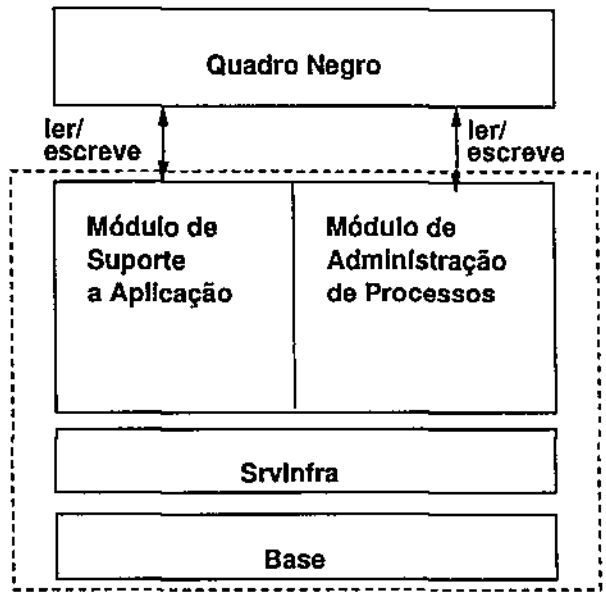


Figura 4.1: Quadro negro sendo acessado pelos dois fluxos principais de execução

parâmetros de configuração e estados através de uma classe de encapsulamento de dados denominada de *CMapProperties*. Para a leitura de arquivos de configuração utiliza-se métodos para realizar o *parsing* dos dados a partir de arquivos texto e repassá-las ao *CMapProperties*.

Uma outra estrutura compartilhada no portal de voz é a *Server Infrastruture (SrvInfra)*. Esta estrutura fornece ferramentas que possibitam a configuração, inicio e finalização da aplicação e controle das diferentes linhas de controle. Esta estrutura é composta de classes abstradas, e pode ser especializada para necessidades diferentes das aplicações.

A estrutura do *SrvInfra* é instanciada pelo módulo TCP/IP do gerenciador para promover a sincronização das mensagens enviadas e recebidas pela rede. Computam também o estado global do servidor quando as mensagens são processadas ou perdidas. Além disso, criam contextos diferentes para as conexões TCP/IP, e mecanismos de tratamento e sincronização desses contextos. O GRP global e o local utilizam estruturas semelhantes para o gerenciamento e o escalonamento dos vários contextos para as conexões. A diferença é nas mensagens recebidas por cada gerenciador e no tratamento que é designado para cada mensagem.

O suporte de rede utiliza estruturas presentes no *VoPBase*. A classe principal deste estrutura é denominada de *CSocketComm*, e as outras classes a instanciam: a *CSoc-*

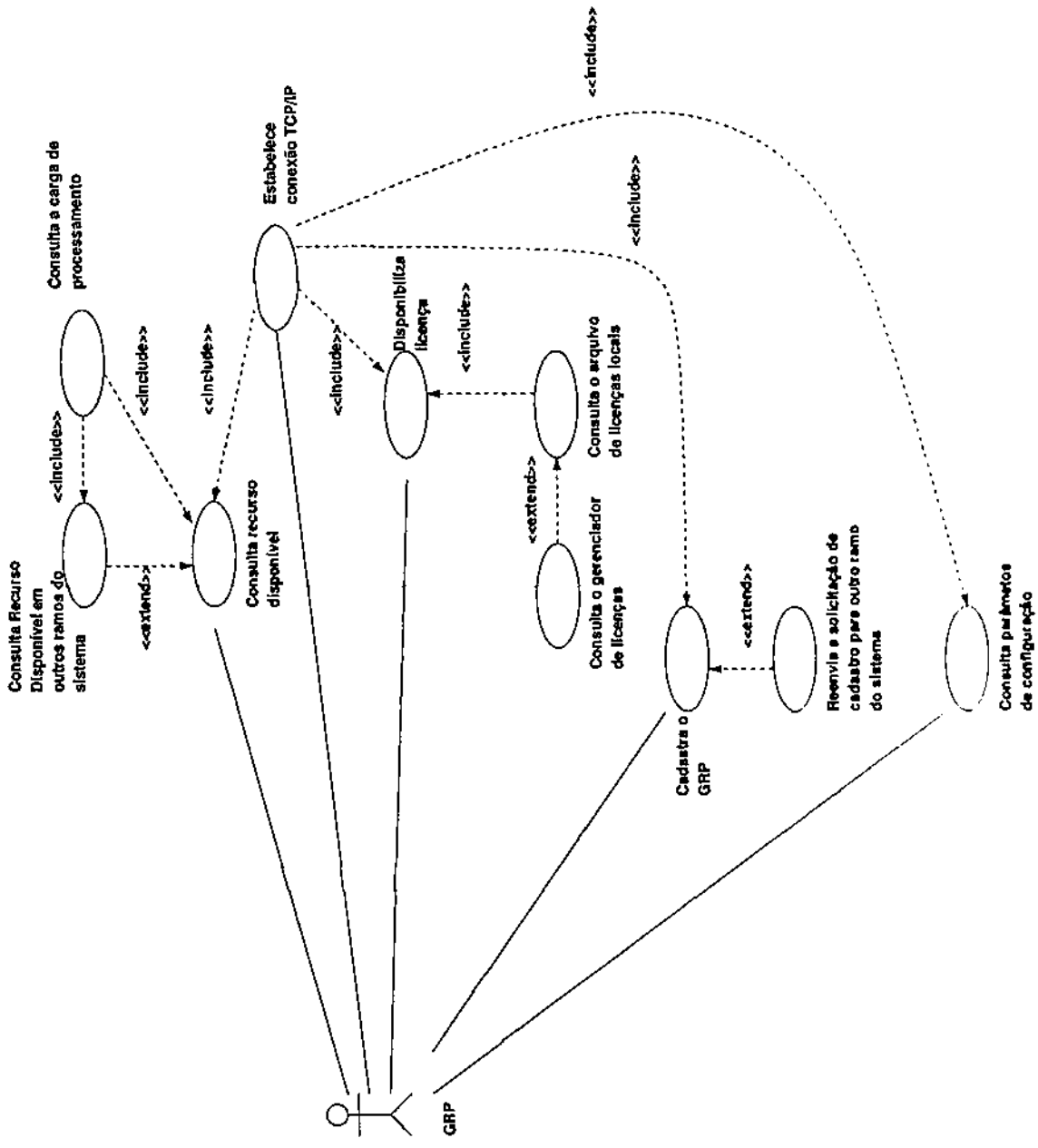


Figura 4.2: Caso de caso para as conexões remotas entre os Gerenciadores

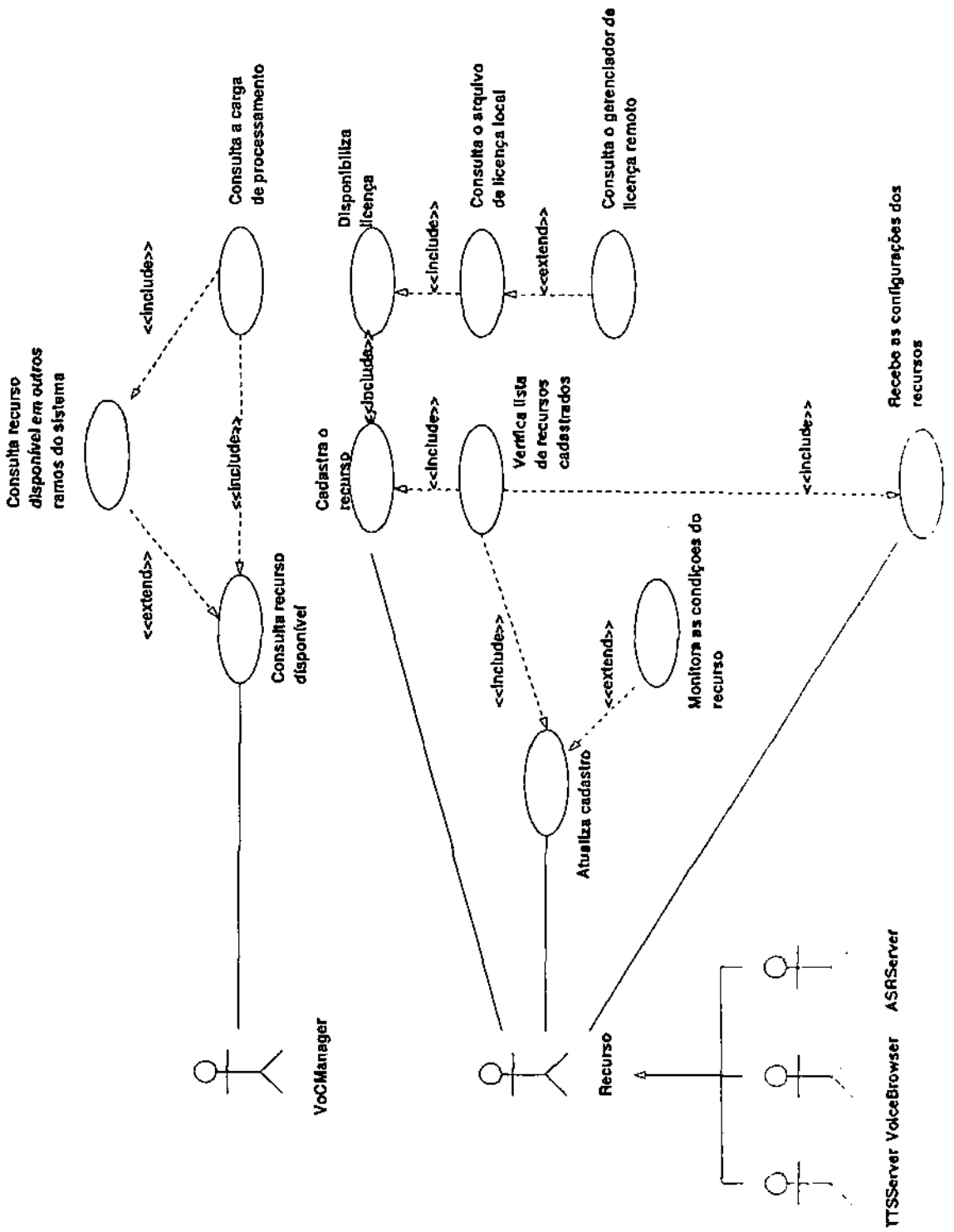


Figura 4.3: Caso de uso para as conexões locais entre os recursos e o gerenciador

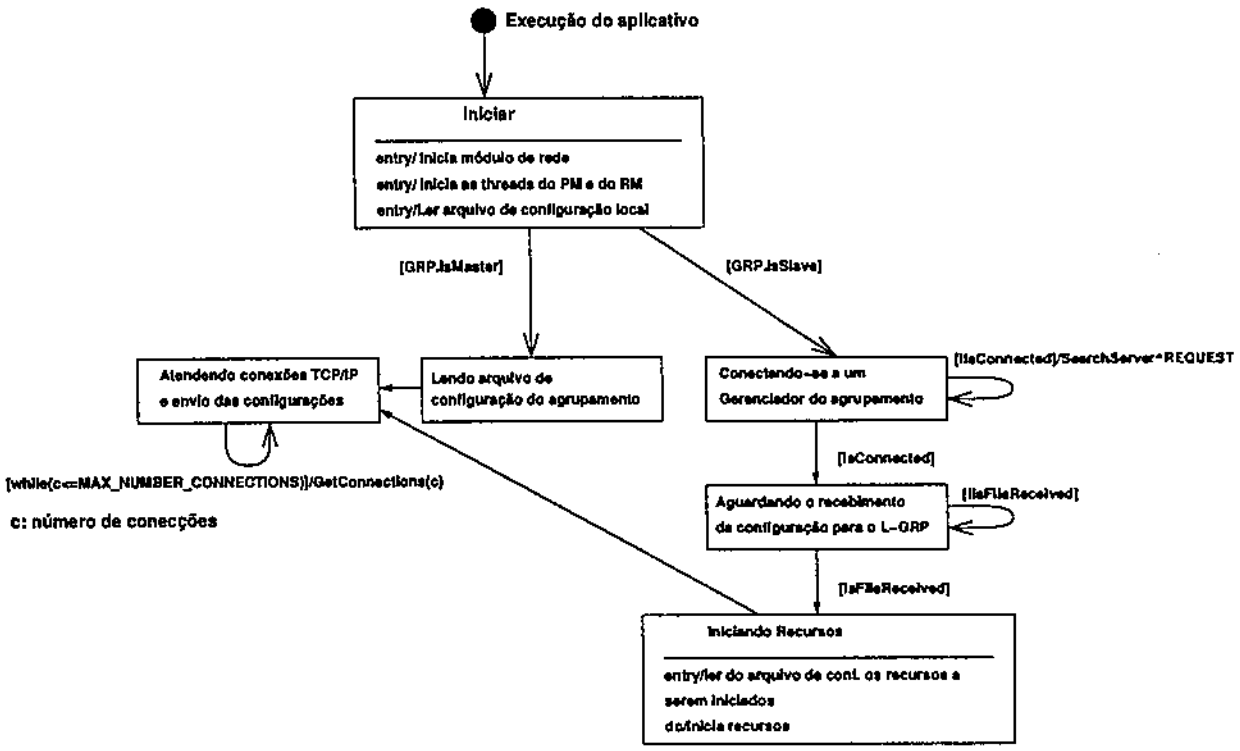


Figura 4.4: Diagrama de Atividade para o carregamento da configuração do sistema

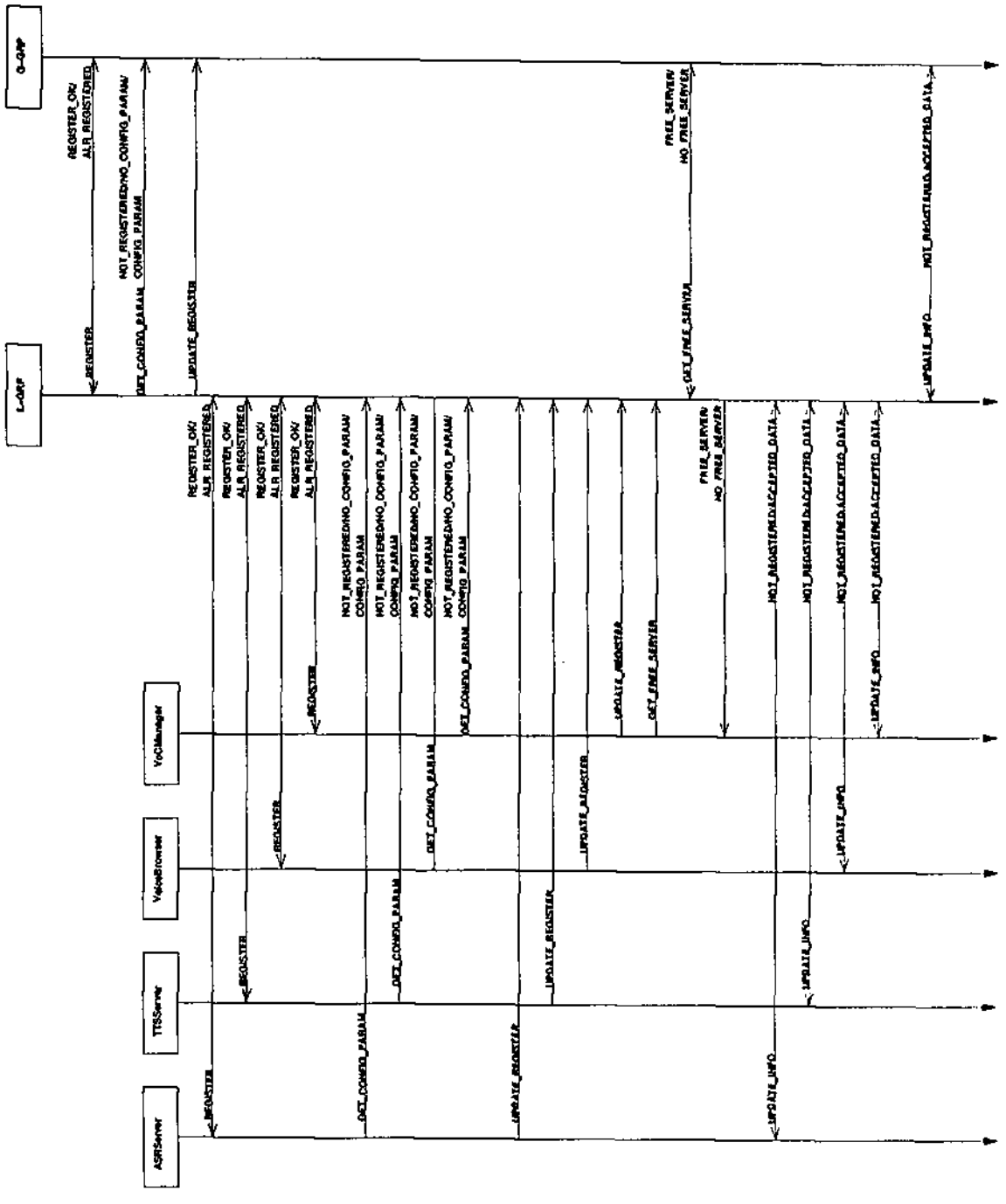


Figura 4.5: Diagrama de seqüência de mensagens

ketServer fornece mecanismos para atendimento das conexões de soquete dos clientes, e então são tratadas pelo *CSocketUser* quando as conexões são efetuadas. O *CSocketUser* implementa funções específicas que são usadas pelo soquete que atenderá as requisições TCP/IP dos clientes; O *CSocketClient* implementa funções que permite um cliente efetuar uma conexão a um servidor de soquete. A seriação e deseriação das informações são efetuadas através de uma estrutura denominada de *CMessage*, esta estrutura possibilita a troca de mensagens através do armazenamento assíncrono dos dados.

4.2 Diagramas

Os diagramas são meios utilizados para a visualização dos blocos de um *software*. Sendo uma representação gráfica de um conjunto de elementos, geralmente representados como um gráfico conectados de vértices(itens) e arcos(relacionamentos). Os diagramas fornece uma visualização do sistema sob diversas perspectivas. A UML (*Unified Modeling Language*) é uma linguagem de modelagem que define um número de diagramas que permite dirigir o foco para a representação conceitual e física de um sistema [BRJ98]. Nesta seção apresenta-se os diagramas de caso de uso fornecendo a visualização dinâmica do sistema, e os de classes fornecendo a visualização estática do sistema.

Nas outras seções apresenta-se alguns diagramas de atividades fornecendo a dinâmica de funcionamento do GRP.

4.2.1 Caso de Uso

Um caso de uso especifica o comportamento de um sistema ou de parte de um sistema e é uma descrição de um conjunto de seqüências de ações, incluindo variantes realizadas pelo sistema para produzir um resultado observável do valor de um ator [BRJ98].

Aplicou-se o caso de uso para capturar o comportamento pretendido do gerenciador de recursos e processos que foi desenvolvido, sem ser necessário especificar como esse comportamento é implementado. Dividiu-se os casos de uso em dois, denotando somente o comportamento essencial do sistema: caso de uso para as conexões locais (Figura 4.3) e o caso de uso para as conexões remotas (Figura 4.2). Através desses dois casos de uso pode-se identificar os requisitos necessários para definir o comportamento

desejado para o sistema. O estudo de caso para as conexões locais define a interação entre os recursos e um gerenciador local, e o estudo de caso para as conexões remotas define a interação entre dois gerenciadores.

4.2.2 Diagrama de Classes

Nos diagramas de classes pode-se visualizar um conjunto de classes, interfaces e colaborações e os seus relacionamentos. Os diagramas de classes são usados para fazer a modelagem estática do projeto de um sistema.

Na Figura 4.6 apresenta-se o diagrama de classes para o GRP, os relacionamentos do GRP com o *VoPBase* e o *SrvInfra* é apresentado na Figura 4.7. Para maiores detalhes do *SrvInfra* e do *VoPBase*, apresenta-se nas Figuras 4.8 e 4.9 os diagramas de classes e os seus relacionamentos.

4.3 Processo de inicialização do GRP

O GRP é disponibilizado em uma rede de computadores. Sendo está rede estruturada em agrupamentos hierárquicos, onde um gerenciador global é responsável pela configuração geral deste agrupamento e os gerenciadores locais são responsáveis pela gerencia dos recursos da sua máquina local. Em uma única máquina podemos ter um gerenciador global e/ou local. Na Figura 4.4 temos o diagrama de atividades para o carregamento da configuração para os gerenciadores globais e locais. Neste diagrama temos a diferença quando o GRP é global (*isMaster*) ou local (*isSlave*).

A inicialização do portal de voz ocorre em duas etapas: a primeira envolve a passagem de parâmetros de configuração para todos os computadores que fazem parte da plataforma; a segunda inicia e configura os aplicativos que serão executados em cada um dos computadores.

A primeira etapa da inicialização: Quando o GRP global é iniciado, este carrega um arquivo de configuração contendo os parâmetros de inicialização de todos os aplicativos que serão executados para dar o suporte para o portal de voz. Quando cada GRP local é disparado, este se conecta ao seu gerenciador global para obter as informações necessárias dos aplicativos que serão executados localmente e como configurá-los.

A segunda etapa da inicialização: Cada GRP local utiliza as configurações recebidas do gerenciador global para iniciar os aplicativos locais da máquina. Após cada

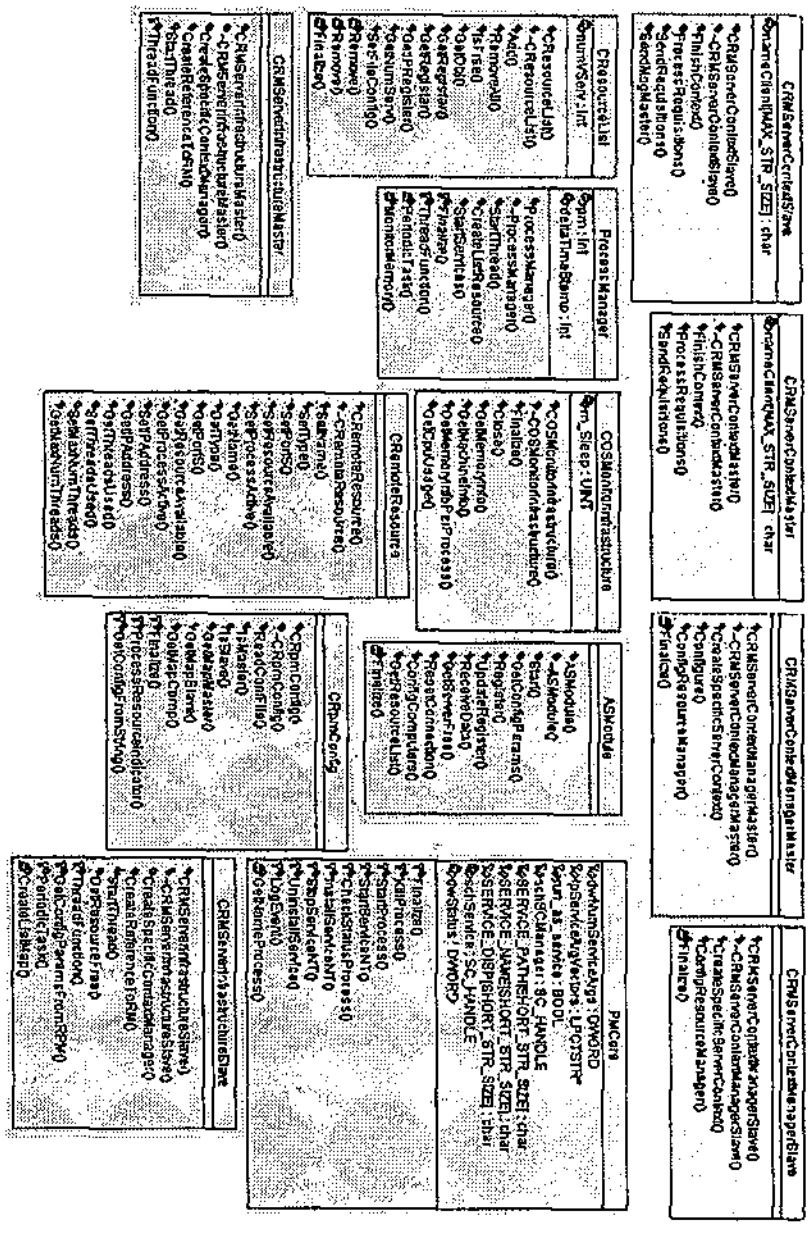


Figura 4.6: Diagrama de classes para o GRP

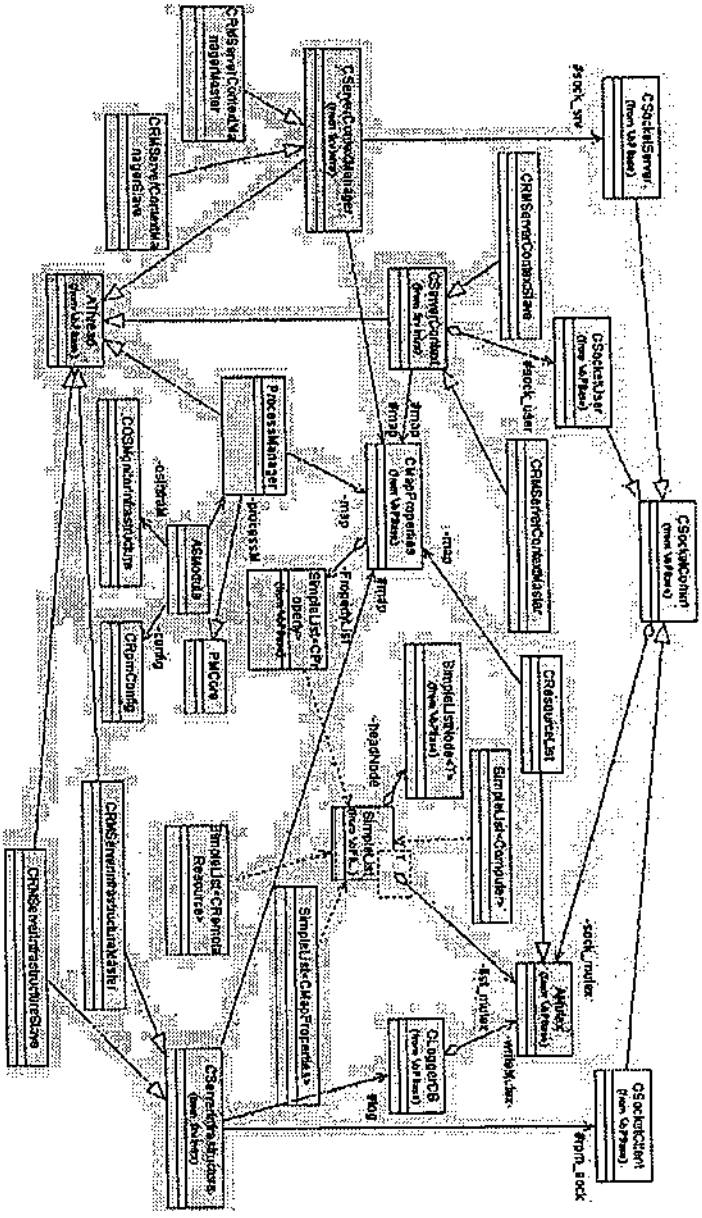


Figura 4.7: Diagrama de classes para o GRP com os seus relacionamentos

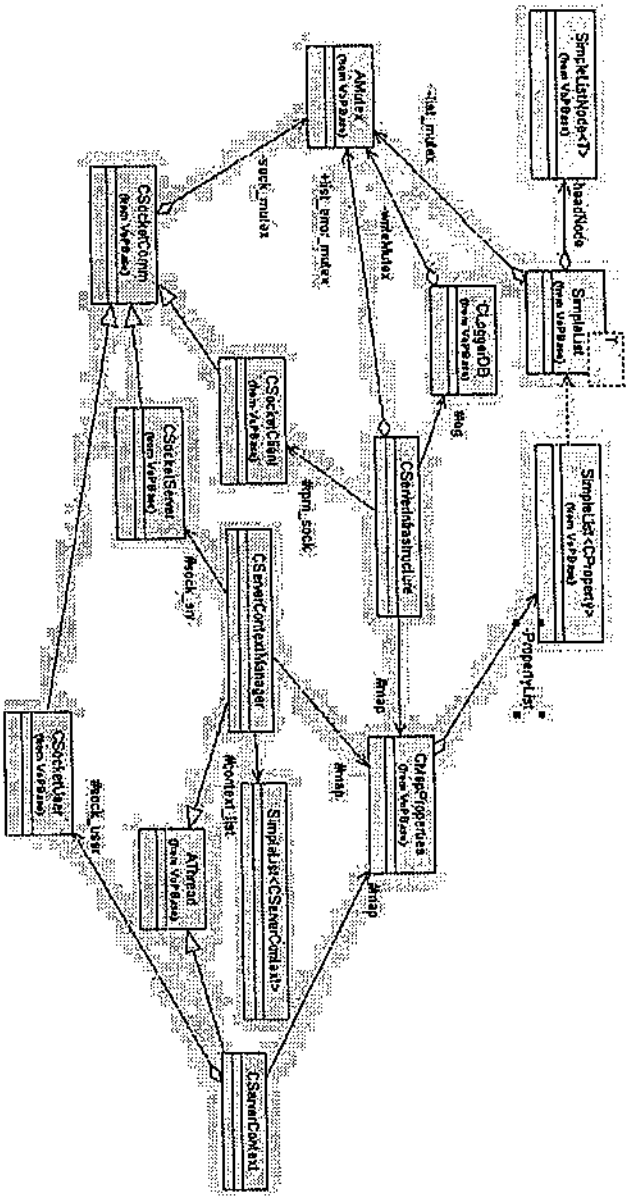


Figura 4.8: Diagrama de classes para o SrvInfra com os seus relacionamentos

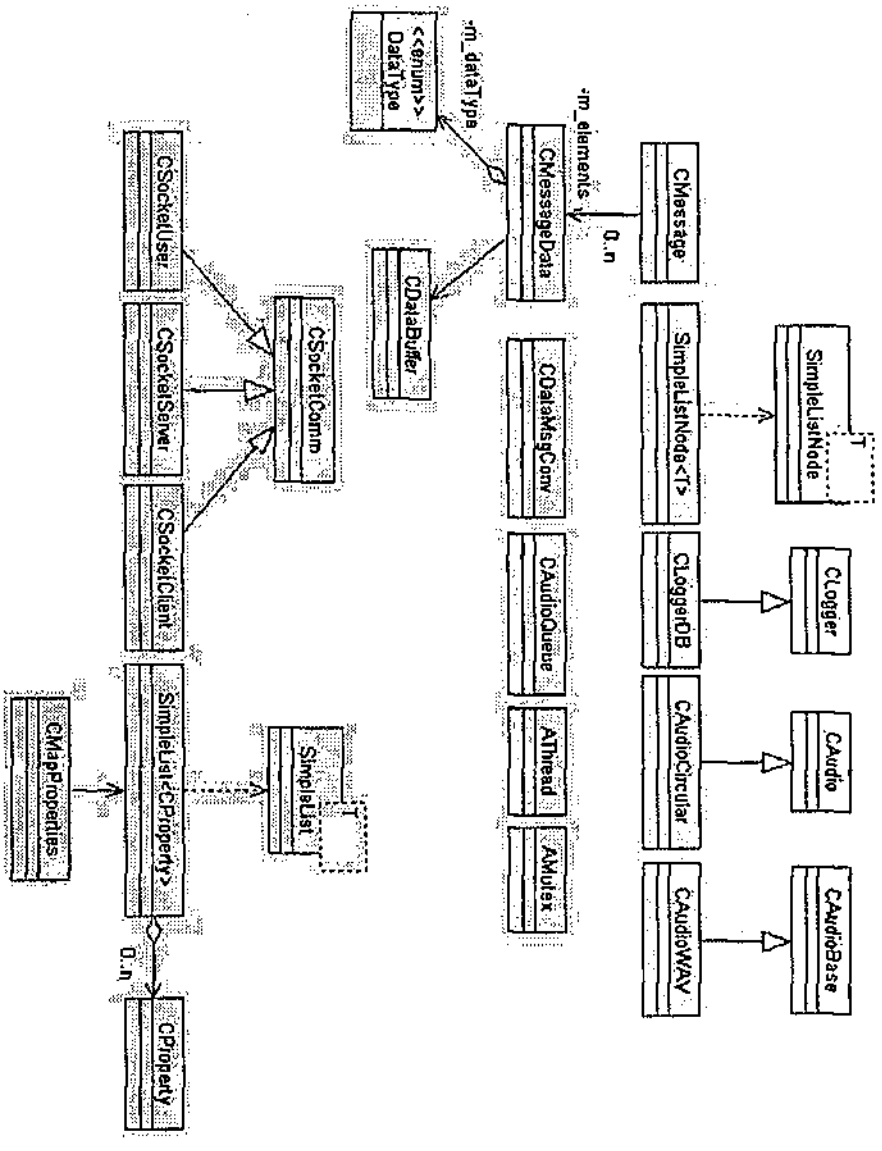


Figura 4.9: Diagrama de classes para o VoPBase com os seus relacionamentos

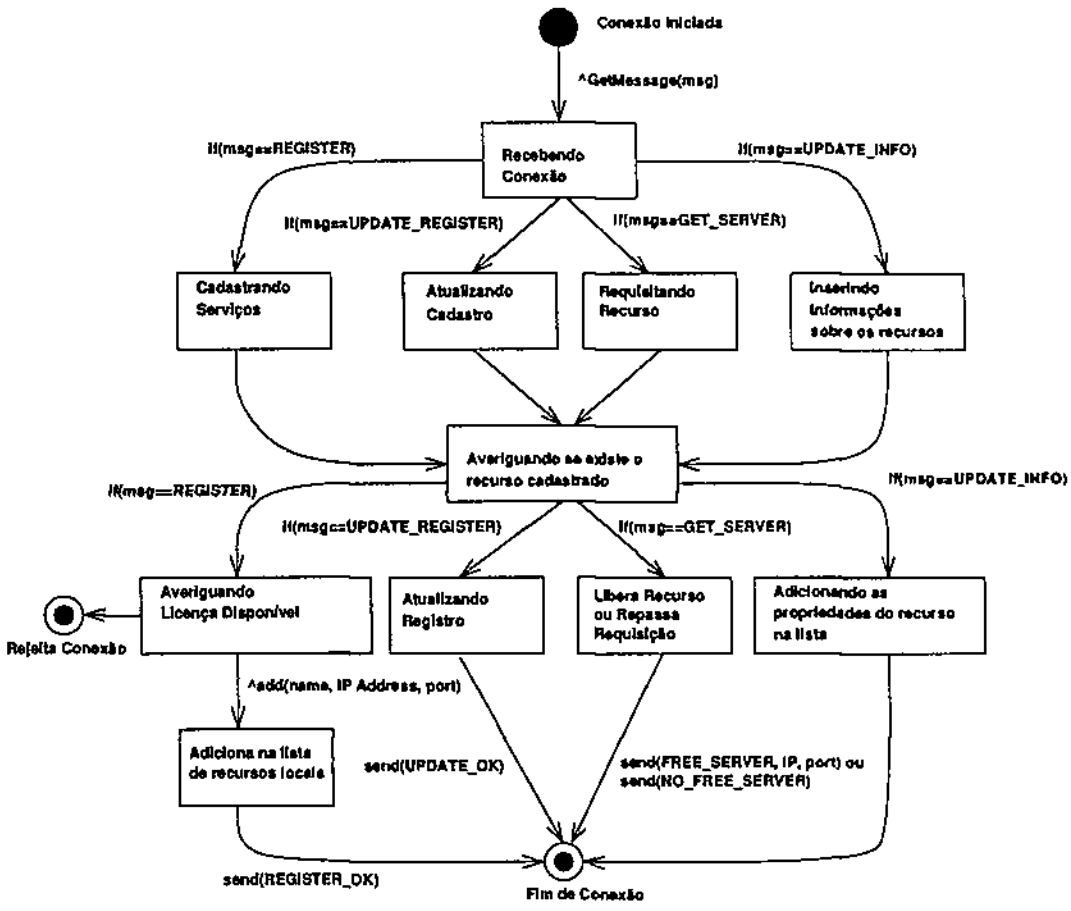


Figura 4.10: Diagrama de atividade para o registro, atualização do registro e dados

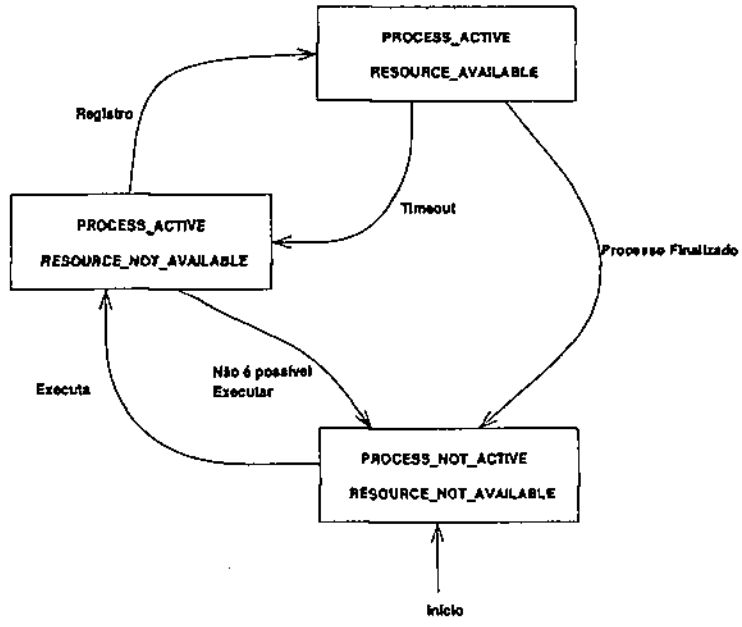


Figura 4.11: Estados dos Recursos

aplicativo ter sido iniciado localmente, este deve se conectar ao seu gerenciador local e obter os parâmetros de configuração específicos.

4.4 Tipos de Mensagens

As mensagens enviadas ao GRP são encapsuladas em uma estrutura *CMessage* e encaminhada via TCP/IP ao GRP Local ou Global. Os identificadores de cada mensagem são: *REGISTER, UPDATE_REGISTER, UPDATE_INFO, GET_FREE_SERVER.* Na Figura 4.5 observa-se um diagrama de seqüencia do fluxo de informações entre os recursos e o gerenciador local e este com o gerenciador global.

1. *REGISTER* - Mensagem enviada ao GRP quando o recurso pretende se registrar. Esta mensagem é composta de quatro parâmetros: identificador da mensagem, endereço de IP e número da porta. Na Figura 4.10 temos o diagrama de atividade para o registro;

2. *UPDATE_REGISTER* - Mensagem enviada ao GRP para a atualização do registro do recurso. Esta mensagem é composta de três parâmetros: identificador da

mensagem e nome do recurso. Na Figura 4.10 temos o diagrama de atividade para a atualização do registro;

Na Figura 4.11 apresenta-se um diagrama de estados que os recursos monitorados podem assumir:

(a) *PROCESS_NOT_ACTIVE,RESOURCE_NOT_AVALIABLE* - Estado inicial;

(b) *PROCESS_ACTIVE, RESOURCE_NOT_AVALIABLE* - O recurso é executado pelo gerenciador;

(c) *PROCESS_ACTIVE, RESOURCE_AVALIABLE* - O recurso realiza o registro no gerenciador.

Quando um recurso não faz a atualização do seu registro em tempo hábil, o recurso entra em estado de recurso não disponível, retornado ao estado (*PROCESS_ACTIVE,RESOURCE_NOT_AVALIABLE*). Neste caso, o "Módulo de Administração de Processos´´ é acionado para averiguar se o recurso realmente não está disponível, verificando se o recurso está ativo ou não. Caso o recurso não esteja ativo, o recurso volta ao seu estado inicial. A partir deste instante o "Módulo de Administração de Processos´´ destrói o processo ou para o serviço, e em seguida tenta reiniciá-lo;

3. *UPDATE_INFO* - Mensagem enviada ao GRP quando o recurso envia informações específicas da sua condição, como: identificação do processo, número de *threads* em execução, etc. Esta mensagem é composta de diversos parâmetros: identificador da mensagem, nome e parâmetros que dependem do tipo de recurso registrado. Na Figura 4.10 temos o diagrama de atividade para a atualização das informações dos recursos;

4. *GET_CONFIG_PARAM* - Mensagem enviada ao GRP quando o recurso requisita parâmetros de configuração. Esta mensagem é composta de três tipos de parâmetros: identificador da mensagem, tipo do recurso e o endereço de IP no caso de gerenciadores locais;

5. *GET_FREE_SERVER* - Mensagem enviada ao GRP quando uma cliente requisita um recurso disponível. Este recurso pode estar localmente ou em uma rede de computadores. Esta mensagem é composta por três parâmetros: identificador da mensagem, tipo do recurso requisitado.

As mensagens que o GRP retorna são: *REGISTER_OK, ALR_REGISTERED, UPDATE_OK, NOT_REGISTERED, ACCEPTED_DATA, FREE_SERVER, NO_FREE_SERVER*, CONFIG_PARAM, NO_CONFIG_PARAM.

1. *REGISTER_OK* - Mensagem retornada do GRP quando o registro do recurso foi bem sucedido. Esta mensagem é composta de dois parâmetros: identificador da mensagem, e o nome do recurso dado pelo GRP;

2. *ALR_REGISTERED* - Mensagem retornada do GRP para avisar que o recurso já está registrado. Esta mensagem é composta somente pelo identificador da mensagem;

3. *NOT_REGISTERED* - Mensagem retornada do GRP para avisar que o recurso ainda não foi registrado. Esta mensagem é composta somente pelo identificador da mensagem;

4. *ACCEPTED_DATA* - Mensagem retornada do GRP para avisar que os dados do *UPDATE_INFO* foram aceito. Esta mensagem é composta somente pelo identificador da mensagem;

5. *FREE_SERVER* - Mensagem retornada do GRP para avisar que existe um recurso livre e disponível Esta mensagem é composta de três parâmetros: identificador da mensagem, nome do recurso disponível, endereço IP e porta;

6. *NO_FREE_SERVER* - Mensagem retornada do GRP para avisar que não existe nenhum recurso livre ou disponível no momento. Esta mensagem é composta somente pelo identificador da mensagem.

7. *CONFIG_PARAM* - Esta mensagem é composta pelo identificador da mensagem e os parâmetros de configuração do recurso que o solicitou;

8. *NO_CONFIG_PARAM* - Mensagem retornada do GRP para avisar que não existe nenhum parâmetro de configuração para o recursos chamador. Esta mensagem é composta somente pelo identificador da mensagem.

Capítulo 5

Resultados Experimentais

Neste capítulo são apresentados resultados experimentais para validação do Gerenciador de Recursos e Processos. A abordagem utilizada foi a definição e execução de testes para um conjunto de máquinas em uma rede de computadores.

5.1 Ambiente de Teste

Esses experimentos foram realizados nos laboratórios do Genius Instituto de Tecnologia com no máximo quatro máquinas conectadas a um *HUB*[1] de 100 Mbps, e todas executando o GRP.

As máquinas tem as seguintes configurações, veja Figura 5.1:

- Máquina 1 e Máquina 2: Windows 2000 Professional, Processador Pentium III 866 MHz e 256 MB de RAM;
- Máquina 3: Windows XP Home, Processador Celeron 1.2 GHz e 120 MB RAM;
- Máquina 4: Windows 2000 Professional, Processador Pentium III 866 MHz e 128 MB RAM.

A máquina 1 executa o G-GRP e todas as outras o L-GRP. A máquina 4 é a única executando o serviço de telefonia, pois a mesma possui uma placa de telefonia digital *Dialogic Springware* [Int02a] [Int03]. Na realização desse experimento as máquinas estavam sendo utilizadas exclusivamente para esta tarefa.

[1]Um *HUB* é usado para conectar computadores. Sendo responsável pela troca de mensagens entre outros *HUBs* e computadores

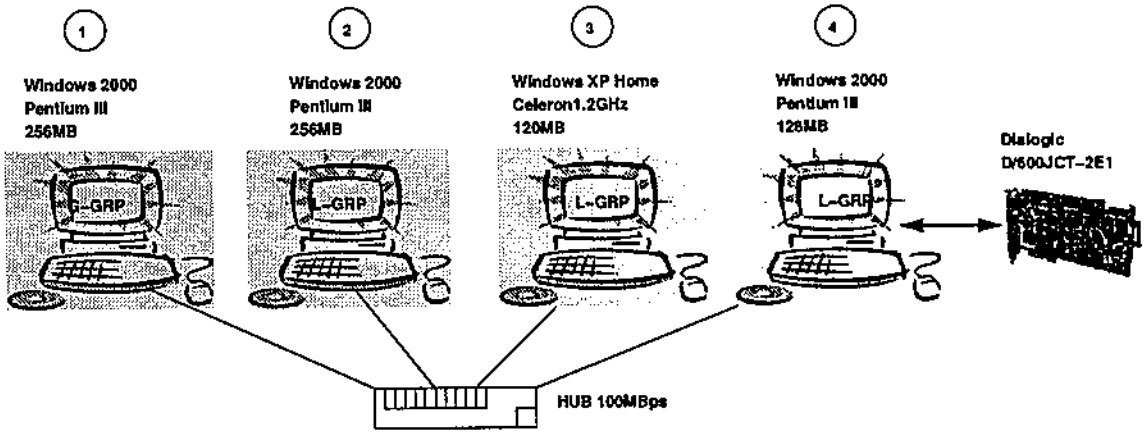


Figura 5.1: Configuração das máquinas no agrupamento

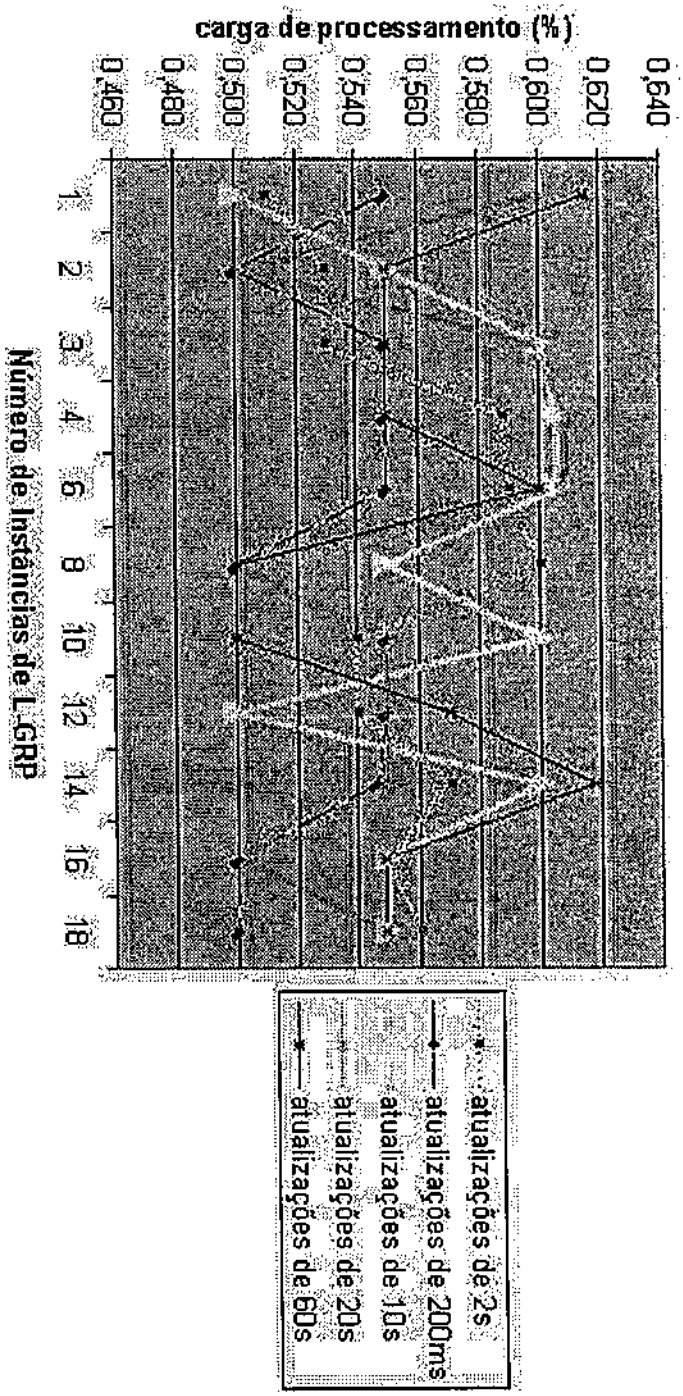


Figura 5.2: Carga de processamento requerida pelo gerenciador dependendo do número de gerenciadores no agrupamento

5.2 Testes Realizados

Os dados de teste foram obtidos a partir de uma monitoração periódica do sistema. A utilização de CPU e memória foram feitas com atendimento e sem nenhum atendimento de chamada do usuário.

O nosso objetivo é simular as condições operacionais e analisar os impactos no desempenho em uma rede local, uma vez que a carga da rede e no GRP seria praticamente a mesma. Analisando a sobrecarga causada por G-GRP e L-GRP em um sistema distribuído e estabelecendo um ambiente para testes compostos por dezenas de instâncias do GRP.

Em situações normais, o GRP local envia atualizações ao gerenciador global no máximo a cada 2,5 min e os recursos atualizam o seu estado a cada 5 min.

O primeiro teste foi realizado com uma máquina e todos os recursos do portal de voz sendo executados. Este primeiro teste foi realizado com a máquina 4 (Figura 5.1) executando: *ASRServer, TTSServer, VoiceBrowser e VoCManager* com suporte a *Dialogic D/600JCT-2E1* [Int02b].

O segundo teste foi utilizado um agrupamento e variou-se o tempo de atualização do registro pelos gerenciadores locais, e também o número de gerenciadores conectados ao gerenciador global do agrupamento.

5.2.1 Configuração dos Gerenciadores

A configuração dos gerenciadores locais e dos recursos é enviada pelo gerenciador global a cada máquina que participa do agrupamento. Entretanto uma configuração mínima é requerida tanto para os gerenciadores locais e para os recursos. Esta configuração mínima abrange endereço IP do gerenciador global e porta de conexão, no caso do gerenciador local. Para os recursos, a configuração abrange a porta a qual o gerenciador local atende as conexões.

A configuração completa de todo o agrupamento é armazenada em um arquivo de configuração do Gerenciador Global. Esta configuração é distribuída para as máquinas quando os gerenciadores registram-se e solicitam a configuração. Veja em anexo um exemplo da configuração completa de um gerenciador global.

5.2.2 Carga da configuração dos recursos

Para o primeiro teste, a carga completa da configuração do ASR varia entre 5 s a 20 s e utiliza 100% da capacidade de processamento. O sintetizador utiliza 44% do processamento e toda a configuração é carregada em torno de 20 s. Menos de 1 s é necessário para carregar e configurar a infra-estrutura de telefonia e o navegador do portal de voz, e utilizam menos que 20% da capacidade de processamento da máquina.

A requisição de parâmetros de configuração ao gerenciador necessita de aproximadamente 70 ms para o *ASR*, ainda, são necessários 30 ms para o recebimento de pacotes pelo *TTS* e 11 ms *VoiceBrowser* e pelo *VoCManager*.

No segundo teste, com um agrupamento, a requisição de parâmetros de configuração e registro do gerenciador local ao global é menor que 120 ms, para o recebimento total dos pacotes de toda a configuração para os recursos locais da máquina.

5.2.3 Solicitação de Recurso

A requisição por um servidor (recurso) do *VoCManager* ao gerenciador local é atendido em cerca de 10 ms.

A requisição por um servidor (recurso) do gerenciador local para o gerenciador global necessita de aproximadamente 2 s para o retorno de uma resposta de recurso disponível ou não.

Caso o gerenciador local não possua o recurso ou a máquina não possua as condições de processamento livre determinada na configuração ou pelo programador, neste caso, o gerenciador local encaminhará a requisição de recurso para o gerenciador global do agrupamento. O tempo de aquisição do recurso é de aproximadamente 2 s e 10 ms, e 1 s para o *VocManager* conectar-se com o recurso solicitado.

5.2.4 Carga de processamento no Gerenciador Global

O gráfico da Figura 5.2 apresenta o percentual de utilização do processador pelo *GRP Global* quando o número de *GRP local* cresce de 1 a 18 gerenciadores, com período de atualização variando de 200 ms, 2 s, 10 s, 20 s e 60 s. Como observado no gráfico, a carga do processador não ultrapassa os 0,62% e a sua variação depende exclusivamente do número de gerenciadores conectados e o tempo de atualização desses gerenciadores. Para um melhor entendimento do teste, apresenta-se na Figura 5.3 o caso de uso para

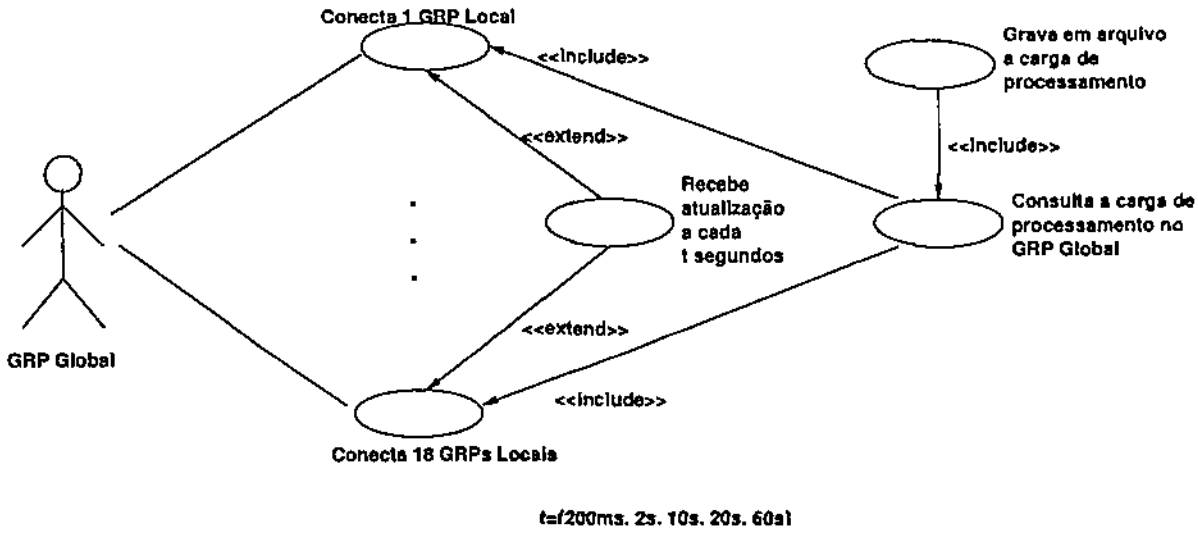


Figura 5.3: Caso de uso para o teste de carga de processamento no GRP Global

o teste de carga de processamento no gerenciador global.

O gráfico apresentado na Figura 5.4 apresenta a relação da carga de processamento com o tempo de atualização. Pode ser notado que a variação estabelece em um patamar razoável e independe do número de máquinas conectadas ao global.

Conclui-se que a infra-estrutura pode acomodar estes dezoito gerenciadores, e portanto não há comprometimento da escalabilidade do sistema, e que a máquina que hospeda o gerenciador poderia ser utilizada também para executar outras funcionalidades. Observe que neste cenário o processador estava sendo utilizado exclusivamente para este teste, e na sua taxa de consumo não se constatou variações significativas na utilização do processador e da memória dos clientes.

5.2.5 Utilização da memória no Gerenciador Global

No gráfico da Figura 5.6 observou-se o aumento da utilização da memória pelo gerenciador, aumentando-se o número de gerenciadores conectados. Isto porque, para cada novo gerenciador conectado, é criado uma referência para ele no quadro negro, veja Seção 3.1.2. A variação da utilização de memória dependendo do tempo de atualização do registro é considerada desprezível. Para um melhor entendimento do teste, apresenta-se na Figura 5.5 o caso de uso para o teste de carga de utilização da memória no gerenciador global.

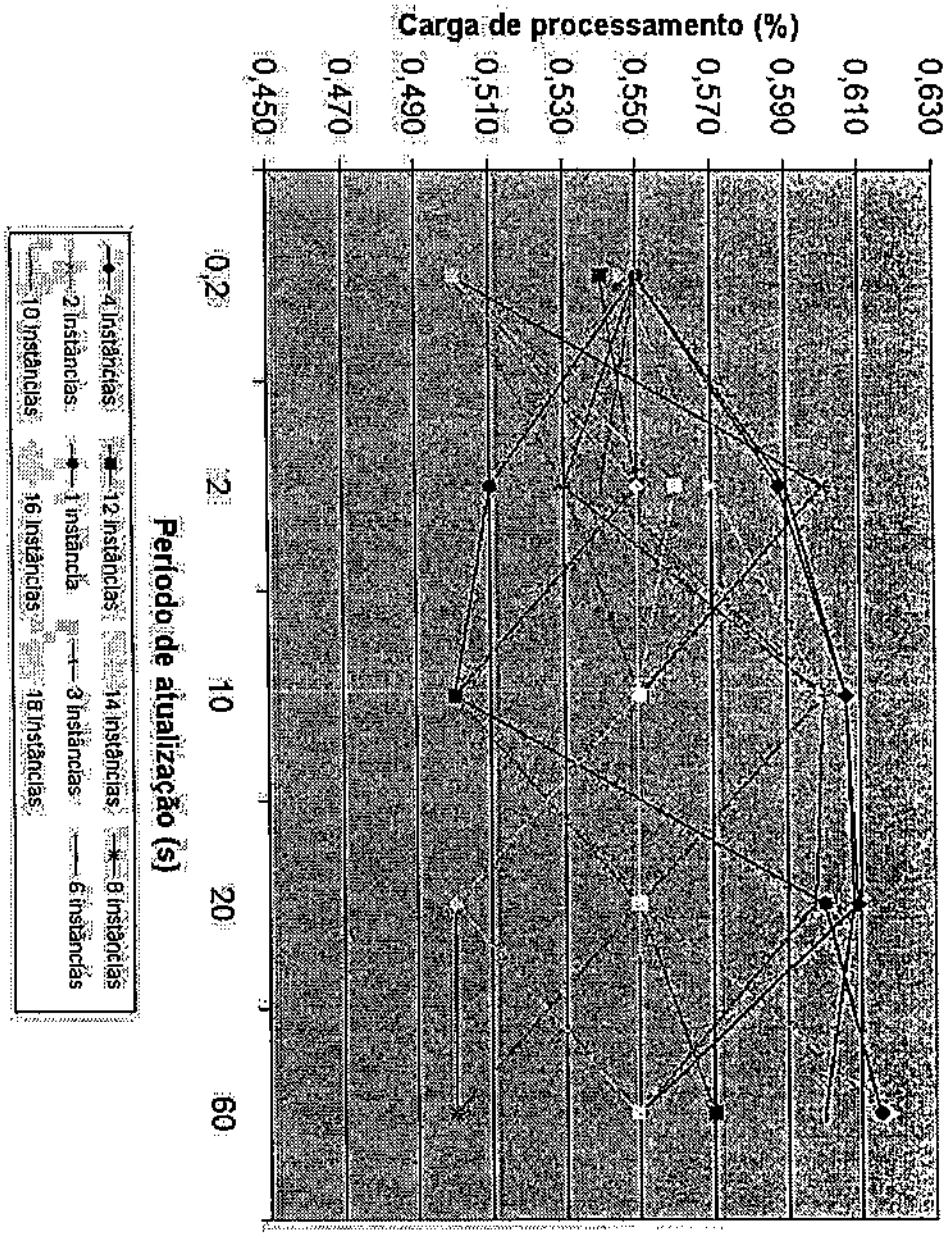


Figura 5.4: Carga de processamento requerida pelo gerenciador dependendo do tempo de atualização

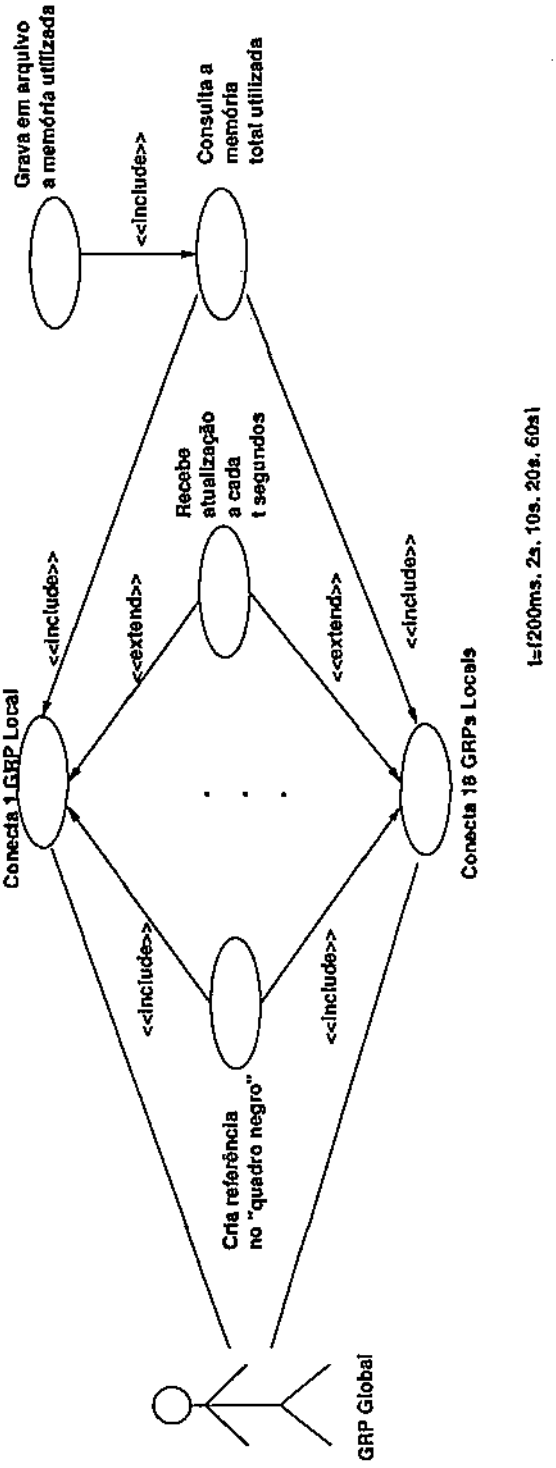


Figura 5.5: Caso de uso para o teste de utilização da memória no GRP Global

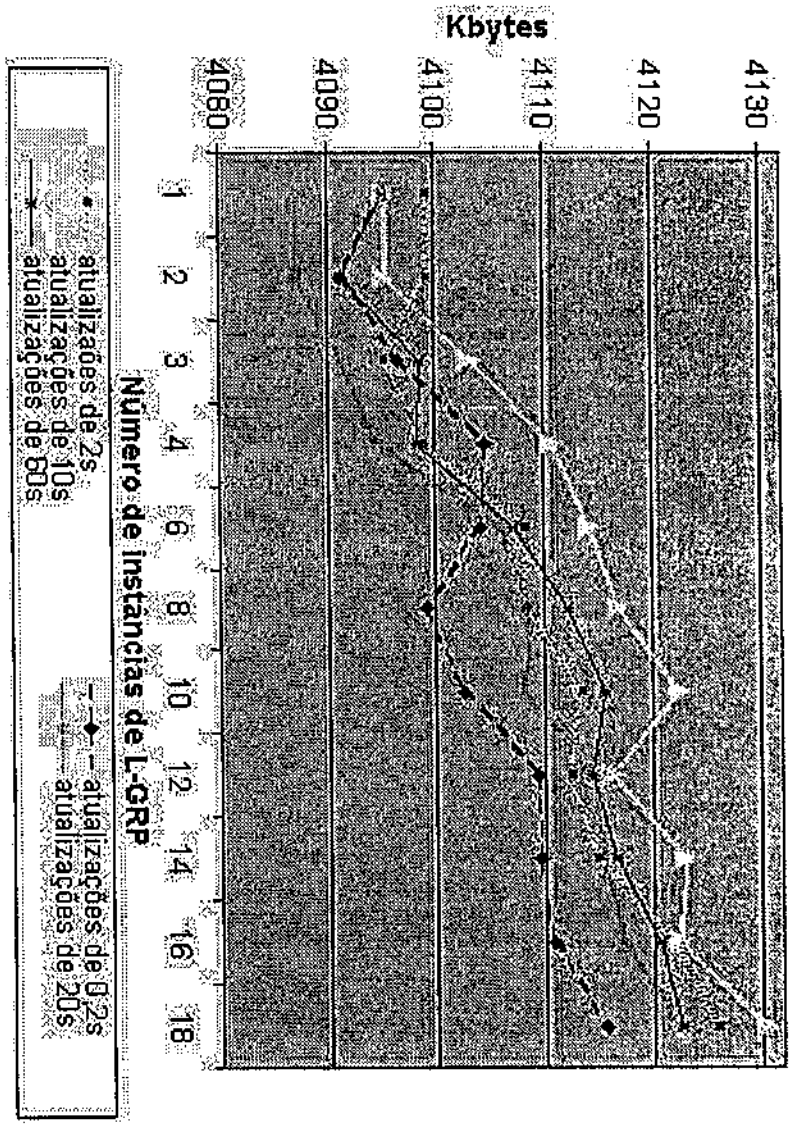


Figura 5.6: Memória requerida pelo gerenciador dependendo do número de gerenciadores no agrupamento

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Nesta dissertação apresentou-se como recursos, de modo a disponibilizar uma infra-estrutura para aplicações distribuídas em uma rede local, podem ser controlados em uma estrutura de agrupamentos. Introduziu-se o desenvolvimento de um gerenciador de recursos e processos, denominado GRP. Este foi desenvolvido para ser aplicado em um contexto de um portal de voz executando em um sistema operacional *Windows NT*, implementado no ambiente *Visual C++ 6.0 da Microsoft*.

O GRP aloca os recursos necessários ao portal de voz para um conjunto de máquinas, onde os processos correspondentes são executados. A arquitetura do gerenciador permite o escalonamento dos aplicativos em diferentes máquinas em uma rede local de computadores. Isto ocorre de forma transparente ao usuário que efetua uma ligação para o serviço do portal de voz, onde os recursos necessários (*ASR, TTS, Voice Browser* e a estrutura de telefônia) podem ser apresentados em máquinas e até mesmo em redes diferentes ao do atendimento da chamada. Esta estrutura necessita uma configuração geral, em que dividimos o agrupamento em um conjunto de máquinas, onde cada máquina abrigará os recursos e processos que irão fornecer o serviço necessário ao funcionamento do portal de voz.

Uma importante contribuição deste trabalho foi na definição e implementação de mecanismos para a automação da configuração e administração de recursos e processos dispersos em uma rede local em sistemas distribuídos, numa arquitetura orientada a objeto. A realização dos resultados experimentais demonstrou que o serviço implementado do portal de voz, impõe uma sobrecarga pequena na máquina a medida que

o número de gerenciadores crescem no agrupamento.

6.2 Trabalhos Futuros

O próximo passo seria disponibilizar a infra-estrutura do portal de voz em sub-redes que poderiam estar localizados em diferentes zonas regionais, desta forma poderiam ter uma única infra-estrutura que atenderia milhares de clientes ao mesmo tempo. Com os recursos de reconhecimento e síntese de voz espalhada por múltiplas redes locais localizadas em diferentes zonas geográficas.

Apesar de contemplarmos tolerância a falhas na infra-estrutura do agrupamento, espera-se nas próximas versões inserirmos um segundo gerenciador de reserva que permutaria com o gerenciador principal, caso este tenha problema.

Outro ponto importante nas próximas versões é utilizarmos o padrão CORBA no GRP para interagirmos com outros gerenciadores que poderiam ser executados em diversos sistemas operacionais e plataformas de hardware, incluindo PC/Linux, PC/Windows e PC/Solaris.

Bibliografia

[Ani01a] Hey Anita. VoiceXML Grammar Reference, 2001. FreeSpeech Developer Network.

[Ani01b] Hey Anita. VoiceXML Reference, April 2001. FreeSpeech Developer Network.

[BRJ98] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide.* Addison-Wesley, 1998.

[Cir02] Walfredo Cirne. Grids computacionais: Arquiteturas, tecnologias e aplicações. Technical report, Universidade Federal de Campina Grande, dezembro 2002.

[CKKG99] Steve J. Chapin, Dimitrios Katramatos, John Karpovich, and Andrew Grimshaw. Resource management in Legion. *Future Generation Computer Systems*, 15(5–6):583–594, 1999.

[Con01] The International Engineering Consortium. Voice portal solutions: Where can you go first? Technical report, WEB ProForum Tutorials - International Engineering Consortium, 2001.

[FFO00] Jean-Marie Farines, Jonidas S. Fraga, and Romulo S. De Oliveira. *Sistemas de Tempo Real.* IME-USP (Instituto de Matemática e Estatistica da Universidade de São Paulo), 2000.

[For00] VoiceXML Forum. Voice Extensible Markup Language, March 2000.

[Gar99] Felix C. Gartner. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Computing Surveys*, 31(1):1–26, 1999.

[Gup99] M. Gupta. Reservation based distributed resource management. Master's thesis, 1999.

[Int02a] Intel Corporation, Intel Corporation - Network Processing Group 1515 Route 10 Parsippany NJ 07054 U.S.A. *System Release 5.1.1 for Windows - Release Guide*, 2002.

[Int02b] Intel®. *Intel® Dialogic® DualSpan-JCT Series.* Intel®, 1515 Route Ten Parsippany, NJ 07054 Phone: 1-973-993-3000 Fax: 1-973-993-3093, 2002.

[Int03] Intel Corporation, Intel Corporation - Network Processing Group 1515 Route 10 Parsippany NJ 07054 U.S.A. *System Release 5.1.1 for Windows - Release Update*, 2003.

[Käh00] Pekka Kähkipuro. Performance modeling framework for corba based distributed systems. Technical Report 13–25, University of Helsinki, Department of Computer Science, P.O. Box 26 (Teollisuuskatu 23), FIN-00014 Finland, August 2000.

[KWP01] Kwansik Kim, Craig M. Wittenbrink, and Alex Pang. Extended specifications and test data sets for data level comparisons of direct volume rendering algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 7(4):299–317, 2001.

[KYH+01] Fabio Kon, Tomonori Yamane, Christopher Hess, Roy Campbell, and M. Dennis Mickunas. Dynamic Resource Management and Automatic Configuration of Distributed Component Systems. In *Proceedings of the 6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'2001)*, pages 15–30, San Antonio, Texas, February 2001.

[MK02] Jeferson R. Marques and Fabio Kon. Gerenciamento de Recursos Distribuídos em Sistemas de Grande Escala. *20 Simpósio Brasileiro de Redes de Computadores (SBRC2002)*, 2002.

[NhCN97] Klara Nahrstedt, Hao hua Chu, and Srinivas Narayan. QoS-Aware Resource Management for Distributed Multimedia Applications. Technical Report UIUCDCS-R-97-2030, 1997.

[NR93] B. Clifford Neuman and Santosh Rao. Resource management for distributed parallel systems. *Proceedings of the 2nd Internationational Symposium on High Performance Distributed Computing*, July 1993.

[NR94] B. Clifford Neuman and Santosh Rao. The Prospero Resource Manager: A Scalable Framework for Processor Allocation in Distributed Systems. *Appears in Concurrency: Practice and Experience*, 6(4):339–355, June 1994.

[NS95] Klara Nahrstedt and Jonathan M. Smith. The QoS Broker. *IEE Multimedia*, 2(1):53–67, Spring 1995.

[PdAdA03] Angelo Perkusich, Hyggo Oliveira de Almeida, and Denis Hipólito de Araujo. A Software Framework for Real-Time Embedded Automation and Control Systems. In *9th IEEE International Conference on Emerging Technologies and Factory Automation, Lisboa*, volume 2, pages 181–184, Piscatway, NJ, September 2003. IEEE.

[Sch90] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

[SG96] Mary Shaw and David Garlan. *Software architecture - Perspectives on emerging discipline.* Prentice-Hall, 1996.

[SGG02] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating Systems Concepts.* John Wiley & Sons, INC, sixth edition edition, 2002.

[Sie99] Otto Sievert. A Gentle Introduction to Globus. *Parallel Computation*, Spring 1999.

[Tan92] Andrew S. Tanenbaum. *Modern Operating Systems.* Prentice Hall, Englewood Cliffs, N.J., 1992.

[TP00] Wilfredo Torres-Pomales. Software Fault Tolerance: A Tutorial. Technical report, NASA Center of AeroSpace, Langley Research Center, Hampton, Virginia, October 2000.

[WY01] Kihun Kim Wanghong Yuan, Klara Nahrstedt. R-EDF: A Reservation-Based EDF Scheduling Algorithm for Multiple Multimedia Task Classes. In *7th IEEE Real-Time Technology and Applications Symposium*, Taipei, Taiwan, May 2001.

Anexos

Arquivo de Configuração para o Gerenciador Global

```
#=========================================================
#Configuração para o Gerenciador de Recursos e Processos
#---------------------------------------------------------
# Configuração do Global GRP
[master]

server.name RPM_MASTER
server.port 8000

log.file.name c:/log file
log.file.enable         1
log.screen.enable 1
log.debug.level 2
log.db.enable 0

#=========================================================
# Localhost
[computer=localhost]
#---------------------------------------------------------
# Configuração para o Local GRP -
#cada computador tem uma configuração
[application=RPM]

server.port 7000
```

```
rpm.connect 1
rpm.ip localhost
rpm.port 8000

delta.time.stamp 500

load.processor 80

log.file.name c:/log file
log.file.enable         1
log.screen.enable 0
log.debug.level 2
log.db.enable 0

#-----------------------------------------------------
# Config for ASRServer
[application=ASRServer]

server.file ASRServer.exe
server.config config_asr.txt
server.port 5001

#=====================================================
#Parâmetros de Log
log.file.name c:/log file
log.file.enable         1
log.screen.enable 0
log.debug.level 4
log.db.enable 0
log.db.type             MSSQL
log.db.host ip address
log.db.login login
log.db.passwd password
```

```
log.db.table ocurrence
#=========================================================
#ASR Engine Selection
asr.type asr type
number.ports            2

#Outros parâmetros ...

#=========================================================
# 10.0.0.1
[computer=10.0.0.1]
#---------------------------------------------------------
# Configuração para o Local GRP - cada computador
#tem uma configuração
[application=RPM]

server.port 7000

delta.time.stamp 300

load.processor 80

log.file.name c:/log file
log.file.enable         1
log.screen.enable 1
log.debug.level 4
log.db.enable 0

#---------------------------------------------------------
# Configuração para o Gerenciador de Canal
[application=VoCManager]

server.file VoCManager.exe
server.config config_voc.txt
```

```
server.port 10000

#=============================================================
#Parâmetros de Log
log.channel.path c:/log path
log.audio.enable 1
log.basename chan
log.debug.level 4
log.screen.enable 0
log.file.enable 1
log.db.enable 0
log.db.type              MYSQL
log.db.host localhost
log.db.login login
log.db.passwd password
log.db.table log_vb

#=============================================================
# Parâmetros para conexão com servidores -Estes parâmetros
#são usados quando o rpm.connect é assinalado para "0"
asrserver.ip localhost
asrserver.port 4000
ttsserver.ip localhost
ttsserver.port 5000
voicebrowser.ip localhost
voicebrowser.port 6000
#=============================================================
#Configuração dos parâmetros para o canal de aúdio
channel.type   DIALOGICCARD
channel.dialogic.type  E1
channel.number   1
enable.bargein   1
channel.ports   34
vad.mean   200.0
```

```
vad.std    60.0

#Outros parâmetros ...

#=============================================================
# Configuração para o Local GRP - 10.0.0.2
[computer=10.0.0.2]
#-------------------------------------------------------------
# Configuração para o Local GRP - cada computador tem uma
#configuração
[application=RPM]

server.port 7000

#Time(s)
delta.time.stamp 300

#Porcent(%)
load.processor 80

log.file.name c:/log file
log.file.enable         1
log.screen.enable 1
log.debug.level 4
log.db.enable 0

#-------------------------------------------------------------
# Config for Voice Browser
[application=VoiceBrowser]

server.file VoiceBrowser.exe
server.config config_vob.txt
server.port 6000
```

```
#-----------------------------------------------------
# Parâmetros para o servidor de soquete do portal de voz

vxi.vob.operationMode 1
vxi.vob.socketServerAddress localhost
vxi.vob.socketServerPort 6000

#=====================================================
#Parâmetros de Log

log.file.name c:/log file
log.file.enable          1
log.screen.enable 1
log.debug.level 4
log.db.enable 0
log.db.enable 0
log.db.type MYSQL
log.db.host localhost
log.db.login login
log.db.passwd password
log.db.table log_vb

#Outros parâmetros ...

#-----------------------------------------------------
# Config for TTSServer
[application=TTSServer]

server.file TTSServer.exe
server.config config_tts.txt
server.port 5000

#=====================================================
#Parâmetros de Log
```

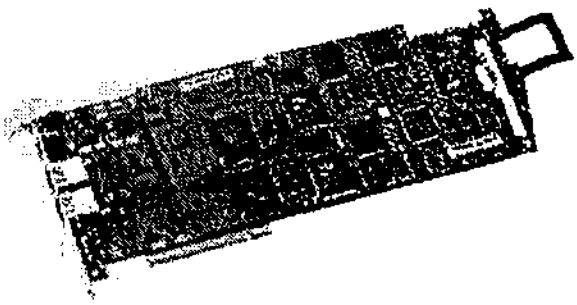
```
log.file.name c:/log file
log.file.enable 1
log.screen.enable 0
log.debug.level 4
log.db.enable 0
log.db.type MSSQL
log.db.host ip address
log.db.login login
log.db.passwd password
log.db.table ocurrence


#Outros parâmetros ...
```

Especificação Técnica da Placa de Telefonia

Intel in Communications

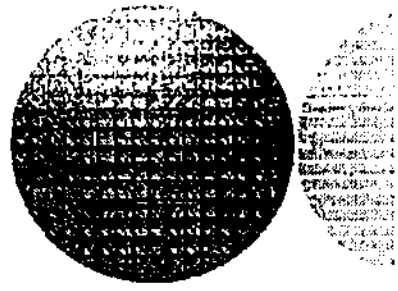
Intel® Dialogic® DualSpan-JCT Series

Intel® Dialogic® DualSpan-JCT boards provide two spans of digital network interfaces in a H.100-compliant universal PCI form factor. The boards contain rich media features such as voice processing, continuous speech processing (CSP), fax, tone signaling, global tone detection, global tone generation, and call progress analysis, ideal for service providers and large enterprises.



Features and Benefits

- High channel-per-slot density: two T-1 ISDN PRI trunks with 48 channels of voice processing or two E-1 ISDN PRI trunks with 60 channels of voice processing
- Supports continuous speech processing[1], a flexible speech processing technology, coupled with efficient drivers, off-loads critical real-time signal processing in speech-enabled applications to onboard DSPs. Reduces system latency, increases recognition accuracy, and improves overall system response time for high-density speech solutions.
- Supports DSP-based onboard fax and host-based speech recognition to maximize the number of boards in the system[2]
- Supports G.726 bit exact and GSM coders, letting developers implement unified messaging applications that meet VPIM standards
- Offered in industry-standard 32-bit PCI form factor with universal connector
- Silence-compressed recording eliminates silence and preserves hard disk space
- H.100 connector lets developers take advantage of the industry-standard CT Bus and increases the board's capacity to interoperate with other CT Bus-compatible boards
- Downloadable signal and call processing firmware provides easy feature enhancement and field-proven performance based on over four million installed ports
- Unified call control access through Global Call interface provides worldwide application portability and shortens development time by using the same API for almost any network protocol
- Intel® Dialogic® CT Media™ server software support facilitates multiapplication development
- Supports the Board Watch tool, the SNMP-compatible software for remote CT board management
- Enables system integrators and developers to lower costs by incorporating more ports per chassis, using less expensive desktop-style machines, and easing configuration/installation effort
- Software development kits (SDKs) for Windows NT®, Windows® 2000, and Linux® yield faster time to market



[1] On D/480JCT-2T1 robbed-bit systems only. Please refer to the continuous speech processing datasheet for more information.
[2] Fax and host-based speech recognition are mutually exclusive.

The Intel® Dialogic® D/480JCT-2T1 and D/600JCT-2E1 boards are the next generation of DualSpan products based on SpringWare firmware. They are ideal for developers seeking to provide cost-effective, highly scalable, high-density communications applications requiring multimedia resources such as voice, software-based speech recognition, fax, and digital network interface in a single personal computer (PC) slot. These boards offer a rich set of advanced features and support digital signal processing (DSP) technology and industry-standard PCI bus and CT Bus technologies.

Support for the innovative continuous speech processing technology enables seamless integration of software-based speech recognition software from leading speech technology vendors. Onboard DSP-based fax and support for software-based speech recognition lets developers maximize the number of boards in the system for multimedia communications applications such as Web-enabled call centers, voice portals, unified messaging, or speech-enabled interactive voice response (IVR). The option to use new voice coders such as GSM and G.726 (the de facto standards when complying with Voice Profile for Internet Messaging (VPIM) standards) provides the capability to build unified messaging solutions while working with existing legacy messaging systems. In addition, support under Global Call and Intel® Dialogic® CT Media™ software facilitates global deployment and adds the flexibility to scale systems to meet the growing needs of your business.

Configurations

Use Intel Dialogic DualSpan-JCT boards to develop sophisticated, multimedia communications systems incorporating capabilities such as voice processing, facsimile, text-to-speech (TTS), and automatic speech recognition (ASR). These boards share a common hardware and software architecture with other SCbus and CT Bus boards for maximum flexibility and scalability. You can add features and grow the system while protecting your investment in hardware and application code. Applications can be ported easily to lower or higher density platforms, with only minimum modifications.

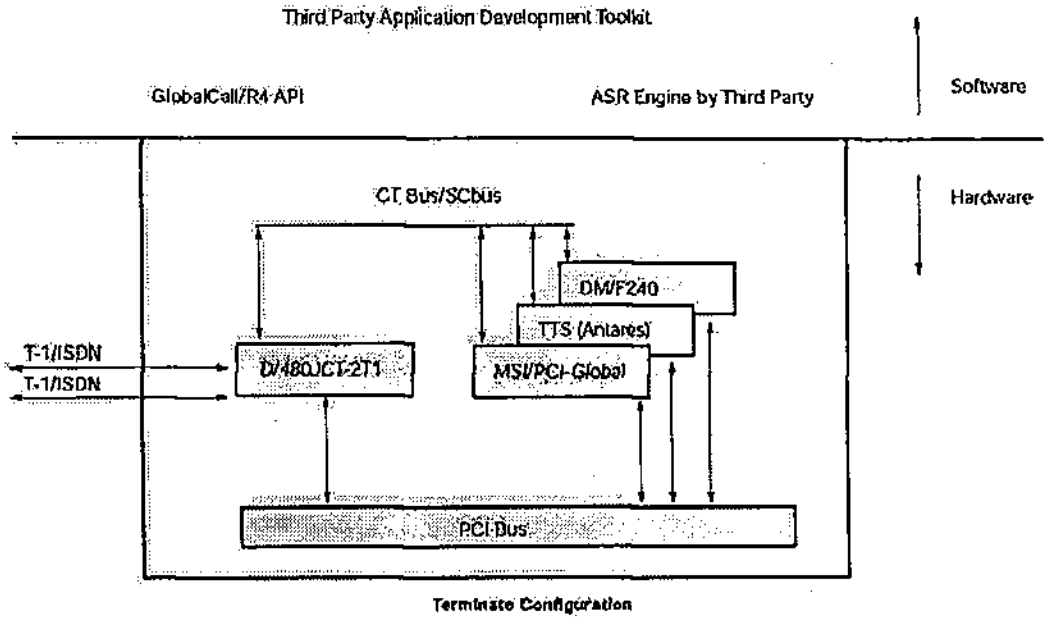
DualSpan-JCT boards install in any PCI-based PC or server (PCI bus or mixed PCI/ISA bus) and compatible computers (Intel386™, Intel486™, or Pentium® processors or Sun UltraSPARC®). Each board occupies a single expansion slot and up to 10 boards can be configured in a system. The number of boards and channels supported varies depending on the application, the operating system, the amount of disk I/O required, the number of CT Bus loads per board, and the host computer's CPU(s), and power supply.

DualSpan-JCT boards can operate in either terminate or drop-and-insert configurations. In a terminate configuration, the board handles the call processing of the digital audio and telephony signaling, facsimile, and the software-based speech recognition. If additional resources are required, such as TTS, these resources can be switched to the call via the CT Bus/SCbus. A DualSpan-JCT board installed as a terminating device eliminates the need for a channel bank. The system operates as a standalone call processing node.

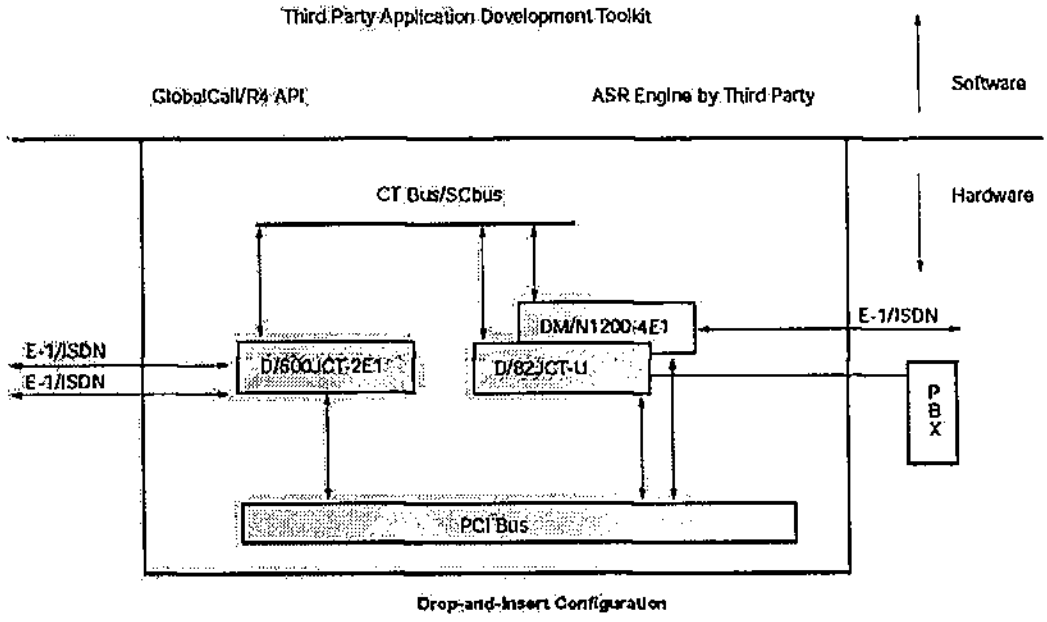
In a drop-and-insert configuration, use DualSpan-JCT boards and a DTI board connected via the CT Bus/SCbus to pass T-1 or E-1 time slots through to each other. This configuration joins two separate T-1 or E-1 lines, or it can be placed in-line between a T-1 or E-1 line and a switch (a PBX, for example). Calls on individual channels can either terminate at a call processing resource on a DualSpan-JCT series board, or "flow through" transparently to the DTI board.

Applications

- Messaging and enhanced services
- Voice portal
- Contact center and e-Business
- PC-PBX
- Switching and call completion
- Prepaid/debit card
- Gateway switch



Terminate Configuration



Drop-and-Insert Configuration

ISDN-PRI Support

The ISDN Primary Rate Interface (PRI) firmware is a standard feature of the DualSpan-JCT Series. The PRI firmware is approved for use with many popular protocols in major market segments, based on both T-1 (1.544 Mb/s) and E-1 (2.048 Mb/s) physical interfaces.

Features and benefits of ISDN PRI include

- ISDN PRI connectivity to computer telephony (CT) systems
- Dialed Number Identification Service (DNIS) lets the application route incoming calls by automatically identifying the number the caller dialed
- Automatic Number Identification (ANI) lets the application identify the calling party
- ANI-on-Demand feature saves money by selectively requesting ANI information only when needed
- ISDN offers inherent benefits to call center applications with its fast call setup and fast retrieval of DNIS and ANI information on inbound calls
- Call-By-Call Service Selection lets an application select the most efficient bearer channel service on a call-by-call basis
- Subaddressing allows direct connection to individual extensions or devices sharing the same phone number, or as a proprietary messaging mechanism
- Powerful and universal software interface simplifies access for developers who are unfamiliar with ISDN, yet enables sophisticated control of features
- *Multinational approvals with many popular protocols*
- User-to-User Information lets an application send proprietary messages to remote systems during call establishment
- Facility, Notify, and optional Information Elements (IEs) let applications work with network-specific supplementary services
- Layer 2 access empowers developers to build customized Layer 3 protocol
- Ability to dynamically set protocol timers through host *application programming interfaces (APIs)*
- Programmable Startup Cause Value presentation to the network lets the user reject an incoming call with a preassigned cause value if the host has not yet done a waitcall on that channel
- Maskable Layer 2 Control lets the application toggle between bringing Layer 2 up and down as desired
- Support for SERVICE, SERVICE_ACK, and STATUS ENQUIRY messages

Software Support

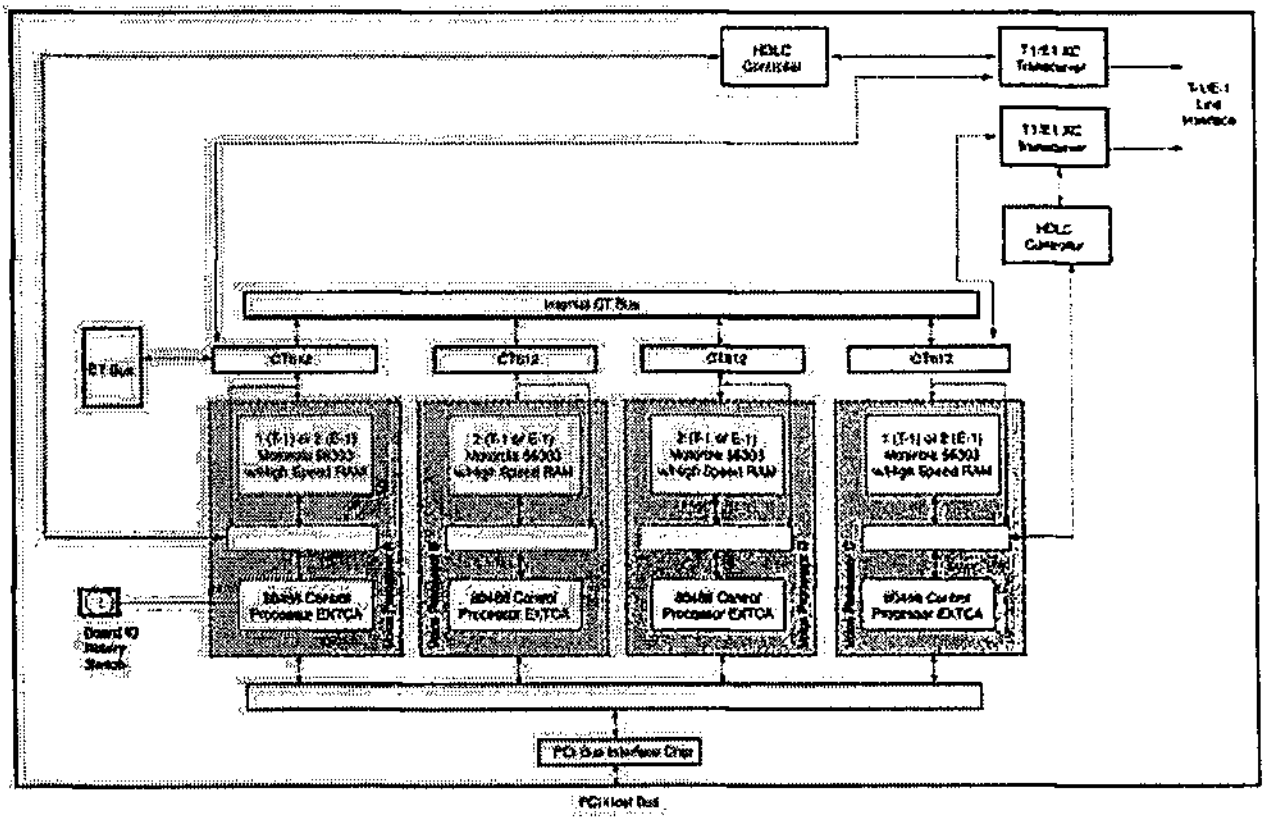
Intel Dialogic DualSpan-JCT boards are supported by the System Software and Software Development Kits (SDKs) for Windows NT®, Windows® 2000, and Linux®. These packages contain a set of tools for developing sophisticated, multimedia communications applications.

DualSpan-JCT boards can use Global Call software, as well as support Intel Dialogic CT Media server software which facilitates multiapplication development. These boards also support the Board Watch tool, the SNMP-compatible software for remote CT board management. Board Watch software simplifies the management of CT devices and lowers the total cost of operation. Centralized management capabilities provide a single point of configuration and inventory for all network devices. Fault management for high-availability systems includes diagnostics, detection, and recovery capabilities.

Global Call

Global Call software provides a common signaling interface for network-enabled applications, regardless of the signaling protocol needed to connect to the local telephone network. Global Call is the recommended API for unified call control for Spring Ware and DM3 architectures. The signaling interface provided by Global Call facilitates the exchange of call control messages between the telephone network and virtually any network-enabled application. Global Call lets developers create an application that can work with signaling systems worldwide, regardless of the network to which they are connected.

Global Call is ideal for high-density, network-enabled solutions for voice, data, and video, where the supported hardware and signaling technology can vary widely. Rather than requiring the application to handle the low-level details, Global Call software offers a consistent, high-level interface to the user, handling each country's unique protocol requirements in a way that is transparent to the application



Functional Description

D/480JCT-2T1

The Intel Dialogic D/480JCT-2T1 board connects directly to a channel service unit (CSU), digital service unit (DSU), or to other network terminating equipment. The CSU chosen must support the D4 or ESF (within ISDN) superframe format. Most functions traditionally performed by a DSU (such as unipolar to bipolar format conversion, framing, etc.) are performed by the D/480JCT-2T1 board. (The only exception is the ability to interpret certain bipolar violation patterns such as loopback start and stop commands from the T-1 network.)

The board processes the digital on-hook/off-hook signaling information and digital voice signals from the telephone network. Digital T-1 signals enter the board via a T1XC line interface (see block diagram). The line interface contains a software-switchable clock that can be set to any of the following settings:

- loop (clocking is slaved to the external network)
- independent (clocking is derived from an onboard oscillator)
- expansion (clocking is slaved to another bus clock master board)

The incoming T-1 bit stream is applied to a CT612 chip, which acts as a traffic coordinator for each channel and as an interface to the CT Bus. This serial bit stream contains the digitized voice data and the signaling information for the incoming call.

Each of four CT612 functional modules on the D/480JCT-2T1 board transmits several lower speed data streams over a single high-speed channel. The bus configuration is set when the firmware is downloaded at system initialization. These chips incorporate matrix switching capabilities. Under control of an onboard control processor, a CT612 functional module can connect a call being processed or an available external resource to any of the CT Bus time slots. This lets the application route calls to any added resources such as fax, TTS, or ASR.

A DSP resource receives digital voice data via a CT612 module. The DSP processes the data based on Spring Ware firmware loaded in its high-speed RAM. Each

DSP performs the following signal analysis and operations on this incoming data:

- applies automatic gain control (AGC) to compensate for variations in the level of the incoming audio signal
- applies an Adaptive Differential Pulse Code Modulation (ADPCM), Pulse Code Modulation (PCM), GSM, or G.726 algorithm to compress the digitized voice and save disk storage space
- detects the presence of tones — DTMF, MF, or an application-defined, single- or dual-frequency tone
- detects silence to determine whether the line is quiet and the caller is not responding

For outbound data, the DSP performs the following operations:

- expands stored, compressed audio data for playback
- adjusts the volume and rate of speed of playback upon application or user request
- generates tones — DTMF, MF, or any application-defined, general-purpose tone

The dual processor combination associated with each line interface also performs the following outbound dialing and call progress monitoring functions:

- transmits an off-hook signal to the telephone network
- dials out (makes an outbound call)
- monitors and reports call progress results
  - line busy or congested
  - operator intercept
  - ring, no answer
  - or if the call is answered, whether answered by a person, an answering machine, a facsimile machine or modem

The board's line interface extracts or inserts telephony signaling information, which is processed by an onboard control processor. The DSPs only process the digitized voice data.

When recording speech, the DSP can use digitizing rates from 13 Kb/s to 64 Kb/s as selected by the application for the best speech quality and most efficient storage. The digitizing rate is selected on a channel-by-channel basis and can be changed each time a record or play function is initiated. The DSP-processed speech is transmitted by the control processor to the host PC for disk storage. When replaying a stored file, the processor retrieves the voice information from the host PC and passes it to the DSP, which converts the file into digitized voice. The DSP uses the CT Bus circuitry to send the digitized voice responses to the caller via the T1XC line interface.

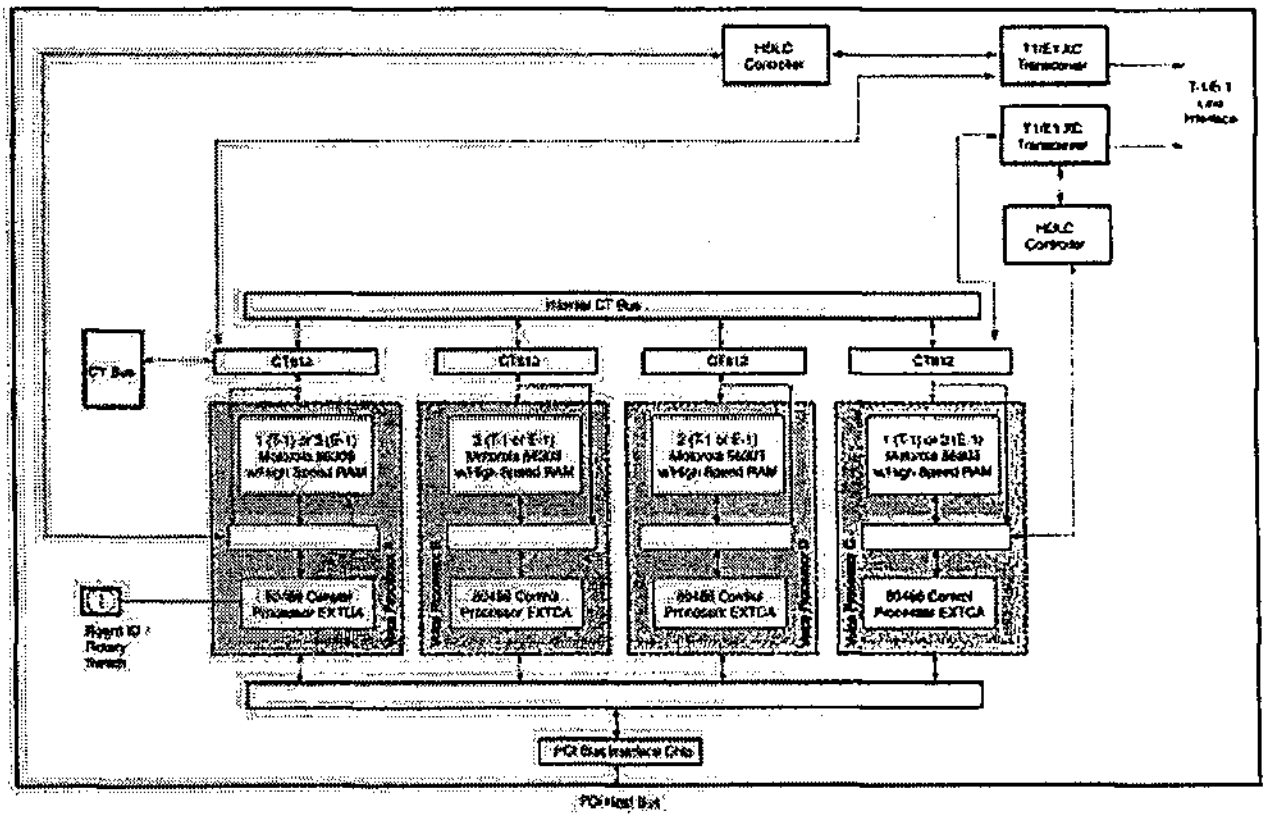
For CT Bus/SCbus configurations, the internal local buses operate at 2.048 Mb/s. A high-level data link controller (HDLC) formats ISDN data. The HDLC receives ISDN signaling data from the T1XC interface and CT612 ASIC and makes it available to the control processor. It also formats and sends outbound signaling data from the control processor to the network interface through the CT612 ASIC and T1XC transceiver chip.

The onboard control processor(s) controls all operations of the board via local buses and interprets and executes commands from the host PC. These processors

- handle real-time events
- manage data flow to the host PC to provide faster system response time
- reduce PC host processing demands
- process DTMF and telephony signaling before passing them to the application
- free the DSPs to perform signal processing

Communications between a processor and the host PC is via the shared RAM, which acts as an input/output buffer, increasing the efficiency of disk file transfers. This RAM interfaces to the host PC via the PCI bus. All operations are interrupt-driven to meet the demands of real-time systems. When the system is initialized, Spring Ware firmware is downloaded from the host PC to the onboard code/data RAM and DSP RAM to control all board operations. This firmware gives the board all of its intelligence and enables easy feature enhancement and upgrades.

The traffic controller ASIC is the Intel486 control processor interface that handles all peripheral devices (CT612, HDLC, DSPs, T1XC) and host PC functions (board locator technology, programmable interrupts, and shared RAM). The board locator technology circuit inside the traffic controller ASIC operates in conjunction with a rotary switch, eliminating the need to set confusing jumpers or DIP switches.



D/600JCT-2E1

The Intel Dialogic D/600JCT-2E1 board processes the digital on-hook/off-hook signaling information and digital voice signals from the telephone network. Digital E-1 signals enter the board via an E1XC line interface (see block diagram). The line interface supports CRC4 error detection (Cyclic Redundancy Check) and contains a software-switchable clock that can be set to any of the following settings:

- loop (clocking is slaved to the external network)
- independent (clocking is derived from an onboard oscillator)
- expansion (clocking is slaved to another bus clock master board)

Each of four CT612 functional modules on the D/600JCT-2E1 board transmits several lower speed data streams over a single high-speed channel. The bus configuration is set when the firmware is downloaded at system initialization. These chips incorporate matrix switching capabilities. Under control of an onboard control processor, a CT612 functional module can connect a call being processed or an available external resource to any of the CT Bus/SCbus time slots. This lets the application route calls to any added resources such as fax, TTS, or ASR.

A DSP resource receives digital voice data via a CT612 module. The DSP processes the data based on Spring Ware firmware loaded in its high-speed RAM. Each DSP performs the following signal analysis and operations on this incoming data:

- applies AGC to compensate for variations in the level of the incoming audio signal
- applies an ADPCM, PCM, GSM, or G.726 algorithm to compress the digitized voice and save disk storage space
- detects the presence of tones — DTMF, R2MF, or an application-defined, single- or dual-frequency tone
- detects silence to determine whether the line is quiet and the caller is not responding

For outbound data, the DSP performs the following operations:

- expands stored, compressed audio data for playback
- adjusts the volume and rate of speed of playback upon application or user request
- generates tones — DTMF, R2MF, or any application-defined, general-purpose tone

The dual processor combination also performs the following outbound dialing and call progress monitoring functions:

- transmits an off-hook signal to the telephone network
- dials out (makes an outbound call)
- monitors and reports call progress results
  - line busy or congested
  - operator intercept
  - ring, no answer
  - or if the call is answered, whether answered by a person, an answering machine, a facsimile machine or modem

The board's line interface extracts or inserts telephony signaling information, which is processed by an onboard control processor. The DSPs only process the digitized voice data.

When recording speech, the DSP can use digitizing rates from 13 Kb/s to 64 Kb/s as selected by the application for the best speech quality and most efficient storage. The digitizing rate is selected on a channel-by-channel basis and can be changed each time a record or play function is initiated. The DSP-processed speech is transmitted by the control processor to the host PC for disk storage. When replaying a stored file, the processor retrieves the voice information from the host PC and passes it to the DSP, which converts the file into digitized voice. The DSP uses the CT Bus circuitry to send the digitized voice responses to the caller via the E1XC line interface.

For CT Bus/SCbus configurations, the internal local buses operate at 2.048 Mb/s. An HDLC formats ISDN data. The HDLC receives ISDN signaling data from the E1XC interface and CT612 ASIC and makes it available to the control processor. It also formats and sends outbound signaling data from the control processor to the network interface through the CT612 ASIC and E1XC transceiver chip.

The onboard control processor(s) controls all operations of the board via local buses and interprets and executes commands from the host PC. These processors

- handle real-time events
- manage data flow to the host PC to provide faster system response time
- reduce PC host processing demands
- process DTMF and telephony signaling before passing them to the application
- free the DSPs to perform signal processing

Communications between a processor and the host PC is via the shared RAM, which acts as an input/output buffer, increasing the efficiency of disk file transfers. This RAM interfaces to the host PC via the PCI bus. All operations are interrupt-driven to meet the demands of real-time systems. When the system is initialized, Spring Ware firmware is downloaded from the host PC to the onboard code/data RAM and DSP RAM to control all board operations. This firmware gives the board all of its intelligence and enables easy feature enhancement and upgrades.

The traffic controller ASIC is the Intel486 control processor interface that handles all peripheral devices (CT612, HDLC, DSPs, E1XC) and host PC functions (board locator technology, programmable interrupts, and shared RAM). The board locator technology circuit inside the traffic controller ASIC operates in conjunction with a rotary switch, eliminating the need to set confusing jumpers or DIP switches

Technical Specifications** for D/480JCT-2T1

	Number of ports	48
	Max. boards/system	10. Number may be limited by application, system performance, and the number of CT Bus loads per board.
	CT Bus loads per board	2.5
	Maximum CT Bus loads per system	20 (See CT Bus specification for further details.)
	Digital network interface	Onboard DSX-1 interface
	Resource sharing bus	H.100 CT Bus
	Control processors	Four Intel486™ GX processors @ 32.7 MHz, 0 wait state
	Digital signal processors	Six Motorola* DSP56303 @ 100 MHz, each with 256 K word private, 2 wait state SRAM
Host Interface		
	Bus compatibility	PCI. Complies with PCISIG* Bus Specification, Rev. 2.2
	Bus speed	33 MHz maximum
	Bus mode	32- to 16-bit conversion in target mode
	Shared memory	4 x 64 KB page
	I/O ports	None
	Support	3.3 V or 5 V signaling environment (universal connectivity)
Telephone Interface		
	Clock rate	1.544 Mb/s ±32 ppm
	Level	3.0 V (nominal)
	Pulse width	323.85 ns (nominal)
	Line impedance	100 Ohm ±10%
	Other electrical characteristics	Complies with AT&T* TR62411 and ANSI T1.403-1989
	Framing	SF (D3/D4) ESF for ISDN
	Line coding	AMI AMI with B7 stuffing B8ZS
	Clock and data recovery	Complies with AT&T TR62411 and Bellcore* TA-TSY-000170
	Jitter tolerance	Complies with AT&T TR62411 and ANSI T1.403-1989
	Connectors	RJ-48C
	Telephony bus connector	H.100-style 68-pin fine pitch card edge connector
	Loopback	Supports switch-selectable local analog loopback and software selectable local digital loopback
Power Requirements		
	+5 VDC	3.36 A typical; 4.03 A maximum
	+12 VDC	7.3 mA typical; 8.0 mA maximum
	-12 VDC	Not required
	Operating temperature	0°C to +50°C
	Storage temperature	-20°C to +70°C
	Humidity	8% to 80% noncondensing
	Form factor	PCI long card 12.3 in. long (30.75 cm) (without edge retainer) or 13.3 in. long (33.25 cm) (with edge retainer) 0.79 in. wide (1.975 cm) (total envelope) 3.87 in. high (9.675 cm) (excluding edge connector)

Technical Specifications** for D/480JCT-2T1 (cont.)

Safety and EMI Certification		
	United States	FCC part 68 ID#: EBZUSA-20078-XD-N UL: 1950 (E96804)
	Canada	IC: 865 5959 A UL: CSA 950 (E96804)
	Estimated MTBF	162,000 hours per Bellcore Method I
	Warranty	Intel® Telecom Products Warranty Information at http://www.intel.com/network/csp/products/3144web.htm

Technical Specifications** for D/600JCT-2E1

	Number of ports	60
	Max. boards/system	10. Number may be limited by application, system performance, and the number of CT Bus loads per board.
	CT Bus loads per board	2.5
	Maximum CT Bus loads per system	20 (See CT Bus specification for further details.)
	Digital network interface	Onboard E-1 interface
	Resource sharing bus	H.100 CT Bus
	Control processors	Four Intel486™ GX processors @ 32.7 MHz, 0 wait state
	Digital signal processors	Six Motorola® DSP56303 @ 100 MHz, each with 256 K word private, 2 wait state SRAM
Host Interface		
	Bus compatibility	PCI. Complies with PCISIG® Bus Specification, Rev. 2.2.
	Bus speed	33 MHz maximum
	Bus mode	32- to 16-bit conversion in target mode
	Shared memory	4 x 64 KB page
	I/O ports	None
	Support	3.3 V or 5 V signaling environment (universal connectivity)
Telephone Interface		
	Network clock rate	2.048 Mb/s ±50 ppm
	Internal clock rate	2.048 Mb/s ±32 ppm
	Level	2.37 V (nominal) for 75 Ohm lines 3.0 V (nominal) for 120 Ohm lines
	Pulse width	244 ns (nominal)
	Line impedance	75 Ohm, unbalanced 120 Ohm, balanced
	Other electrical characteristics	Complies with CCITT Rec. G. 703
	Framing	CCITT G. 704-1988 with CRC4
	Line coding	HDB3
	Clock and data recovery	Complies with CCITT Rec. G.823-1988
	Jitter tolerance	Complies with CCITT Rec. G.823, G.737, G.738, G.742-1988
	Connectors	BNC for 75 Ohm lines RJ-48C for 120 Ohm lines
	Telephony bus connector	H.100-style 68-pin fine pitch card edge connector
	Loopback	Supports switch-selectable local analog loopback and software selectable local digital loopback

Technical Specifications** for D/600JCT-2E1 (cont.)

Power Requirements

+5 VDC	3.7 A typical; 4.3 A maximum
+12 VDC	7.4 mA typical; 8.8 mA maximum
–12 VDC	Not required
Operating temperature	0°C to +50°C
Storage temperature	–20°C to +70°C
Humidity	0% to 80% noncondensing
Form factor	PCI long card 12.3 in. long (30.75 cm) (without edge retainer) or 13.3 in. long (33.25 cm) (with edge retainer). 0.79 in. wide (1.975 cm) (total envelope) 3.87 in. high (9.675 cm) (excluding edge connector)

Safety and EMI Certification

United States	FCC part 68 ID#: EBZUSA-20078-XD-N UL: 1950 (E96804)
Canada	IC: 885 5959 A. UL: CSA 950 (E96804)
Estimated MTBF	157,000 hours per Bellcore Method I
Warranty	Intel® Telecom Products Warranty Information at http://www.intel.com/network/csp/products/3144web.htm

Spring Ware Firmware Technical Specifications**

Facsimile

Fax compatibility	ITU-T G3 compliant (T.4, T.30) ETSI NET/30 compliant
Data rate	14,400 b/s (v.17) send 9600 b/s receive
Variable speed selection	Automatic step-down to 12,000 b/s, 9600 b/s, 7200 b/s, 4800 b/s, and lower
Transmit data modes	Modified Huffman (MH) Modified Read (MR)
Receive data modes	MH MR
File data formats	Tagged Image File Format (TIFF/F) for transmit/receive MH and MR
ASCII-to-fax conversion	Host-PC-based conversion Direct transmission of text files All Windows® fonts supported Page headers generated automatically
Error correction	Detection, reporting, and correction of faulty scan lines
Image widths	8.5 in. (21.25 cm) 10 in. (25 cm) 11.9 in. (29.75 cm)
Image scaling	Automatic horizontal and vertical scaling between page sizes
Polling modes	Normal Turnaround
Image resolution	Normal (203 pels/in. x 98 lines/in.) Fine (203 pels/in. x 196 lines/in.)
Fill minimization	Automatic fill bit insertion and stripping

SpringWare Firmware Technical Specifications** (cont.)

Audio Signal

Receive range	(T-1) –40 dBm0 to +2.5 dBm0 nominal, configurable by parameter[1] (E-1) –43 dBm0 to +2.5 dBm0 nominal, configurable by parameter[1]
Automatic gain control	Application can enable/disable. Above –18 dBm0 (T-1) or –21 dBm0 (E-1) results in full-scale recording, configurable by parameter.[1]
Silence detection	–38 dBm0 nominal, software adjustable[1]
Transmit level (weighted average)	(T-1) –9 dBm0 nominal, configurable by parameter[1] (E-1) –12.5 dBm0 nominal, configurable by parameter[1]
Transmit volume control	40 dB adjustment range, with application-definable increments and legal limit cap

Frequency Response

24 Kb/s	300 Hz to 2600 Hz ±3 dB
32 Kb/s	300 Hz to 3400 Hz ±3 dB
48 Kb/s	300 Hz to 2600 Hz ±3 dB
64 Kb/s	300 Hz to 3400 Hz ±3 dB

Audio Digitizing

13 Kb/s	GSM @ 8 kHz sampling
24 Kb/s	OKI ADPCM @ 6 kHz sampling
32 Kb/s	OKI ADPCM @ 8 kHz sampling
32 Kb/s	G.726 @ 8 kHz sampling
48 Kb/s	A-law PCM @ 6 kHz sampling
64 Kb/s	A-law PCM @ 8 kHz sampling
48 Kb/s	μ-law PCM @ 6 kHz sampling
64 Kb/s	μ-law PCM @ 8 kHz sampling
Digitization selection	Selectable by application on function call-by-call basis
Playback speed control	Pitch controlled Available for 24 Kb/s and 32 Kb/s data rates Adjustment range: ±50% Adjustable through application or programmable DTMF control

DTMF Tone Detection

DTMF digits	0 to 9, *, #, A, B, C, D per CCITT Q.23
Dynamic range	(T-1) –36 dBm0 to –3 dBm0 per tone, configurable by parameter[1] (E-1) –39 dBm0 to 0 dBm0 per tone, configurable by parameter[1]
Minimum tone duration	40 ms, can be increased with software configuration
Interdigit timing	Detects like digits with a 40 ms interdigit delay. Detects different digits with a 0 ms interdigit delay.
Acceptable twist and frequency variation	(T-1) Meets Bellcore LSSGR Sec 6 and EIA 464 requirements (E-1) Meets appropriate CCITT specifications[1]
Noise tolerance	Meets Bellcore LSSGR Sec 6 and EIA 464 requirements for Gaussian, impulse, and power line noise tolerance
Cut-through	(T-1) Local echo cancellation permits 100% detection with a >4.5 dB return loss line (E-1) Digital trunks use separate transmit and receive paths to network. Performance dependent on far-end handset's match to local analog loop.
Talk off	Detects less than 20 digits while monitoring Bellcore TR-TSY-000763 standard speech tapes. (LSSGR requirements specify detecting no more than 470 total digits.) Detects 0 digits while monitoring MITEL speech tape #CM 7291.

Spring Ware Firmware Technical Specifications** (cont.)

Global Tone Detection		
	Tone type	Programmable for single or dual
	Max. number of tones	Application-dependent
	Frequency range	Programmable within 300 Hz to 3500 Hz
	Max. frequency deviation	Programmable in 5 Hz increments
	Frequency resolution	±5 Hz. Separation of dual frequency tones is limited to 62.5 Hz at a signal-to-noise ratio of 20 dB.
	Timing	Programmable cadence qualifier, in 10 ms increments
	Dynamic range	(T-1) Programmable, default set at –36 dBm0 to -0 dBm0 (single tone), –3 dBm0 (dual tone) (E-1) Programmable, default set at –39 dBm0 to +0 dBm0 per tone
Global Tone Generation		
	Tone type	Generate single or dual tones
	Frequency range	Programmable within 200 Hz to 4000 Hz
	Frequency resolution	1 Hz
	Duration	10 ms increments
	Amplitude	(T-1) –43 dBm0 to –3 dBm0 per tone nominal, programmable (E-1) –40 dBm0 to +0 dBm0 per tone nominal, programmable
MF Signaling (T-1)	R1	
	MF digits	0 to 9, KP, ST, ST1, ST2, ST3 per Bellcore LSSGR Sec 6, TR-NWT-000506 and CCITT Q.321
	Transmit level	Complies with Bellcore LSSGR Sec 6, TR-NWT-000506
	Signaling mechanism	Complies with Bellcore LSSGR Sec 6, TR-NWT-000506
	Dynamic range for detection	–25 dBm0 to –3 dBm0 per tone
	Acceptable twist	6 dB
	Acceptable freq. variation	Less than ±1 Hz
MF Signaling (E-1)	R2	
	MF digits	All 15 forward and backward signal tones per CCITT Q.441
	Transmit level	–8 dBm0 per tone, nominal, per CCITT Q.454; programmable
	Signaling mechanism	Supports the R2 compelled signaling cycle and non-compelled pulse requirements per CCITT Q.457 and Q.442
	Dynamic range for detection	–35 dBm0 to –5 dBm0 per tone
	Acceptable twist	6 dB
	Acceptable freq. variation	Less than ±1 Hz

13

SpringWare Firmware Technical Specifications** (cont.)

Call Progress Analysis		
	Busy tone detection	Default setting designed to detect 74 out of 76 unique busy/congestion tones used in 97 countries as specified by CCITT Rec. E., Suppl. #2. Default uses both frequency and cadence detection. Application can select frequency only for faster detection in specific environments.
	Ring back detection	Default setting designed to detect 83 out of 87 unique ring back tones used in 96 countries as specified by CCITT Rec. E., Suppl. #2. Uses both frequency and cadence detection.
	Positive voice detection accuracy	>99% based on tests on a database of real world calls in North America. Performance in other markets may vary.
	Positive voice detection speed	Detects voice in as little as 1/10th of a second
	Positive answering machine detection accuracy	95% based on tests on a database of real world calls in North America. Performance in other markets may vary.
	Fax/modem detection	Preprogrammed
	Intercept detection	Detects entire sequence of the North American tri-tone. Other SIT tones can be programmed.
	Dial tone detection before dialing	Application enable/disable Supports up to three different user-definable dial tones Programmable dial tone drop out debouncing
Tone Dialing		
	DTMF digits	0 to 9, *, #, A, B, C, D per Bellcore LSSGR Sec 6, TR-NWT-000506
	Frequency variation	Less than ±1 Hz
	Rate	10 digits/s, configurable by parameter[1]
	Level	–7.5 dBm0 per tone, nominal, configurable by parameter[1]
Pulse Dialing		
	10 digits	0 to 9
	Pulsing rate	10 pulses/s, nominal, configurable by parameter[1]
	Break ratio	60% nominal, configurable by parameter[1]
Analog Display Services Interface (ADSI)		
		FSK generation per Bellcore TR-NWT-000030 CAS tone generation and DTMF detection per Bellcore TR-NWT-001273

** All specifications are subject to change without notice.
[1] Configurable to meet country-specific PTT requirements. Actual specification may vary from country to country for approved products.

Hardware System Requirements

- Intel386™, Intel486™, or Pentium® microprocessor PCI bus or mixed PCI/ISA bus computer
- Operating system hardware requirements vary according to the number of channels being used
- System must comply with PCISIG Bus Specification Rev. 2.1 or later

Additional Components (with Item Market Names)

- Multidrop CT Bus cables (CBLCTB68C3DROP, CBLCTB68C4DROP, CBLCTB68C8DROP, CBLCTB68C12DROP, CBLCTB68C16DROP)
- CT Bus/SCbus adapter (CTBUSTOSCBUSADP)
- SCbus terminator kits (1SCBUS1TERMKIT, 2SCBUS1TERMKIT, 3SCBUS1TERMKIT)

To learn more, visit our site on the World Wide Web at **www.intel.com**

1515 Route Ten
Parsippany, NJ 07054
Phone: 1-973-993-3000
Fax: 1-973-993-3093

Printed in the USA Printed on recycled paper. 11/02 [illegible]