SUPERVISÃO E CONTROLE EM TEMPO REAL DE SISTEMAS ELÉTRICOS BASEADA EM COMPUTAÇÃO GRID

Pedro Daniel Zarur

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

_____

Prof. Aloysio de Castro Pinto Pedroza, Dr.

_____

Prof. Jorge Lopes de Souza Leão, Dr.Ing

_____

Eng. Marco Antonio Macciola Rodrigues, D.Sc

RIO DE JANEIRO, RJ – BRASIL

FEVEREIRO DE 2005

ZARUR, PEDRO DANIEL

Supervisão e Controle em Tempo Real de Sistemas Elétricos baseada em Computação Grid [Rio de Janeiro] 2005

IX, 128 p. 29,7 cm (COPPE/UFRJ, M.Sc.,Engenharia Elétrica, 2005)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Supervisão e controle em tempo real.
2. Computação grid.

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada á COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.).

SUPERVISÃO E CONTROLE EM TEMPO REAL DE SISTEMAS ELÉTRICOS BASEADA EM COMPUTAÇÃO GRID

Pedro Daniel Zarur

Fevereiro/2005

Orientador: Aloysio de Castro Pinto Pedroza

Programa: Engenharia Elétrica

Com o avanço da digitalização na operação de sistemas elétricos, verifica-se que, hoje, a questão não é mais apenas supervisionar e controlar sistemas elétricos. O grande desafio é difundir a informação obtida de forma segura, eficiente e consistente. A computação grid foi concebida com a finalidade de proporcionar a universalização do acesso a recursos computacionais de forma segura, consistente e de baixo custo. O objetivo deste trabalho é avaliar a utilização da arquitetura grid no projeto de grandes sistemas de supervisão e controle.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc).

REAL TIME SUPERVISORY AND CONTROL OF POWER SYSTEMS BASED ON GRID COMPUTING

Pedro Daniel Zarur

February/2005

Advisor: Aloysio de Castro Pinto Pedroza

Department: Electrical Engineering

With the progress of the digitization in the operation of power systems, it's verified that, today, the subject is not just to supervise and to control electric power systems. The great challenge is to diffuse the obtained information in a safe, efficient and solid way. The grid computing was conceived with the purpose of providing universal access to computational distributed resources. The objective of this work is to evaluate the use of the architecture grid in the project of real time supervision systems and control.

Sumário

1 - Introdução

O estado atual da computação digital é análogo, em muitos aspectos, à eletricidade por volta de 1910 [1]. Naquela época, já existia um grande mercado para equipamentos elétricos, porém a necessidade de cada usuário, de manter seu próprio gerador restringia muito sua utilização. O desenvolvimento revolucionário não foi de fato a descoberta da eletricidade, mas a infra-estrutura associada à geração, transmissão e distribuição de energia elétrica. Juntas essas tecnologias formaram um serviço robusto, padronizado e de baixo custo, tornando universal o acesso à eletricidade. O grid é uma infra-estrutura computacional (hardware e software) que permite o compartilhamento de recursos computacionais de forma universal, segura, consistente e de baixo custo [1]. A computação grid associada ao avanço tecnológico dos canais físicos utilizados na implementação de redes computacionais locais e de longa distância, possibilitará o acesso universal a recursos computacionais, sejam eles hardware ou software independentemente de onde quer que eles estejam geograficamente localizados e de forma transparente a quem estiver utilizando. De fato o termo Grid foi escolhido por analogia às redes elétricas.

O objetivo deste trabalho é analisar a utilização dessa tecnologia no desenvolvimento de aplicativos de supervisão e controle em tempo real de sistemas elétricos. Serão analisadas soluções para o estabelecimento de ligações dinâmicas entre componentes de um sistema de supervisão e controle em tempo real e também construções interessantes que permitem o compartilhamento de recursos computacionais, tais como a implementação de servidores de armazenamento de dados históricos da operação e provedores de processamento remoto de aplicações de alto desempenho. Também serão avaliadas as ferramentas e mecanismos disponíveis para integração com o sistema legado.

1.1 Estado Atual

Com o avanço da digitalização na operação de sistemas, verifica-se que hoje, a questão não é meramente supervisionar e controlar sistemas elétricos. Além do grande crescimento das bases monitoradas, os dados da operação do sistema transcenderam à sala de controle, sendo utilizados pelos vários departamentos das empresas de energia elétrica. O grande desafio é integrar o enorme volume de informação obtida de forma segura, eficiente e ao menor custo possível. Atualmente os centros de supervisão e controle funcionam como concentradores de informação sobre a operação de uma determinada área elétrica. Com a interligação entre as diversas áreas elétricas, surgiu a necessidade de centros que tivessem uma visão mais global do sistema elétrico interligado. Foram criadas então redes de supervisão e controle interligando vários centros e, permitindo o intercâmbio de informações necessárias para operação das áreas elétricas interligadas.

Naturalmente estabeleceu-se uma organização hierárquica entre os centros de supervisão e controle que se refletiu na arquitetura da comunicação de dados. Atualmente, as ligações de supervisão e controle baseiam-se em protocolos de comunicação que estabelecem relacionamentos estáticos e fortemente acoplados, voltados principalmente para robustez e alto desempenho em detrimento da flexibilidade. Qualquer reconfiguração da rede elétrica monitorada deve ser feita *off-line*, obrigando a desativação de todo sistema de supervisão e controle em tempo real, para que as mudanças sejam efetuadas. Na figura abaixo vemos uma configuração típica de rede de supervisão e controle de uma empresa de energia elétrica.

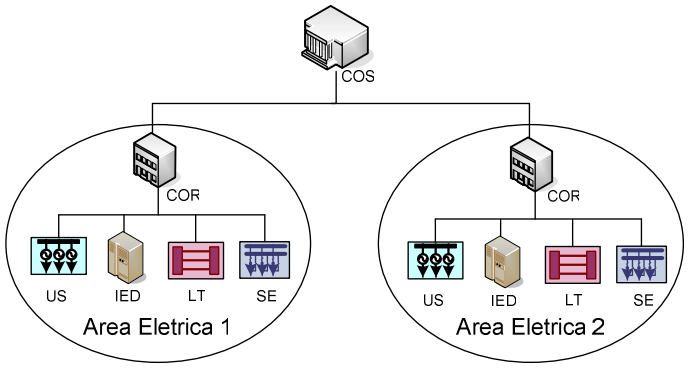


Figura 1-1: Organização hierárquica entre centros de controle.

O modelo ilustrado da Figura 1-1 pode ser observado em várias empresas do setor de geração, transmissão e distribuição de energia elétrica. Uma análise dessa estrutura nos permite observar algumas características do modelo:

- A interação entre os centros se restringe à troca de informações entre níveis hierárquicos;
- Não há compartilhamento de recursos. Os eventuais investimentos feitos em um determinado centro de controle não podem ser compartilhados pelos demais centros.
- Arquitetura pouco flexível, entrada ou saída de um centro afeta aos demais. Por exemplo, a saída de um centro de um nível superior prejudica a comunicação entre os centros de nível imediatamente inferior.

Ao subir de nível na hierarquia existe uma perda de informação. Um centro não tem acesso a todas as informações disponíveis em outro centro dois níveis abaixo do seu.

1.2 Arquitetura Grid e Redes de Supervisão e Controle

Baseados na arquitetura Grid, os centros de controle seriam organizados em grids computacionais que podem ser intra ou interinstitucionais. Nesta arquitetura, cada centro teria acesso direto e seguro aos serviços disponibilizados por outro centro sem a necessidade de intermediação. Além disso, a arquitetura é versátil a ponto de permitir reconfigurações dinâmicas da rede de supervisão e controle. Esta característica, somada à implementação de redundâncias, permitiria a migração das funções de supervisão e controle de áreas distintas entre os centros membros. A arquitetura permite, ainda, configurações interessantes nas quais se integrariam novos tipos de participantes na comunidade de supervisão e controle. Provedores de aplicações e armazenamento, por exemplo, permitiriam a construção de históricos mais precisos e o acesso a aplicações que necessitam de grande capacidade computacional. Estes novos recursos estariam disponíveis a todos os membros diluindo custos de investimento e proporcionando uma visão única de dados do sistema.

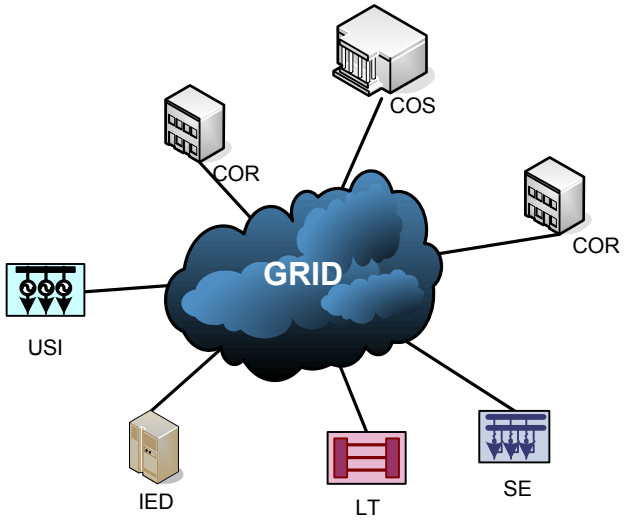


Figura 1-2: Grid de supervisão e controle, integração dinâmica de recursos

Dentre outras, podemos destacar as seguintes facilidades na manutenção de redes de supervisão em controle quando adotada a computação grid.

- Acesso universal. A aquisição de dados é feita diretamente a partir de sua origem sem intermediários, conexões ponto a ponto.
- Investimentos em recursos computacionais podem ser compartilhados por todos.
- Capacidade de um centro substituir outro em eventualidades
- Versatilidade. A rede de supervisão e controle pode ser reconfigurada dinamicamente, ou seja, a entrada ou saída de um componente só afeta aos que dependem diretamente da informação produzida por este, não afetando nenhum outro centro. Essa característica acrescida de redundâncias entre componentes garante alta disponibilidade do sistema.

1.3 Estrutura do trabalho

Uma vez exposta a motivação, cabe agora descrever como este trabalho foi estruturado.

No capítulo 2 é feita uma introdução às tecnologias e ferramentas analisadas no estudo. Inicialmente é feita uma introdução à computação grid onde é apresentada sua arquitetura básica, principais características. Em seguida, ainda no tópico de computação em grid, será apresentado o Globus Toolkit, um pacote de ferramentas para desenvolvimento de aplicações grid. No tópico seguinte, é feita uma incursão nos principais aspectos de alguns dos protocolos de comunicação de dados mais utilizados na interligação de componentes da rede de supervisão e controle. Neste tópico são analisados os protocolos ICCP/TASE.2 e DNP3.0.

No terceiro capitulo é feito um aprofundamento na proposta do trabalho, iniciando com uma visão do estado atual da operação de sistemas elétricos, para em seguida projetarmos como seria a implementação de um grid de supervisão e controle em tempo real, descrevendo seus componentes. Ao final do capitulo é proposto um protótipo de um grid de supervisão e controle implementado em laboratório para validação da tecnologia e análise de suas principais características. O penúltimo capítulo é reservado à descrição do protótipo proposto e à especificação dos ensaios feitos sobre o protótipo.

O último capítulo é reservado para observações sobre o impacto da adoção da computação em grid em redes de supervisão e controle em tempo real avaliando os seguintes aspectos:

- Flexibilidade de reconfiguração;
- Integração com sistema legado;
- Segurança;
- Facilidade de implementação de novos serviços;

Ao final são feitas considerações sobre o uso da tecnologia e o futuro das redes de supervisão e controle.

2 - Conceitos

Neste capítulo são apresentadas as principais características das tecnologias analisadas durante a confecção desse trabalho. Inicialmente, é feita uma introdução à computação grid na qual são apresentadas as características básicas da arquitetura e uma introdução ao Globus Toolkit. No tópico seguinte, é feita uma breve descrição dos protocolos de comunicação para supervisão e controle de sistemas elétricos que foram analisados para confecção deste trabalho.

2.1 Computação em Grid

Desenvolvida inicialmente a fim de atender às necessidades reais e imediatas da comunidade científica, a computação em grid possui um excelente pedigree, pois por trás dela, está o mesmo grupo que desenvolveu a WWW - *World Wide Web*, os pesquisadores da área de física de partículas. A geração atual de aceleradores de partículas gera uma enorme massa de dados a grande velocidade, e a tecnologia foi desenvolvida para viabilizar o processamento, análise e interpretação desses dados por milhares de pesquisadores ao redor do mundo. Considerado o próximo passo evolucionário na tecnologia para internet, ao contrário de sua antecessora, a WWW que implementa o intercâmbio de informações, o grid é voltado ao compartilhamento de recursos computacionais, ou seja, capacidade de processamento, espaço para armazenamento, acesso a aplicações, etc. O usuário final, quando conectado, vê o grid como um grande computador [40].

A computação grid estabelece que todos os relacionamentos entre os componentes de um grid computacional sejam dinâmicos, ou seja, podem ser estabelecidos a qualquer instante e de forma desacoplada, isto é, um usuário de um serviço não sabe de antemão quem vai servi-lo. A estrutura de um grid computacional é definida a partir de um conjunto de regras dinâmicas que estabelecem claramente o papel de cada participante, determinando quem pode fazer o quê, quando e onde. As principais características dos grid computacionais são:

- **Dinamismo**: Permite a entrada e saída de membros a qualquer momento;
- **Heterogeneidade**: Permite a convivência entre as mais diversas tecnologias computacionais;
- **Compartilhamento**: Recursos computacionais são vistos como um insumo disponível universalmente;
- **Transitoriedade**: Todo relacionamento tem o tempo de vida pré-determinado e expansível;
- **Segurança**: Utiliza, como base de sua construção, regras bem definidas sobre o papel de cada membro;
- **Desacoplamento**: O usuário de um determinado serviço não sabe de antemão quem vai servi-lo;
- **Arquitetura**: Baseada em serviços.

Sua arquitetura implementa duas construções básicas, a primeira consiste de clusters de computadores onde centenas, até milhares de máquinas são interligadas e funcionam como se fosse um único supercomputador de processamento paralelo. A segunda construção básica consiste em comunidades virtuais, que agregam todos os participantes do grid, como mostra a Figura 2-1.

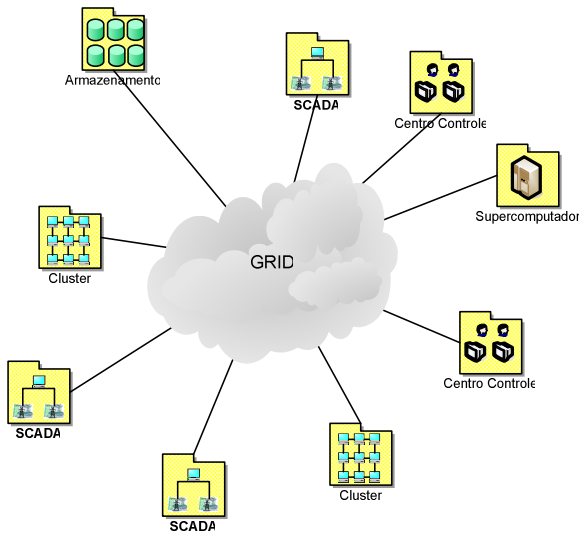


Figura 2-1: Comunidade virtual e seus participantes

Neste trabalho daremos ênfase às comunidades virtuais, pois este conceito é o que melhor se encaixa na proposta de se criar grids para supervisão e controle de sistemas elétricos. Uma comunidade virtual é definida a partir de regras, que estabelecem quem são seus participantes e quais operações lhe são permitidas. Segurança é um aspecto fundamental na computação grid, todo grid tem que ter pelo menos um componente certificador que é responsável pela autenticação e autorização dos participantes.

Em um grid computacional cada participante pode ser visto como um provedor e/ou consumidor de serviços que são acessados de forma muito semelhante a um site da internet. Cada componente do grid possui um identificador único denominado GSH, *Grid Service Handler*, que é muito semelhante às URL's (*Uniform Resource Locator*) utilizadas para acessar páginas na web. Porém as semelhanças param por aí. Enquanto na web um relacionamento cliente servidor se baseia apenas em mecanismo de troca de informações do tipo pergunta/resposta, os relacionamentos em um grid computacional possuem um ferramental muito mais sofisticado do qual podemos destacar os seguintes mecanismos que fazem parte de sua implementação básica:

- **Notificação**: Um componente qualquer pode solicitar que outro componente o notifique a ocorrência de um evento pré-determinado.
- **Fábrica/instância**: Um provedor cria uma instância dos seus serviços para atender a uma determinada solicitação de um cliente.
- **Dados de serviço**: É possível associar informações aos serviços. Os dados podem ser livremente estruturados em elementos denominados SDE, *Service Data Elements*, que podem ser utilizados para pesquisa e indexação de recursos no grid.

Uma descrição mais detalhada sobre a arquitetura grid e sua implementação pode ser encontrada em [2,3,5,14,15,46]

2.1.1 Globus Toolkit 3.2.1

O Globus Toolkit é um software de código aberto para desenvolvimento de grids computacionais conforme a definição feita pela OGSA, *Open Grid Services Architecture*, que implementa a infra-estrutura de serviços especificada pela OGSI, *Open Grid Services Infrastructure*.[46] Segundo a infra-estrutura implantada, todo o componente ativo do grid é uma instância de uma entidade denominada **GridService,** que já traz os mecanismos descritos anteriormente e que na versão 3.2.1 é implementado como uma extensão da tecnologia *Web Services*. A Figura 2-2 que foi extraída de [46] faz um bom resumo para o entendimento.

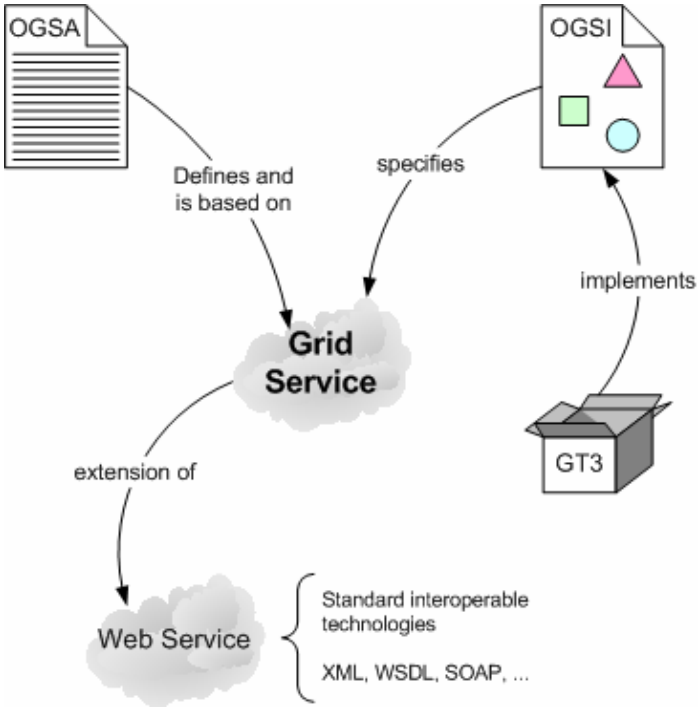


Figura 2-2: GT3-Globus Toolkit, arquitetura e infra-estrutura

2.1.2 Componentes do Globus Toolkit

Os componentes do Globus Toolkit são organizados em dois grupos segundo sua implementação. No primeiro grupo encontram-se os componentes baseados em *web services*. O segundo grupo contêm os componentes denominados *pre-web services*, com implementações compatíveis com versões anteriores do Globus. Neste trabalho só foram utilizados componentes do primeiro grupo. A seguir são mostrados os componentes com uma breve descrição de suas funcionalidades.

2.1.2.1 Serviços Básicos

O componente é responsável pela implementação da infra-estrutura especificada pela OGSI. É o componente central do Globus onde são definidas as funcionalidades básicas dos serviços do grid. Uma descrição mais detalhada pode ser encontrada em [5,17,46].

2.1.2.2 Segurança

A infra-estrutura disponível no Globus atende aos requisitos mais atuais de segurança. Estão presentes mecanismos de certificação baseado no padrão X-509, comunicação baseada no protocolo SSL, *Secure Sockets Layer.* e criptografia baseada em chaves publica e privada. Os serviços de segurança são implementados por dois pacotes. O GSI, *Grid Security Infrastructure* é responsável pela infra-estrutura de certificação e criptografia. E o CAS, *Community Authorization Service,*permite a implantação de regas de acesso aos recursos disponíveis no grid. Uma descrição mais detalhada pode ser encontrada em [5,10,46].

2.1.2.3 Gerenciamento de dados

Uma das funcionalidades mais importantes da computação grid é a capacidade de manipular grandes massas de dados. Essa tarefa é executada por três serviços do Globus. O GridFTP é um protocolo para transferência de arquivos baseado no conhecido FTP, *File Transfer Protocol*, que implementa mecanismos avançados que permitem uma transferência mais eficiente de dados. O RFT, *Reliable File Transfer*, é responsável pela gerência das transferências de dados, implementa mecanismo de

recuperação de erros permitindo operação mais robusta O RLS, *Replica Location Service* é um serviço de manutenção de cópias de dados no grid. Uma descrição mais detalhada pode ser encontrada em [5].

2.1.2.4 Gerenciamento de recursos

O serviço GRAM, *Grid Resource Allocation and Management,* disponibiliza uma interface para utilização de recursos computacionais remotos na execução de aplicações. Suas funcionalidades não foram exploradas no escopo deste trabalho Uma descrição mais detalhada pode ser encontrada em [5].

2.1.2.5 Serviços de Pesquisa e Indexação

O MDS, *Monitoring and Discovery System*, implementa o serviço de informações sobre o grid computacional. Através dele é possível determinar os recursos disponíveis no grid e em que estado se encontram. Uma descrição mais detalhada pode ser encontrada em [5,45,46].

2.1.2.6 XIO

O XIO, *eXtensible Input and Output,* disponibiliza uma API que permite o desenvolvimento de soluções customizadas para comunicação e dados. Esse componente é chave da integração de sistemas de supervisão e controle legados por permitir a implementação de virtualmente qualquer protocolo de comunicação através de uma API que permite um bom grau de abstração dos detalhes da comunicação. Uma descrição mais detalhada pode ser encontrada em [5].

2.2 Protocolos de comunicação

A fim de determinar as necessidades a serem atendidas por um grid de supervisão e controle em tempo real de redes elétricas foi feito um breve estudo sobre as funcionalidades oferecidas por alguns dos principais protocolos de comunicação utilizados atualmente.

2.2.1 ICCP/TASE.2

O ICCP, Inter Control Center Protocol, também conhecido como TASE.2 *(Tele-control Application Service Element*) , é um protocolo para integração de centros de controle, que utiliza uma arquitetura cliente-servidor, e seus elementos são modelados como objetos, sendo os serviços do protocolo implementados como métodos desses objetos. O ICCP/TASE.2 é definido pelas normas IEC/60870-6-503, IEC/60870-6-802, IEC/60870-6-702 e IEC/60870-6-505, sendo um dos vários protocolos da arquitetura UCA (*Utilities Communication Architecture*), que é baseado na especificação MMS (*Manufacturing Message Specitication*), definida nas normas ISO/9506-1 e ISO/9506-2. Sua arquitetura prevê, porém não especifica, uma API para integração de aplicações.

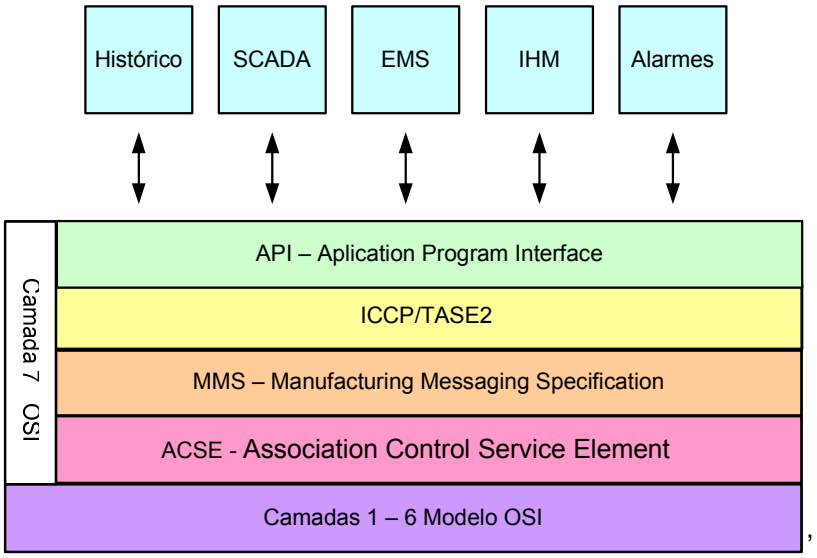


Figura 2-3: Arquitetura ICCP/TASE.2

As funcionalidades do protocolo são organizadas em blocos que podem ser implementados em módulos:

Bloco 1 Contém o conjunto mínimo de objetos e serviços que devem ser implementados para troca de dados em tempo real.

Bloco 2 Estende a capacidade de transmissão de dados tempo real. Implementa o envio de dados por evento.

Bloco 3: Implementa a transferência de dados em blocos.

Bloco 4 Implementa troca de mensagens entre centros de controle.

Bloco 5: Implementa envio de mensagens de controle aos equipamentos elétricos.

Bloco 6: Implementa mecanismos de controle de aplicações no servidor.

Bloco 7 Implementa a mecanismo de seqüência de eventos.

Bloco 8 Permite a definição de objetos pelo usuário.

Bloco 9: Implementa transferência de séries temporais.

O ICCP por ser um protocolo voltado à interligação entre centros de controle e aplicações, é uma excelente base para determinar quais a funcionalidades mínimas um grid de supervisão e controle deve atender.

2.2.2 DNP3

O DNP3, criado pela GE Harris em 1990, é um protocolo de domínio publico baseado nas primeiras especificações do IEC/60870-6-802,. Com uma implementação mais simples que a do ICCP suas principais funcionalidades são:

- Enviar e receber dados de tipos diferentes em uma mesma mensagem.
- Quebrar as mensagem em pequenos pacotes de forma a garantir um mecanismo de detecção e recuperação de erros robustos.
- Somente envia medidas que variaram.
- Permite estabelecimento de prioridades entre os serviços
- Implementa mecanismo de notificação
- Suporta mecanismos de sincronização.
- Permite múltiplos mestres e operações ponto a ponto.
- Possibilita o envio de arquivos.

Apesar de mais simples, as características do DNP3 dão um bom exemplo das funcionalidades mínimas, em termos de envio/recebimento de dados, necessárias pelo grid de supervisão e controle.

2.3 Grid de Supervisão e Controle

No próximo capítulo será apresentada a proposta conceitual desse trabalho que é o desenvolvimento de sistemas de supervisão e controle de redes elétricas em tempo real baseados em computação grid. Será modelada uma solução para a operação dos sistemas elétricos de uma empresa fictícia baseada no que será denominada “grid de supervisão e controle”, uma entidade que agrega todos os recursos computacionais da empresa destinados á operação de sistemas elétricos em uma organização virtual conforme definida pela OGSA.

3 - Proposta Conceitual

Neste capítulo, é apresentada a idéia central desse trabalho, uma nova geração de aplicações de apoio à operação de sistemas elétricos baseadas em computação grid. É importante nesse momento enfatizar que todas as soluções foram idealizadas a partir da arquitetura definida pela OGSA, *Open Grids Services Architecture* [1,2], e das funcionalidades definidas pela OGSI, *Open Grid Services Infrastructure* [1,3,14,15] e nos recursos disponíveis no Globus Toolkit versão 3.2.1[5] .

Inicialmente é proposta uma empresa concessionária de energia elétrica fictícia operando um determinado sistema elétrico. Esse exemplo serve como base para uma breve análise comparativa das características de rede de supervisão e controle em tempo real utilizando a tecnologia disponível atualmente com a solução baseada na computação grid. Em seguida são apresentados em tópicos alguns dos aspectos considerados mais importantes em um grid de supervisão e controle. Onde são abordados os seguintes temas:

- O estabelecimento de ligações dinâmicas entre os componentes do grid,
- A implementação do compartilhamento de recursos computacionais.
- A integração das novas aplicações com sistema legado,
- Finalmente é proposto um protótipo de um grid de supervisão e controle para testes de mecanismos e funcionalidades em laboratório.

3.1 Rede de supervisão e controle

Considere o seguinte cenário. Uma concessionária de energia é responsável pela operação de uma determinada área elétrica. A área supervisionada é composta por duas usinas e oitos subestações. A operação do sistema elétrico é executada por três centros de controle regionais, que denominados Centros de Operação Regional - COR, sendo o primeiro responsável pelas usinas e por duas subestações e os outros centros responsáveis por três subestações cada. Encabeçando a estrutura existe um centro de controle responsável pela supervisão do sistema interligado, que denominaremos Centro de Operação do Sistema – COS, como é mostrado na Figura 3-1.

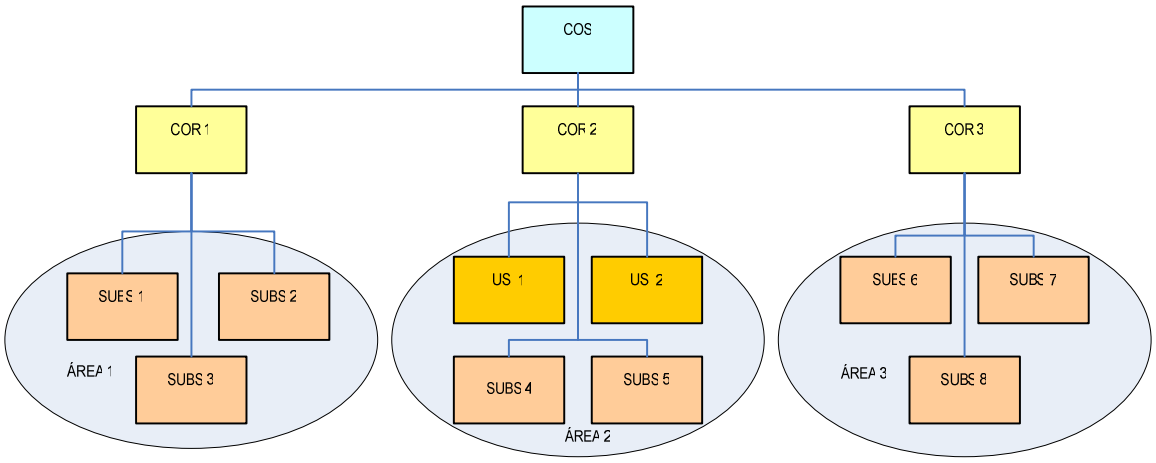


Figura 3-1: Estrutura de supervisão e controle

Em cada centro da empresa existe uma aplicação que implementa uma interface homem máquina, IHM, utilizada na supervisão e controle das instalações hierarquicamente inferiores. As instalações elétricas da empresa são totalmente digitalizadas com equipamentos de diversos fabricantes, e que se comunicam com os sistemas dos centros com protocolos de comunicação diferentes. A empresa tem um contrato com uma concessionária de telecomunicações local que lhe provê a interligação entre todas as suas instalações através de uma rede de pacotes padrão Ethernet., e sobre essa rede é construída a infra-estrutura de supervisão e controle. Para enriquecer o exemplo, no COS da empresa são instalados: um supercomputador com softwares de última geração e um sistema de banco de dados com capacidade de armazenamento de centenas de terabytes.

3.1.1 Estado atual

Em uma implementação com a tecnologia atual as ligações entre os componentes da rede de supervisão e controle são configuradas de forma off-line, através de conexões ponto a ponto e provavelmente baseadas em algum dos protocolos de comunicação apresentados no capítulo 2. Ao serem ativadas, as ligações entram em funcionamento e assim ficam até a desativação de alguma das partes da conexão ponto a ponto. Qualquer alteração na configuração da comunicação de dados ou até em uma área elétrica monitorada obriga a desativação dos sistemas envolvidos. A eventual queda de um COR influencia na operação interligada. O remanejamento das funções do centro

em falta para outro centro tem de ser feita de forma off-line impactando toda a rede de supervisão e controle.

Os recursos computacionais do COS da empresa não estariam acessíveis aos outros centros. Por exemplo, para fazer um histórico da operação de um determinado COR a empresa terá de investir na implantação de toda uma estrutura de banco de dados naquela instalação. Da mesma forma os serviços do supercomputador só estariam disponíveis ao COS da empresa.

Mais adiante será visto que a computação grid oferece meios de estabelecer ligações dinâmicas entre os centros de controle e as instalações da empresa. Permitindo que reconfigurações da rede de supervisão e controle tais como a saída ou entrada de um centro ou planta elétrica sejam feitas sem afetar o restante da rede. Será mostrado também que recursos computacionais como o sistema de banco e dados e o supercomputador, localizados no COS da empresa se tornarão componentes do grid que poderão ser acessados por quaisquer outros componentes permitindo o compartilhamento desses recursos.

3.1.2 Grid de supervisão e controle

De volta ao exemplo, na composição do grid de supervisão e controle da empresa fictícia foram incluídos as plantas elétricas, os centros de controle, o banco de dados e o supercomputador. Além disso, foram adicionados dois novos componentes: um serviço de autenticação e autorização, AUT, que é responsável pela segurança e um serviço de pesquisa e indexação, DIR, onde são publicados os recursos disponíveis no grid e que podem ser pesquisados pelos demais componentes. Esses novos componentes serão analisados detalhadamente mais adiante. A partir deste conjunto de componentes podemos representar o grid de supervisão e controle da empresa conforme a Figura 3-2.

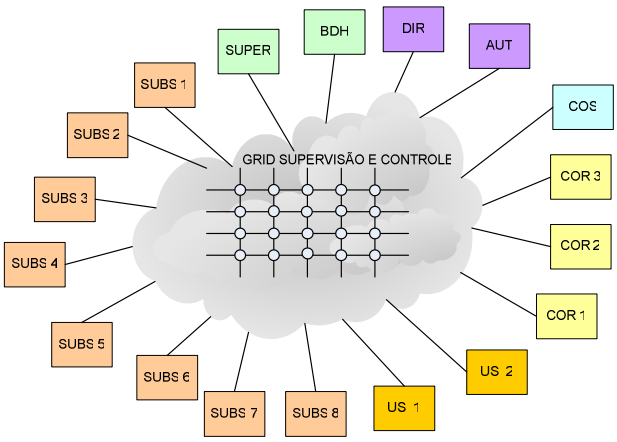


Figura 3-2: Grid de supervisão e controle

O primeiro impacto detectado é o aumento da flexibilidade da rede. Por exemplo, para o COR2 substituir o COR1 basta o COR2 estabelecer ligações com subestações SUBS1, SUBS2 e SUBS3 o que pode ser feito de forma dinâmica e sem afetar o restante dos serviços de supervisão. Essa característica aumenta em muito a capacidade da rede de supervisão e controle resistir à falhas. Verificamos também que o COS não precisa mais aquisitar as medidas a partir dos COR's e sim diretamente das plantas elétricas mais uma vez aumentando a robustez da rede e eliminando gargalos de informação fatores muito importantes em sistemas que a alta disponibilidade é um fator crítico.

Outro aspecto interessante é que os serviços do supercomputador e do banco de dados estão disponíveis para todos os participantes, e da mesma forma que se estabelece uma ligação entre um centro e uma instalação elétrica, pode-se estabelecer uma ligação com um desses recursos. Ao entrar em operação uma usina pode solicitar ao banco de dados que faça um histórico de seu funcionamento, e a partir daí é estabelecida uma ligação onde periodicamente a usina enviará séries temporais com suas medidas que serão armazenadas no banco de dados. Da mesma forma podemos imaginar o supercomputador executando um sistema complexo, que necessita grande capacidade de processamento, disponível para todos os centos de controle.

Os novos componentes introduzidos trazem funcionalidades muito importantes para o grid de supervisão e controle. O serviço de pesquisa e indexação possibilita, por exemplo, que uma instalação elétrica publique características de forma que os demais componentes possam efetuar pesquisas e obtenham informações de como acessá-las.

O servidor de autenticação e autorização é responsável pela segurança do grid de supervisão e controle. As atribuições deste componente são autenticar os outros componentes e informar quais são as operações permitidas em um determinado contexto. Por exemplo, durante o estabelecimento de uma ligação entre uma instalação elétrica e um centro de controle, o servidor de autenticação e autorização é consultado para confirmar a identidade dos componentes e verificar a que medidas o centro de controle tem acesso naquela instalação, ou se o centro tem permissão para enviar controles operativos.

3.2 Ligações dinâmicas

Neste tópico é feita uma análise do estabelecimento de ligações de supervisão e controle em tempo real, baseado nos mecanismos e protocolos disponíveis no Globus Toolkit. O estabelecimento de ligações dinâmicas é certamente a característica da computação grid de maior impacto em sistemas de supervisão em tempo real. Sob o ponto de vista dos grids de supervisão e controle, uma ligação consiste em uma instância de serviços, criada por um determinado componente no papel de um servidor, que execute de forma robusta e eficiente uma comunicação de dados que implemente o envio periódico de valores medidos, estados operacionais e que também permita o envio comandos operativos ao sistema elétrico. A figura abaixo caracteriza uma ligação de supervisão e controle baseada em computação grid integrada ao sistema legado através do Globus XIO como detalhado mais adiante.

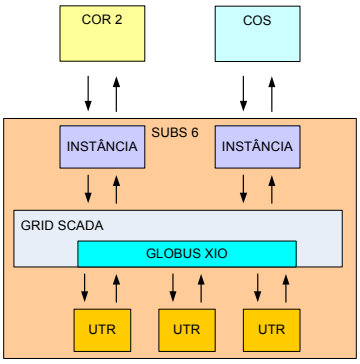


Figura 3-3: Ligação de supervisão e controle

Por default o protocolo SOAP é utilizado para o acesso aos serviços de um componente do grid [3], devido a sua grande versatilidade ele é a escolha ideal para execução de

tarefas mais complexas em detrimento da performance prejudicada pelo tamanho de suas mensagens que são codificadas em XML. Com certeza esse não é o protocolo mais indicado para a transmissão de medidas e estados operacionais devido ao tamanho das mensagens geradas e as necessidades de performance inerentes a um sistema de tempo real. Então podemos dividir o tempo de vida de uma ligação em duas partes, a primeira na qual são executados os procedimentos para estabelecimento da ligação que é baseada em SOAP e a segunda na qual uma vez estabelecida a ligação, a comunicação entre os serviços se processa, através de um protocolo de comunicação mais eficiente. Com já vimos no capitulo 2 o Globus Toolkit contém o Globus XIO que disponibiliza uma API que permite o desenvolvimento de protocolos de comunicação de dados eficientes. O Globus XIO é um componente chave na implantação de grids de supervisão e controle, pois além de implementar uma comunicação eficiente ele permite a integração com o sistema legado como veremos mais adiante.

3.2.1 Estabelecendo uma ligação

Suponha que em nossa empresa fictícia a SUBS5 esteja prestes a entrar em operação na AREA2 que é responsabilidade do COR2. A seguir são descritos, passo a passo, os procedimentos para o estabelecimento de uma ligação de supervisão e controle em tempo real entre o centro e a subestação conforme a Figura 3-4.

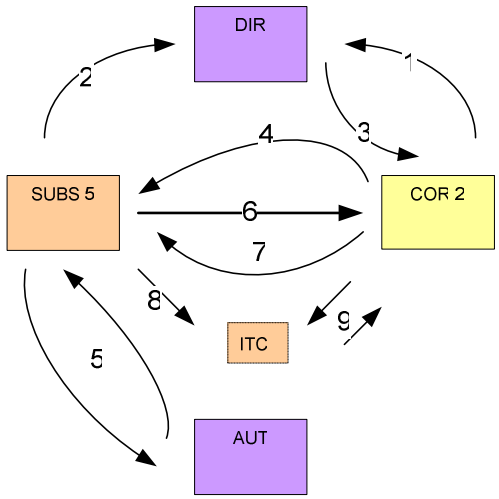
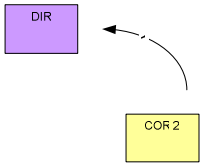
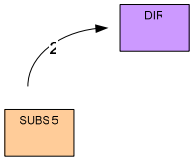


Figura 3-4: Estabelecendo uma ligação dinâmica

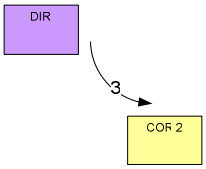
1. Inicialmente o centro de controle COR2 publica seus serviços no serviço de pesquisa e indexação, DIR, e solicita ser notificado toda vez que um componente pertencente à AREA2 fizer uma publicação de serviços.



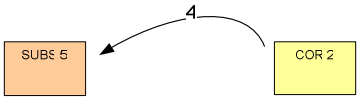
2. Ao entrar em operação a SUBS5 publica seus serviços disponíveis em DIR, passando informações tais como seu endereço, a área elétrica a que pertence, lista de pontos de medição, protocolos de comunicação, etc.



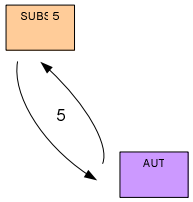
3. Atendendo à solicitação feita no primeiro passo, DIR verifica que SUBS6 pertence a AREA2 e notifica COR2. O modelo de notificação implementado do Globus Toolkit é do tipo push, ou seja, junto com a notificação podem ser enviadas informações relativas ao evento. No caso é enviado o endereço de SUBS5 no grid de supervisão e controle que em computação grid é denominado GSH, Grid Service Handler. Em um grid computacional o GSH de um componente é único, ou seja, não podem existir dois componentes com mesmo GSH.



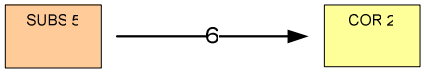
4. De posse do endereço COR2 faz um acesso à SUBS5 a fim de descobrir de que forma pode ser estabelecida uma ligação de supervisão e controle com a subestação.



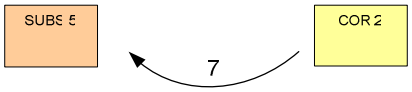
5. Ao ser acessada a subestação solicita ao servidor de autenticação e autorização, AUT, uma lista das operações que COR2 está autorizado a fazer. Por exemplo, quais medidas podem ser acessadas, ou se o centro tem permissão para efetuar controles operativos. Mais adiante será detalhada a implementação da segurança nos grids de supervisão e controle no momento basta saber que SUBS6 confirmou a identidade de COR2 e obteve uma lista das operações permitidas.



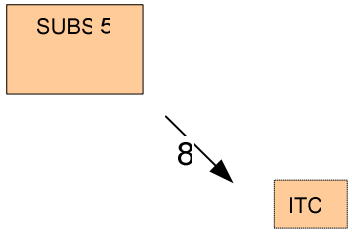
6. A subestação SUBS5 envia para centro de controle COR2 um documento,onde são descritos todos os serviços disponíveis e a forma de acessá-los. Esse documento é denominado GSR, Grid Service Reference. O GSR geralmente tem a forma de um documento GWSDL, Grid Web Services Description Language.



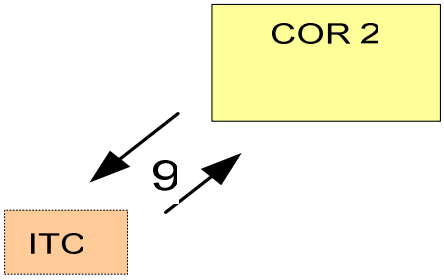
7. O COR2 faz uma chamada direta aos serviços utilizando o protocolo SOAP. O serviço pode ser, por exemplo, solicitar que SUBS5 envie, a cada 2 segundos, os valores de potência ativa e reativa dos transformadores da subestação. Ou então comandar a abertura de um determinado disjuntor.



8. Nesse passo, SUBS5 atendendo a chamada de COR2, cria uma instância do serviço solicitado por COR2. Um aspecto muito interessante da implementação é que a relação cliente X instância não é necessariamente de um para um. O servidor ao atender uma requisição pode alocar dois ou mais clientes em uma mesma instância. A decisão de criar uma nova instância ou aproveitar uma já existente é feita em tempo de execução e pode-se basear nas condições do momento. Isso dá uma grande flexibilidade ao sistema permitindo por exemplo um balanceamento de carga no atendimento a requisições.



9. Uma vez criada a instância, são executadas as tarefas requisitadas. O tempo de vida da instância é determinado pela natureza do serviço a ser executado. Por exemplo para serviços de monitoração de valores analógicos a instância não tem tempo de vida definido, uma vez criada periodicamente são enviados dados medidos de SUBS5 para COR2. Já um serviço que informa o estado de chaves e disjuntores elétrico ou se um equipamento está ou não energizado pode ser implementado através de um mecanismo de notificação que também tem tempo de vida indeterminado. O envio de comandos operativos pode ser visto como um serviço transitório que é criado no momento do envio do comando e é terminado após a confirmação da execução do serviço.



O mecanismo descrito admite diversas variações muito interessantes do ponto de vista da supervisão e controle em tempo real. Por exemplo se ao invés de ser notificado por DIR da entrada de SUBS5, COR2 poderia fazer uma pesquisa em DIR solicitando a lista das subestações pertencentes à AREA2 e em seguida estabelecer ligações com todas as subestações sob sua responsabilidade. Ou ainda, num outro exemplo, o COR 1 solicita a DIR que o notifique toda vez que o COR2 sair do ar e a partir desse evento ele assumiria o lugar do COR2 estabelecendo ligações com as usinas e subestações da AREA2. A entrada/saída de componentes se dá de forma a não prejudicar o restante das ligações.

3.2.2 Pesquisa, indexação e dados dos serviços

O MDS, *Monitoring & Discovery System*, é o componente do GLOBUS Toolkit responsável por prover informações sobre os recursos disponíveis em um grid computacional. Em nosso exemplo ele é representado pelo componente DIR, no qual os demais componentes publicam seus recursos descritos através de seus SDE's, *Service Data Elements*, que são estruturas de dados que podem armazenar informações de tipos básicos como números, cadeias de caracteres, até classes de objetos complexos. As SDE's são utilizadas para armazenar características do serviço prestado e também informações sobre o processamento do serviço. Por exemplo, uma planta elétrica poderia ser representada três SDE's. O primeiro seria o identificador da usina ou subestação, contendo informações como nome, área elétrica, número de pontos de medição, etc. O segundo contendo uma lista dos pontos e medição, e o terceiro uma lista de controles operativos disponíveis. O MDS fornece formas de pesquisa avançada sobre os SDE dos serviços registrados e ainda implementa mecanismos de notificação em cima dessas informações como veremos a seguir. Normalmente o resultado de uma pesquisa é um GSH ou uma lista de GSH's dos componentes que atendem aos parâmetros da consulta.

Nesse ponto cabe uma discussão sobre que semântica utilizada para a pesquisa de recursos elétricos em um grid computacional. Não está no escopo desse trabalho definir uma estrutura de dados que atenda a descrição de todos os recursos disponíveis em um sistema elétrico. Certamente trabalhos de padronização como o CIM, *Common Information Model*, proposto pelo EPRI como um modelo genérico para descrição de

sistemas elétricos, são de grande importância na implementação de uma semântica para pesquisas por recursos elétricos em grids de supervisão e controle.

3.2.3 Notificação

Outro mecanismo importante implementado pelo MDS é o de notificação. Em nosso exemplo, vimos que no início o COR2 solicitou à DIR que fosse notificado toda vez que uma usina ou subestação pertencente à AREA2 entrasse em operação. Esse serviço também é baseado nas SDE's, sobre a qual um cliente solicita ser notificado sempre que o valor mudar. O modelo de notificação implementado é do tipo *push*, ou seja, junto com a notificação segue o conteúdo da SDE em questão. Em nosso exemplo para notificar a entrada ou saída de plantas elétricas, foi definido em DIR um SDE contendo a GSH e o identificador da área elétrica. COR2 solicita a DIR que toda vez que a SDE definida tiver seu valor alterado seja enviada uma notificação contendo o conteúdo da SDE. Toda vez que uma nova usina ou subestação entra em operação e publica suas SDE's em DIR, o servidor de pesquisa atualiza sua própria SDE adicionando a nova planta elétrica em operação no grid e notifica COR2. Junto com a notificação segue a lista que é consultada por COR2 e no caso da inclusão de alguma planta na área sob sua responsabilidade inicia o processo de estabelecimento de uma ligação dinâmica com aquela planta.

O mecanismo de notificação não é uma exclusividade do MDS. Assim como as SDE's, a notificação é uma característica típica de todos os serviços de um grid computacional, constituindo uma poderosa ferramenta no desenvolvimento de sistemas. O mecanismo permite, por exemplo, estabelecer procedimentos para recuperação de falhas em tempo real, aumentando a resiliência dos grids de supervisão e controle.

3.2.4 Autenticação / autorização

Durante o estabelecimento de sua ligação com COR2, SUBS6 acessou AUT para autenticar de COR2 e obter uma lista de operações permitidas ao centro. No Globus Toolkit o serviço autorização é implementado pelo CAS, *Community Authorization Service*, que permite que sejam estabelecidas regas de acesso aos recursos do grid, organizadas por usuários ou grupos de usuários. Sua implementação é baseada no relacionamento entre três tabelas:

- Tabela de recursos disponíveis, contendo identificador do recurso, por exemplo, uma ou um grupo de medidas analógicas, uma chave seccionadora, uma área elétrica, uma subestação, etc.
- Tabela de ações contendo informações do tipo: ler medidas, acionar controle , ligar , desligar, etc.
- Tabela de usuários, que contém listas de componentes do grid que são certificados através de um CA, *Certificate Authority*. Em todo grid computacional deve existir pelo menos um CA.

O CAS foi construído sobre a *Grid Security Infrastructure* - GSI, que é o componente do Globus Toolkit responsável pela infra-estrutura básica de segurança. O GSI implementa serviços de autenticação, certificação e criptografia, e implementa mecanismos avançados de delegação de permissões através de procurações. A criptografia é baseada nos padrões internacionais OpenSSL e X-509, o que permite uma boa integração com CA de terceiros para emissão de certificados e chaves públicas e privadas. O Globus Toolkit traz uma implementação de CA para fins de testes, denominada Globus SimpleCA, que foi utilizada neste trabalho com ferramenta de certificação.

3.3 Compartilhamento de recursos

Podemos classificar o compartilhamento de recursos computacionais implementado pela OGSI em dois tipos. No primeiro tipo, que está fora do escopo desse trabalho, clusters com centenas, até milhares computadores trabalham em paralelo e cooperativamente a fim de executar uma aplicação do tipo "*bag of tasks*". Uma aplicação é dividida em tarefas, que são executadas de forma independente e distribuídas, pelo cluster computacional, geralmente aproveitando o tempo ocioso das máquinas. Esse tipo de compartilhamento permite a implementação de supercomputadores virtuais de baixo custo e é indicado para aplicativos que executem cálculos complexos e que possam ser distribuídos em várias tarefas independentes. Esse trabalho está voltado principalmente a aplicativos de tempo real que não se enquadram no modelo "*bag of tasks*" e apresentam requisitos rígidos de performance.

No caso de grids de supervisão e controle o compartilhamento de recursos é implementado através de provedores de serviços remotos, que agreguem poder computacional para executar tarefas que estão além da capacidade de processamento do sistema local. Em nosso exemplo configuramos o sistema de banco de dados e o supercomputador, como componentes do grid que provêm aos demais, capacidades de armazenamento de dados e processamento respectivamente. Essa característica da computação grid criará um grande impacto na forma de se investir em infra-estrutura computacional.

3.3.1 Implementando histórico da operação

Um banco de dados histórico da operação, armazena séries temporais representativas do funcionamento do sistema elétrico. Devido ao alto custo de implantação e manutenção de sistemas dessa natureza, atualmente adota-se a estratégia de implantar o histórico da operação no topo da hierarquia da rede de supervisão e controle e armazenar somente a informação que chega ao centro através das ligações de supervisão e controle em tempo real. Um dos problemas dessa estratégia é que ao subir de nível na hierarquia existe uma perda de informação natural das necessidades de cada centro, por exemplo, COR2 recebe de USI1 as medidas de potência de cada unidade geradora da usina, porém repassa ao COS somente a potência total gerada pela usina. Para a supervisão em tempo real essa informação é suficiente, porém para fins de histórico seria mais preciso se os valores de potência por unidade geradora fossem armazenados no banco de dados. Acrescentar informações às ligações de transmissão de dados em tempo real implica em um aumento do volume de informações que trafegam em um canal muito sensível a problemas de performance.

Nesse tópico é proposto um serviço de armazenamento de dados históricos da operação, disponível a todos os componentes do grid e supervisão e controle que desejem historiar seus dados de tempo real. Porém não é o objetivo desse trabalho detalhar todas as características de um sistema de histórico, e sim apenas apresentá-lo como um exemplo de implementação do compartilhamento de recursos, no caso espaço em disco, em um grid de supervisão e controle., e identificar que ferramentas o Globus Toolkit fornece para o desenvolvimento do sistema.

A estratégia adotada para gravação do histórico baseia-se no envio de um arquivo texto do tipo CSV,*Comma Separated Values*, contendo valores medidos em um determinado período, para o serviço de armazenamento para lá ser gravado em um bando de dados relacional. A transmissão dos dados é feita através do protocolo GridFTP e gerenciado através do componente RFT, *Reliable File Transfer Service*, ambos componentes do Globus Toolkit.

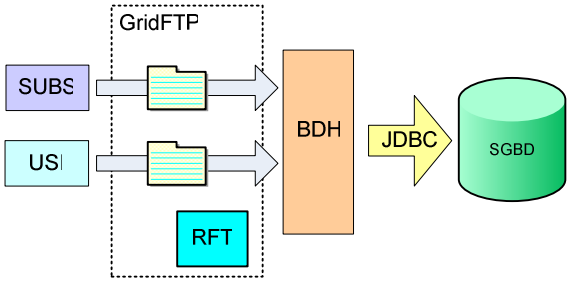


Figura 3-5: Banco de dados histórico da operação

Da mesma forma como foi feito com as ligações dinâmicas será analisado passo a passo o processo de estabelecimento do serviço de armazenagem do histórico da operação. No exemplo a título de simplificação foi omitido o procedimento de autenticação e verificação de autorização pois a implementação é muito semelhante à vista anteriormente no estabelecimento de uma ligação dinâmica.

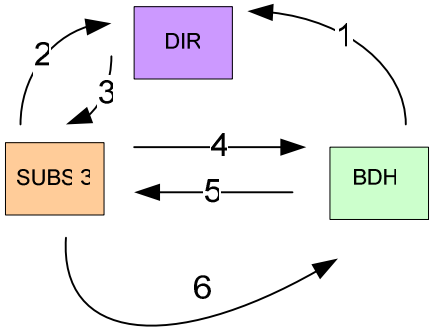
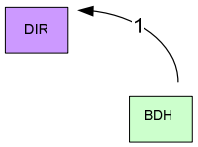
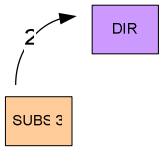


Figura 3-6: Serviço de histórico da operação

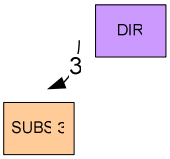
1. Ao ser ativado o serviço e imediatamente, publica informações sobre seus serviços no serviço de pesquisa e indexação do grid de supervisão e controle, disponibilizando seu GSH aos demais componentes do grid. Os SDE's desse serviço devem conter informações que identifiquem o serviço prestado e informações adicionais tais como espaço disponível em disco GB, cota por cliente, etc.



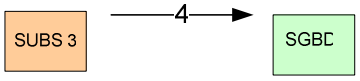
2. Ao entrar em operação SUBS 3 executa uma pesquisa em DIR procurando algum componente que armazene seus dados históricos da operação.



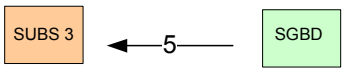
3. DIR envia em resposta à pesquisa de SUBS 3 o endereço no grid de BDH. Como já foi visto o endereço de serviço grid é representado pelo seu GSH, Grid Service Handler, e tem o formato de um URI, Universal Reference: Indentifier que é muito parecido com os endereços que utilizamos para acessar páginas na web.



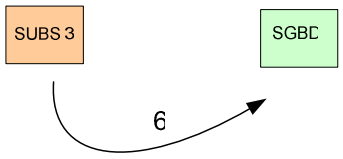
4. De posse do GSH de BDH, SUBS6 acessa o servidor de histórico e solicita uma descrição de seus serviços e como acessá-los. Da mesma forma que no processo de estabelecimento de uma ligação dinâmica, a comunicação é feita com protocolo SOAP.



5. BDH responde a solicitação de SUBS 3 enviando seu GSR, Grid Service Reference, que contém a descrição dos serviços. O GSR, no caso da comunicação estar baseada no protocolo SOAP, tem a forma de um documento do tipo GWSDL, Grid Web Services Description Language, que é uma extensão do padrão utilizado para Web Services em que cada serviço é representado por um PortType que descreve as operações com seus parâmetros de entrada, valores retornados, condições de erro, SDE's, etc.



6. A partir da descrição dos serviços SUBS 3 executa operações que enviarão periodicamente informações sobre a topologia do sistema monitorado e as séries temporais representativas de sua operação. Por exemplo a descrição de um equipamento que entrou em operação e as medidas relacionadas em um determinado período.



O GridFTP é um protocolo de alta performance otimizado para transmissão de grandes massas de dados em alta velocidade. O protocolo é baseado no popular FTP, *File Transfer Protocol*, que foi estendido para atender às necessidades das aplicações baseadas em computação grid. O componente do Globus denominado RFT, *Reliable File Transfer Service,* é um serviço baseado na OGSA que implementa uma interface que permite o desenvolvimento de aplicações que gerenciam transferências de dados utilizando servidores GridFTP. Um aplicativo desenvolvido com a API do RFT é capaz de iniciar, reiniciar e particionar a transferência de grandes massas de dados entre servidores GridFTP, mantendo múltiplas conexões simultâneas.

Um dos principais aspectos de sistemas históricos é justamente a manipulação de grandes massas de dados de forma eficiente e robusta. A associação desses componentes do Globus Toolkit a um sistema de banco de dados relacional permite a construção uma poderosa ferramenta para armazenamento de dados históricos da

operação, acessível a todos os componentes do grid, implementando de fato um compartilhamento de espaço em disco. Os dados podem ser enviados em canal alternativo em períodos maiores, sem a necessidade de introduzirmos excesso de informação nos canais de comunicação em tempo real. Outra vantagem é que estando independente do canal de tempo real são eliminados os problemas de perda de dados devido a possíveis interrupções nesse serviço. A informação é historiada a partir de sua fonte, as medições são enviadas diretamente por quem as produz em pacotes por períodos, aumentando a precisão, e a robustez do sistema.

3.3.2 Serviços de processamento remoto

Uma outra forma de compartilhamento de recursos computacionais que pode ser implantada em um grid de supervisão e controle, é feita através da inserção no grid de componentes com grande capacidade de processamento, e disponibilizar o acesso a esses recursos aos demais componentes. No exemplo proposto o componente SUPER, representa um supercomputador dotado de vários aplicativos avançados de apoio à operação de sistemas elétricos. Sobre esses aplicativos é construída uma camada que implementa um serviço responsável por disponibilizar o acesso às aplicações pelos demais componentes. O acesso às aplicações pode ser feito de várias formas conforme as características da aplicação acessada. Em acesso mais simples, a aplicação estaria disponível através de uma operação na qual são passados parâmetros com informações a serem processadas e ao final do processamento são retornados os resultados e encerrada a sessão. Em outro tipo de acesso mais complexo seria criada uma instância da aplicação que permanece ativa indefinidamente aguardando solicitações de processamento.

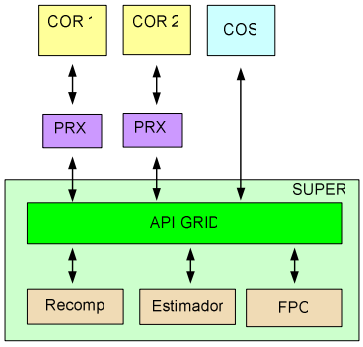


Figura 3-7: Compartilhamento de aplicativos

A seguir analisaremos passo a passo o estabelecimento e uma relação de prestação serviços de acesso a recursos computacionais entre um centro regional e o supercomputador, onde o primeiro utiliza a capacidade de processamento do segundo para solucionar seus problemas. Para fins ilustrativos, suponha que SUPER disponibiliza um software para recomposição de sistemas que é acionado por COR1 toda vez que é detectado um problema na área elétrica monitorada. Uma vez acionado, o programa executa um algoritmo que prepara uma lista de instruções para execução de comandos operativos que resolverão o problema e a envia para o centro de controle para que as ações sejam executadas. Nesse exemplo também serão omitidos os procedimentos de autenticação e autorização que já foram descritos no tópico de estabelecimento de ligações dinâmicas.

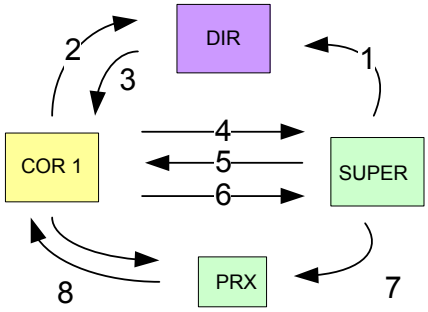
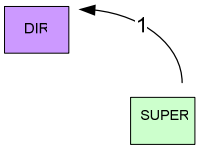
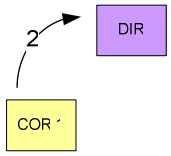


Figura 3-8: Acessando aplicações remotas

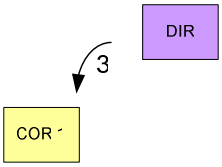
1. Ao ser ativado o componente SUPER publica em DIR seus SDE's descrevendo as aplicações que disponibiliza aos outros componentes do grid de supervisão e controle. Entre elas encontra-se o software de recomposição de sistemas elétricos.



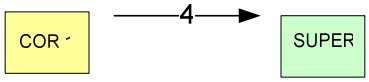
2. Ao entrar em operação o centro de regional, COR 1, efetua uma pesquisa em DIR, buscando um aplicativo de apoio à recomposição de sistemas elétricos.



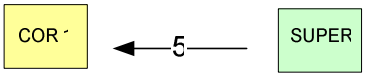
3. Em resposta à solicitação de COR 1, DIR analisa os SDE's publicados pelos demais componentes do grid e encontra o registro de SUPER, que anuncia uma aplicação que atende à COR1. DIR envia o GSH de SUPER para COR 1.



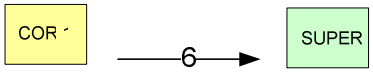
4. De posse da informação de quem fornece o serviço através do GSH. O centro regional agora precisa saber como utilizar os serviços de recomposição disponíveis em SUPER. Para isso é feito o primeiro acesso à SUPER solicitando a descrição detalhada de seus serviços.



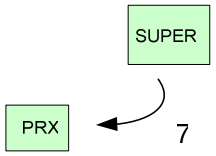
5. A resposta à solicitação é feita através da GSR do componente, que vem sob a forma de um documento GWSDL que contém a lista dos softwares disponíveis e como acessá-los. Dentro do documento é definida uma interface que descreve como acessar o aplicativo de recomposição de sistemas.



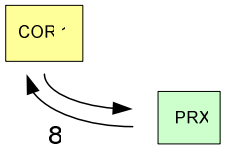
6. O centro de controle de posse do GSR de SUPER executa uma operação solicitando o acesso ao software de recomposição.



7. A resposta de SUPER é criar uma instância do programa de recomposição para atender às requisições de COR 1. Ao ser criada a instância, são carregados todos os dados de não tempo real, necessários para a execução do algoritmo de recomposição. A instância criada é especializada para aquela determinada área elétrica.



8. Uma vez criada a instância do programa de recomposição toda vez que ocorrer uma falta no sistema elétrico monitorado, são enviadas informações de tempo real sobre o problema. Essas informações são processadas pelo programa gerando uma lista de ações para solucionar a falta que é enviada para o centro de controle.



O procedimento descrito não deve ser considerado um modelo e sim como um exemplo de implementação de compartilhamento de recursos. Os procedimentos e mecanismos utilizados podem variar muito dependendo da natureza da aplicação compartilhada.

3.4 Sistema legado

Uma preocupação pertinente a todo processo de evolução tecnológica é a integração com o legado tecnológico em operação no presente. Na área de supervisão e controle de sistemas e elétricos o legado pode ser traduzido em UTR's e CLP's do século passado com 20 até trinta anos de idade utilizando protocolos de comunicação ultrapassados. Também podem fazer parte do sistema legado centros de controle com sistemas SCADA incompatíveis com a computação grid ou que não exista o interesse em investir no porte. O melhor caminho para a integração com este legado é através da implementação dos protocolos de comunicação de dados baseados em computação grid. Como vimos o Globus XIO é uma API presente no Globus Toolkit que permite o desenvolvimento de componentes que implementem protocolos para transmissão e recepção de dados. A API é construída sobre primitivas básicas do tipo *open/close/read/write* e fornece bibliotecas para o desenvolvimento de *drivers* de transporte e conversão de dados modulares e que são organizados em pilha, conforme figura abaixo.

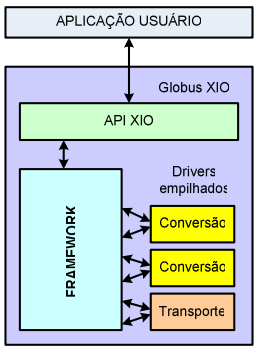


Figura 3-9: Arquitetura Globus XIO

3.5 Protótipo de grid de supervisão e controle

No próximo capitulo serão descritos os procedimentos adotados para o desenvolvimento de um protótipo de um grid de supervisão e controle em tempo real, baseado no exemplo de nossa empresa fictícia definida nesse capítulo. Sobre esse protótipo serão feitas experiências nas quais serão simulados os processos de estabelecimentos de ligações dinâmicas, armazenamento de dados históricos e compartilhamento de aplicativos.

4 - Protótipo de comunidade de supervisão e controle

Nesse capitulo é apresentado o protótipo de grid de supervisão e controle desenvolvido no LASC, Laboratório Avançado de Supervisão e Controle, localizado no CEPEL, Centro de Pesquisas de Energia Elétrica, que gentilmente permitiu o uso de suas instalações para o desenvolvimento desse trabalho.

O objetivo do protótipo é avaliar o comportamento dos mecanismos da computação grid, quando utilizados em aplicações de supervisão e controle em tempo real. Para isso foi implantado no LASC um ambiente para desenvolvimento de aplicações grid baseado no Globus Toolkit versão 3.2.1, onde foram desenvolvidas aplicações que simulam os participantes do grid de supervisão e controle. A empresa fictícia descrita no capitulo anterior foi utilizada como referência para a realização das simulações.

Os componentes foram classificados conforme sua função no grid de supervisão e controle, e para cada tipo de componente foram desenvolvidas uma ou mais aplicações em Java, que implementam um serviço integrado ao contêiner de serviços do Globus ou uma aplicação Java, cliente, que acessa esses serviços. Os componentes desenvolvidos foram alocados pelas estações de trabalho da rede de forma que atendam aos requisitos dos experimentos propostos.

Dentro da rede do LASC uma estação foi escolhida com servidor de infra-estrutura, que armazena todos os arquivos de instalação utilizados na construção do protótipo, e também tem configurados os serviços de segurança e pesquisa de recursos no Globus toolkit.

Os ensaios realizados procuram simular os procedimentos descritos no capitulo 3 para o estabelecimento de ligações dinâmicas e compartilhamento de recursos onde serão avaliados quesitos como a flexibilidade de reconfiguração, tempo de execução de procedimentos e performance na manipulação de grandes massas

O capítulo está dividido em quatro tópicos. No primeiro, é feita a especificação do ambiente de desenvolvimento de aplicações grid, que descreve com detalhes a configuração da rede computacional, softwares instalados e contas de trabalho, diretórios, etc. No segundo tópico são apresentados os componentes do grid de supervisão e controle detalhando a implementação de aplicativos simuladores de suas

funcionalidades e sua integração com o Globus Toolkit. O terceiro tópico descreve a implantação do protótipo na rede do LASC, nele são descritos os procedimentos de instalação e configuração dos componentes do grid de supervisão e controle da empresa fictícia apresentada no capítulo 3 pelas estações da rede do LASC. Os ensaios realizados sobre o protótipo estão descritos no quarto tópico onde os procedimentos para execução dos testes estão detalhados passo a passo e também são apresentados os resultados esperados.

4.1 Especificação do ambiente computacional

Esse tópico descreve o ambiente de testes implantado no LASC onde foram realizados os ensaios relatados esse trabalho.

4.1.1 Rede de computadores

O ambiente para implementação do grid de supervisão e controle está instalado sobre uma rede computacional com 4 PC's e um servidor SUN, conforme é mostrado na Figura 4-1 e descrita na Tabela 4-1.

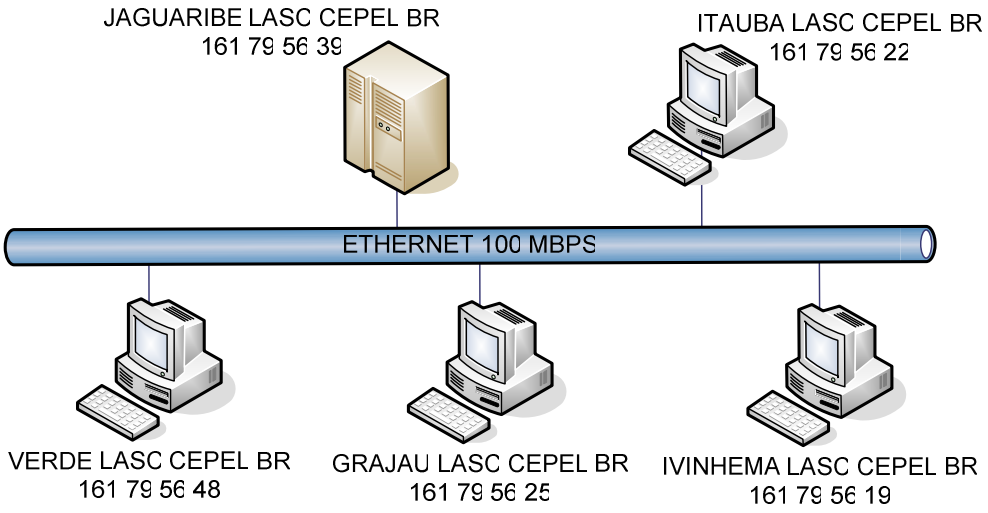


Figura 4-1: Rede computacional

Estação	CPU	Memória (MB)	Disco (GB)	Sistema Operacional
ITAUBA	Pentium IV 1.6 GHz	1000	40	Red Hat EL 3.0
GRAJAU	Pentium IV 1.6 GHz	1000	40	Red Hat EL 3.0
IVINHEMA	Pentium IV 1.6GHz	1000	120 + 40	Red Hat EL 3.0
JAGUARIBE*	Sparc Enterprise 3500	512	70	Solaris 9
VERDE	Pentium III 800 MHz	256	20	Red Hat EL 3.0

* servidor de infra-estrutura

Tabela 4-1 Descrição do hardware e sistemas operacionais

4.1.2 Softwares instalados

Em todas as estações foram feitas instalações típicas do respectivo sistema operacional. Nas estações PC o único cuidado durante a instalação do Linux foi verificar se a opção de instalação das ferramentas de desenvolvimento (gcc, glibc, etc.) está selecionada.

Na estação SPARC foi necessário substituir vários utilitários do Solaris 9, que foram trocados pelas versões descritas na Tabela 4-2.

Utilitário	Versão	Arquivo
AUTOCONF	2.59	autoconf-2.59-sol9-sparc-local.gz
AUTMAKE	1.8b	automake-1.8b-sol9-sparc-local.gz
BIND	9.2.3	bind-9.2.3-sol9-sparc-local.gz
BISON	1.875d	bison-1.875d-sol9-sparc-local.gz
DB	4.2.52	db-4.2.52.NC-sol9-sparc-local.gz
GCC	3.4.1	gcc-3.4.1-sol9-sparc-local.gz
GDBM	1.8.3	gdbm-1.8.3-sol9-sparc-local.gz
LIBICONV	1.8	libiconv-1.8-sol9-sparc-local.gz
M4	1.4.1	m4-1.4.1-sol9-sparc-local.gz
MAKE	3.8.0	make-3.80-sol9-sparc-local.gz
PERL	5.8.5	perl-5.8.5-sol9-sparc-local.gz
TAR	1.14	tar-1.14-sol9-sparc-local.gz
TOP	3.5	top-3.5-sol9-sparc-local.gz

Tabela 4-2 Pacotes substituídos no Solaris 9

Após a instalação dos sistemas operacionais nas respectivas estações e configuração da rede, foi instalada uma série de ferramentas, entre elas o Globus Toolkit, para montar o ambiente de desenvolvimento de aplicações para computação grid.

A seguir é mostrada uma lista com os softwares instalados em todas as estações , especificando a versão e uma breve descrição.

4.1.2.1 Globus Toolkit versão 3.2.1

Kit de ferramentas e aplicativos que implementa a infra-estrutura de serviços para desenvolvimento de aplicações baseadas em computação grid. As instruções para instalação, junto com toda a documentação são encontradas no site do Global Grid Fórum [5]. O pacote foi instalado no diretório /usr/globus/gt321 em todas as máquinas da rede que é referenciado pela variável de ambiente $GLOBUS_LOCATION nas contas de trabalho.

4.1.2.2 Java 2 Standard Edition SDK – Versão 1.4.2_04

Kit de desenvolvimento de aplicações Java. As instruções para instalação, junto com toda a documentação são encontradas em [43]. O pacote foi instalado no diretório /usr/java/j2sdk1.4.2_04 em todas as máquinas da rede que é referenciado pela variável de ambiente $JAVA_HOME nas contas de trabalho. A instalação desta ferramenta é pré-requisito para a instalação do Globus Toolkit.

4.1.2.3 Apache Ant – Versão 1.6.2

Ferramenta de construção de sistemas baseada em Java configurável através de arquivos XML. As instruções para instalação, junto com toda a documentação são encontradas em [41]. O pacote foi instalado no diretório /usr/ant/apache-ant-1.6.2 em todas as máquinas da rede que é referenciado pela variável de ambiente $ANT_HOME nas contas de trabalho. A instalação desta ferramenta é pré-requisito para a instalação do Globus Toolkit.

4.1.2.4 JUnit– Versão 3.8.1

É uma ferramenta feita em Java para preparação e execução de baterias de testes em aplicações. Para instalar basta copiar o arquivo junit.jar contido no arquivo de distribuição para o diretório /usr/ant/apache-ant-1.6.2/lib. O arquivo de distribuição e toda documentação é encontrada no site do Junit [42]. A instalação desta ferramenta é pré-requisito para a instalação do Globus Toolkit.

4.1.2.5 Postgresql - Versão 7.4.5

Banco de dados relacional que é utilizado por alguns componentes do Globus Toolkit e utilizado para implementação do banco de dados histórico da operação. As instruções para instalação, junto com toda a documentação são encontradas em [44]. O pacote foi instalado no diretório /var/lib/pgsql. A instalação desta ferramenta é pré-requisito para a instalação do Globus Toolkit.

4.1.2.6 Eclipse – Versão 3.0.1

O Eclipse é uma ferramenta gráfica de desenvolvimento de sistemas com licença publica de uso gratuito. Além das ferramentas de edição e depuração foi desenvolvido um *plug-in* que permite a geração automática de projetos de serviços integrados ao Globus Toolkit.

4.1.3 Servidor de infra-estrutura

Como já foi mencionada anteriormente, uma estação da rede foi selecionada como servidor de infra-estrutura. Além dos softwares mencionados no tópico anterior foram instalados os seguintes serviços:

4.1.3.1 Globus SimpleCA

Componente do Globus Toolkit para de emissão de certificados digitais utilizados pelo GSI, *Globus Security Infrastructure*, nas operações de validação e criptografia. Foi utilizado para gerar certificados para as estações e usuários do grid de supervisão e controle. A configuração do certificador gerou o arquivo **globus_simple_ca_ec60c7ba_setup-0.17.tar.gz** que foi copiado para as demais estações do grid e instalado no processo de configuração do GSI que é descrito em [5], [19] e [46].

4.1.3.2 CAS - Community Authorization Service

Responsável pela autenticação e autorização no grid, o CAS é configurado através de tabelas em um banco de dados Postgresql. Os certificados gerados pelo Globus SimpleCA são relacionados com os recursos disponíveis do grid através de uma tabela de ações permitidas, estabelecendo um relacionamento n:n. O serviço é acessado pelos componentes do grid para verificar se o cliente requisitando é realmente que se diz ser, e se ele está autorizado a acessar esse serviço. O CAS foi utilizado para implementar o componente AUT do grid de supervisão e controle da empresa fictícia.

4.1.3.3 MDS –Monitoring & Discovery System

O MDS é o componente do Globus Toolkit que implementa um servidor de diretório que permite que os demais componentes do grid publiquem seus recursos através dos SDE's, *Service Data Elements*, e que esses recursos sejam pesquisados pelos demais componentes do grid. No protótipo do grid de supervisão e controle da empresa fictícia o componente DIR é implementado através do MDS.

O servidor de infra-estrutura também armazena todos os arquivos de instalação utilizados para implementar o ambiente e as versões oficiais dos programas fonte e scripts desenvolvidos

4.1.4 Usuário Globus

Em cada estação foram criados um grupo e uma conta local para o usuário globus. A conta é responsável pela administração da instalação do Globus Toolkit e também é utilizada como conta de trabalho para o desenvolvimento de aplicações. A raiz da conta globus é localizada em /**export**/**home**/**globus**, e o processador de comandos default é /**bin**/**tcsh**.

4.1.4.1 Script de configuração

O ambiente da conta globus é configurado através do arquivo **.cshrc** localizado na raiz da conta. Nele são definidas as variáveis de ambiente e executados scripts de configuração do Globus Toolkit. Abaixo segue listagem do arquivo

```
#/bin/tcsh
#
#  Script de inicialização para conta de
#  desenvolvimento de protótipo grid
#  supervisão e controle
#
#  Indica path para instalação do gt321
setenv GLOBUS_LOCATION /usr/globus/gt321

#  Indica path para instalação do java 2 sdk
setenv JAVA_HOME  /usr/java/j2sdk1.4.2_04/

#  Indica path para instalação do Ant
setenv ANT_HOME  /usr/ant/apache-ant-1.6.2

#  Indica path para instalação do Junit para
#  ambiente de execução Java
setenv CLASSPATH "/usr/junit/junit3.8.1/junit.jar"

#  Atualiza path para executáveis dos pacotes
#  instalados
setenv PATH "$ANT_HOME/bin:$JAVA_HOME/bin:$PATH"

# Executa scripts de configuração de ambiente do
# Globus Toolkit 3.2.1
source $GLOBUS_LOCATION/etc/globus-user-env.csh
source $GLOBUS_LOCATION/etc/globus-devel-env.csh
```

Caso seja necessário criar uma segunda conta para desenvolvimento de aplicativos grid, será necessário que essa conta tenha permissão de escrita no diretório de instalação do Globus e quando iniciada execute os procedimentos listados acima.

4.1.4.2 Árvore de diretórios

A figura 4.3 mostra como estão organizados os arquivos da conta globus no servidor de infra-estrutura.

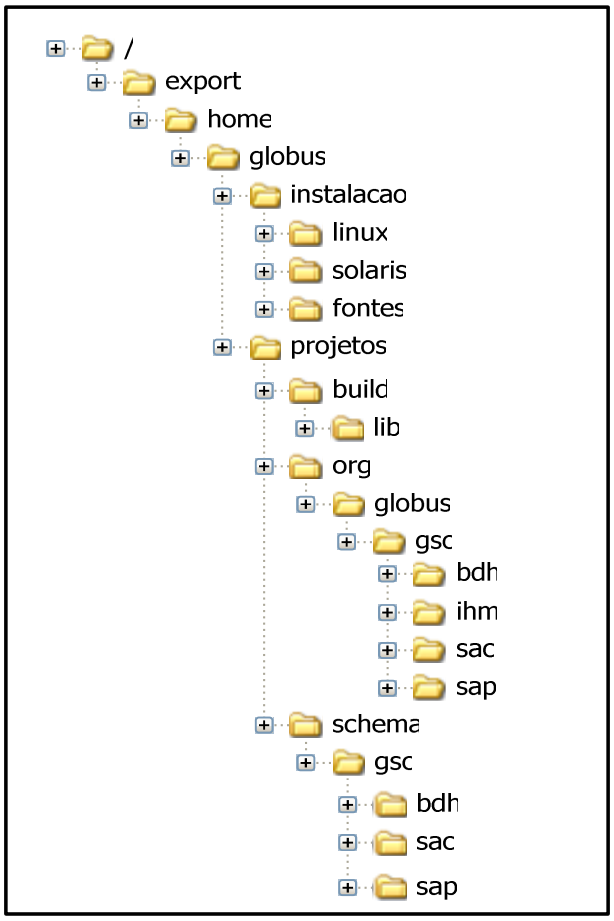


Figura 4-2: Arvore de diretórios da conta globus

O diretório **instalacao,** contém os arquivos utilizados na implantação do ambiente de desenvolvimento no LASC, organizados por sistema operacional. O diretório de projetos contém os programas desenvolvidos para simulação dos componentes do grid de supervisão e controle. Na raiz do diretório de **projetos** ficam os arquivos comuns a todos os componentes, que são utilizados no processo de construção O diretório **build** armazena os arquivos gerados durante a construção do sistema. O subdiretório **org** e seus descendentes armazenam os programas fontes em Java com a implementação do serviço e o arquivo **server-deploy.sdd** utilizados na implantação dos serviços no

contêiner do Globus Toolkit. O subdiretório **schema** contém os arquivos do tipo GWSDL com a definição das interfaces dos serviços disponibilizados pelo componente.

4.1.5 Scripts e arquivos de configuração

A figura a seguir mostra arquivos de configuração e scripts utilizados na construção dos componentes do grid de supervisão e controle. O conteúdo desses arquivos é listado no apêndice ao final deste trabalho.

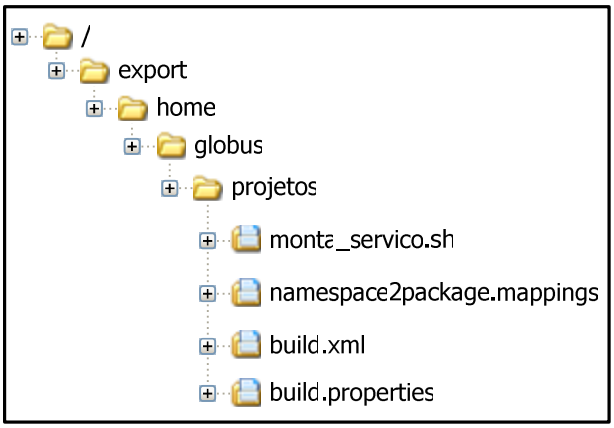


Figura 4-3:Scripts e arquivos de configuração

build.properties

Arquivo contendo a localização da instalação do Globus Toolkit.

build.xml

Arquivo XML utilizado pela ferramenta Ant de forma genérica para construção de componentes do grid de supervisão e controle. É configurado para construir os arquivos do tipo GAR que serão transferidos para o contêiner de serviços do Globus no processo de implantação do serviço.

monta_serviço.sh

Script que executa a construção do serviço. Ao ser ativado recebe como parâmetros o caminho completo para o pacote Java de implementação do serviço e o nome do arquivo GWSDL contendo a definição da interface do serviço.

namespace2package.mappings

Faz o mapeamento entre os endereços dos serviços no formato URI e os pacotes do serviço.

4.2 Componentes do grid de supervisão e controle.

Na implementação do protótipo, os componentes foram classificados conforme sua funcionalidade no grid de supervisão e controle da empresa fictícia, e para cada grupo de atividade foi gerado um aplicativo ou configurada uma ferramenta do Globus que simule seu funcionamento. Foram classificados seis grupos de componentes:

- Serviço de aquisição e controle, as usinas e subestações.
- Centros de controle, centros regionais e COS.
- Servidor de histórico da operação, o componente BDH.
- Servidor de aplicação, o componente SUPER.
- Serviço de registro e pesquisa, o componente DIR.
- Serviço de autenticação e autorização, o componente. AUT

A seguir serão detalhadas as aplicações desenvolvidas para cada grupo. Conforme a características de cada grupo foram desenvolvidas aplicações integradas ao contêiner de serviços do Globus ou uma aplicação Java cliente desses serviços. Os componentes, DIR e AUT, foram implementados através da configuração dos componentes CAS e MDS do Globus Toolkit, portanto não foi necessário o desenvolvimento de programas. Todos os arquivos fonte utilizados na implementação do protótipo, encontram-se listados no anexo A desse documento.

4.2.1 Centro de controle

Provê uma interface homem-máquina para a monitoração de sistemas elétricos. São centros de decisão e, portanto grandes consumidores de informação e serviços. Na empresa fictícia proposta os centros de controle são representados pelos componentes COS, COR1, COR2, COR3. Para simulação destes componentes foi desenvolvido um aplicativo em Java que implementa um cliente dos serviços providos pelos demais

componentes do grid de supervisão e controle. Suas funcionalidades são ativadas através da interface gráfica mostrada na Figura 4-4.

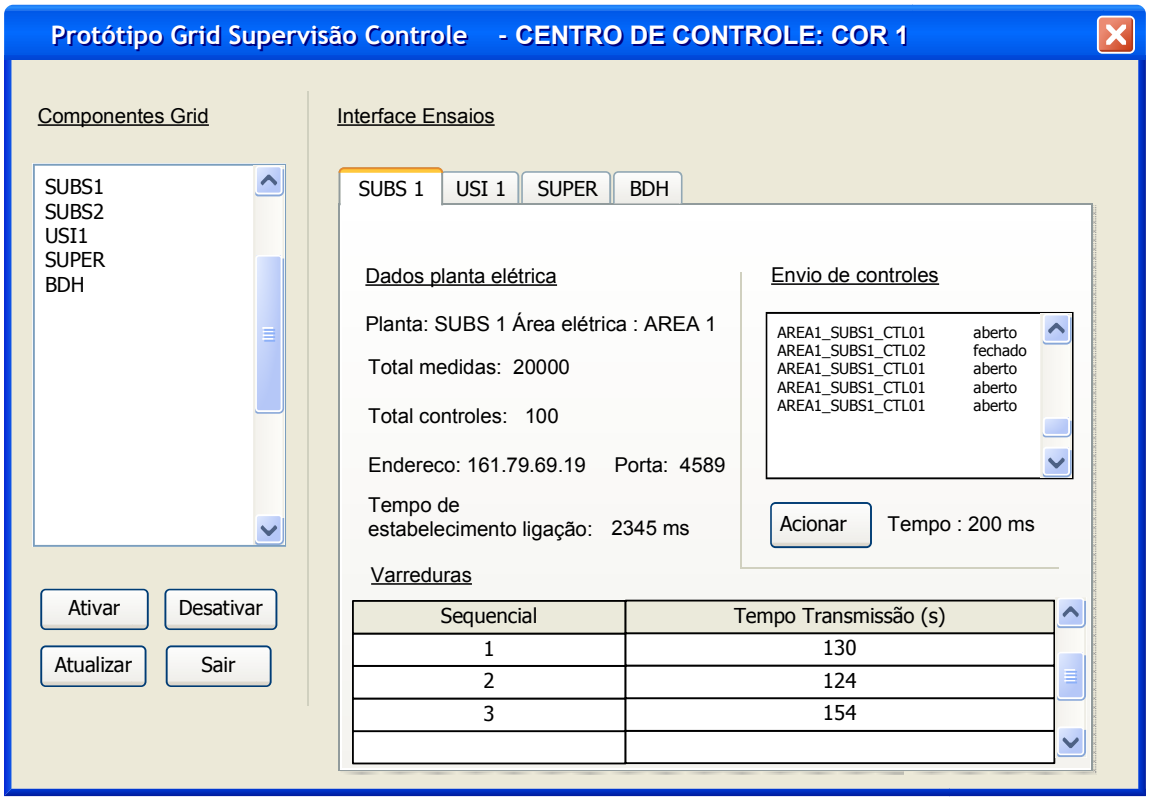


Figura 4-4 Interface centro de supervisão e controle

A interface é dividida em duas partes, a primeira sob o título, **Componentes do Grid,** é utilizada para pesquisar por recursos no grid de supervisão e controle. A segunda parte com o título **Interface Ensaios** é reservada para execução dos ensaios sobre os componentes selecionados.

4.2.1.1 Pesquisa por componentes ativos

O painel esquerdo da interface é destinado a mostrar os componentes disponíveis no grid de supervisão e controle. Ao se clicar no botão **Atualizar** o programa faz uma pesquisa no servidor MDS cujo endereço já vem pré-configurado na interface. Durante a pesquisa são procuradas instâncias dos SDE Identificador que contém o nome e o tipo do componente. A partir dessas informações a lista de componentes da interface é preenchida com os nomes dos componentes ativos no grid de supervisão e controle.

Ao selecionar um componente da lista e clicar no botão **Ativar** é criada no painel direito uma interface que permite a interação com o componente selecionado e que é utilizada na realização de ensaios que serão descritos mais adiante neste capítulo. As interfaces para ensaios estão organizadas em um controle do tipo fichário, o que permite a conexão simultânea do centro de controle com vários componentes de diversos tipos. O botão desativar apaga do painel de ensaios a interface relativa ao componente selecionado na lista. E o botão sair finaliza o programa.

4.2.1.2 Interface para ensaios sobre ligações de supervisão e controle

Ao ser ativada a conexão com um componente do tipo serviço de aquisição e controle, é criada no painel direito uma interface igual à mostrada na Figura 4-5.

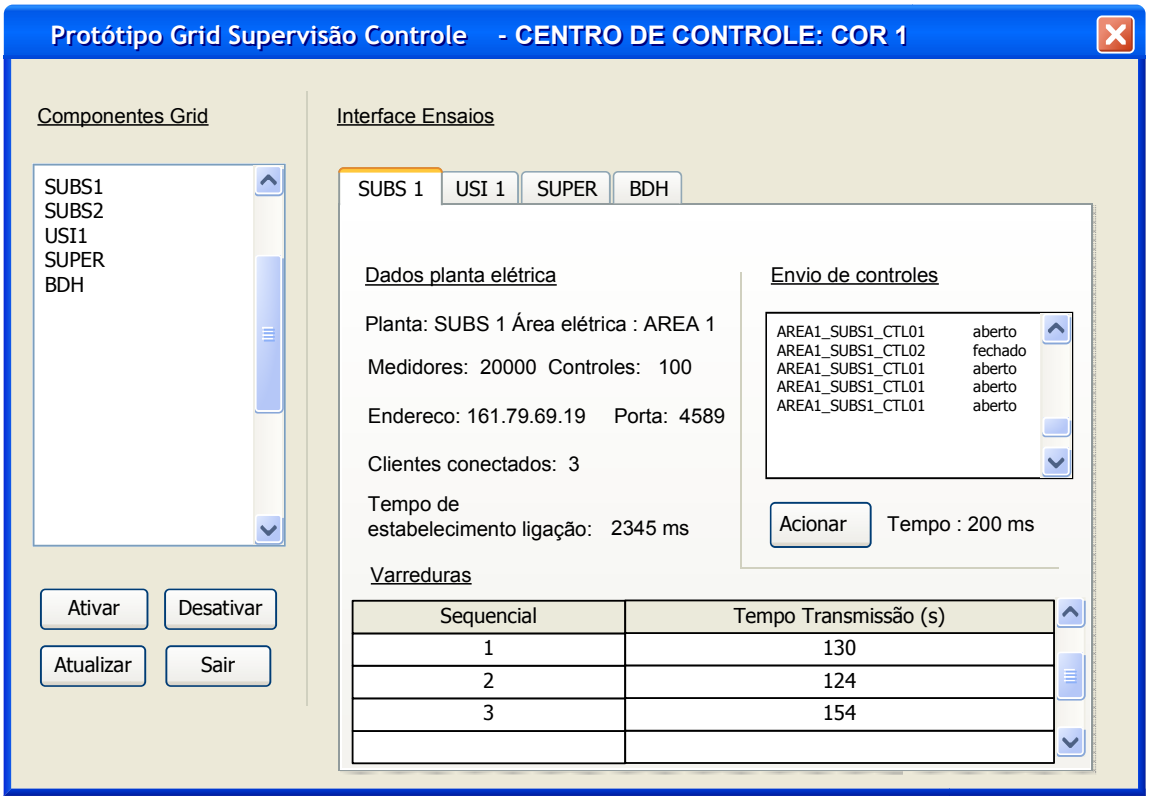


Figura 4-5: Interface ensaios sobre ligações de supervisão e controle

A interface é composta por três partes. Na primeira são mostrados os dados da planta elétrica monitorada onde podem ser visualizadas as informações sobre o nome da planta elétrica, número de pontos de medição, número de pontos de controle , o

endereço e porta utilizados no estabelecimento da ligação de supervisão, número de centros de controle conectados e tempo para gasto no estabelecimento da conexão.

O segundo painel é utilizado para o envio de comandos operativos para a planta elétrica, que pode ser feito selecionando o ponto na lista de controles operativos e clicando no botão **Acionar.** O controle é executado através da operação **executarControle** da interface de serviços do componente, que é mostrada na Figura 4-10, e é enviado através de protocolo SOAP para o componente que simula a planta elétrica, que simplesmente troca o estado operativo do ponto e retorna o novo estado para o centro de controle. Após a execução do comando a lista na interface gráfica com os pontos de controle é atualizada mostrando o novo estado do ponto selecionado. O tempo de execução do comando, ou seja, o tempo entre o clicar do botão e o recebimento da resposta ao chamado da operação confirmando a execução do comando é mostrado à direita do botão.

Ao ser ativada uma conexão, paralelamente é estabelecida uma ligação via *socket* com o servidor de ligações implementado no componente serviço de aquisição e controle. Através dessa ligação, são enviadas varreduras periódicas dos medidores supervisionados. Na parte inferior é apresentada uma tabela contendo um histórico das varreduras recebidas pelo centro de controle e o atraso entre a geração da informação e sua chegada ao centro de controle, o que corresponde no tempo gasto no envio das medidas.

A interface permite conexões simultâneas com várias plantas elétricas que são apresentadas como fichas no controle tipo fichário na interface de ensaios. A tabela **Varreduras** é atualizada automaticamente ao receber medições do serviço de aquisição e controle monitorado. Durante a ativação é solicitado ao serviço de pesquisa e indexação que seja notificado da entrada em operação de plantas elétricas pertencentes à área elétrica monitorada pelo centro.

4.2.1.3 Interface para ensaios sobre histórico da operação

A interface para ensaios sobre o histórico da operação é ativada quando selecionamos BDH no painel de seleção de componentes e clicamos no botão **Ativar**. A interface é mostrada na Figura 4-6.

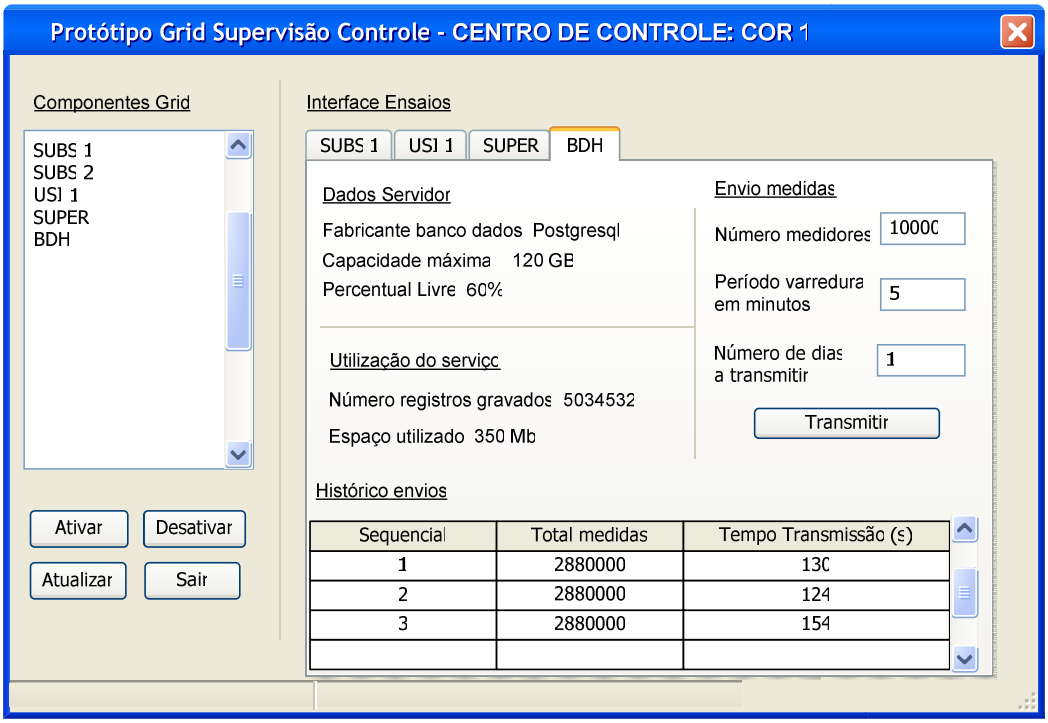


Figura 4-6: Interface de ensaios sobre histórico da operação

No painel esquerdo são mostradas estatísticas sobre o provedor de armazenamento de dados históricos onde são descritas a capacidade total de armazenamento do servidor e o percentual do espaço total livre. Também são mostradas estatísticas do uso do servidor pelo cliente onde são apresentados o número de registros de medidas gravados e o espaço em disco ocupado pelo cliente.

A partir das informações do número de medidores, período de varredura e total de dias a transmitir é gerado um arquivo texto no formato CSV com medidas relativas aos parâmetros configurados, que é enviado ao servidor de histórico via protocolo GridFTP e seu conteúdo é armazenado em um banco de dados relacional PostgreSql. A Figura 4-7 mostra o formato do arquivo que é enviado via protocolo GridFTP. O arquivo é

gravado em um diretório local à estação onde é executado o servidor GridFtp que para simplicidade é a mesma estação onde é executado o componente centro de controle.

```
Nome ,          Data/hora,                  Valor
COR1_MED_0001 , 2005-01-23 00:00:00 -02:00, 234.45
COR1_MED_0001 , 2005-01-23 00:05:00 -02:00, 233.45
COR1_MED_0001 , 2005-01-23 00:10:00 -02:00, 234.75
COR1_MED_0001 , 2005-01-23 00:15:00 -02:00, 235.45
COR1_MED_0001 , 2005-01-23 00:20:00 -02:00, 232.45
COR1_MED_0001 , 2005-01-23 00:25:00 -02:00, 234.46
                                          .
                                          .
                                          .
COR1_MED_0001 , 2005-01-23 23:50:00 -02:00, 232.45
COR1_MED_0001 , 2005-01-23 23:55:00 -02:00, 234.46
COR1_MED_0002 , 2005-01-23 00:00:00 -02:00, 234.45
COR1_MED_0002 , 2005-01-23 00:05:00 -02:00, 233.45
COR1_MED_0002 , 2005-01-23 00:10:00 -02:00, 234.75
                                          .
                                          .
                                          .
COR1_MED_9999 , 2005-01-23 23:55:00 -02:00, 234.46
```

Figura 4-7: Arquivo de transferência de dados históricos

A estatística do envio de medidas é mostrada na tabela na parte inferior da interface onde são mostrados os tempos gastos no envio e armazenamento dos dados.

4.2.1.4 Interface para ensaios sobre servidor de aplicações

A ativação da interface é feita através da seleção de um componente do tipo servidor de aplicações e clicando no botão **Ativar**. A interface é criada no painel direito do console de ensaios conforme é mostrada na Figura 4-8.

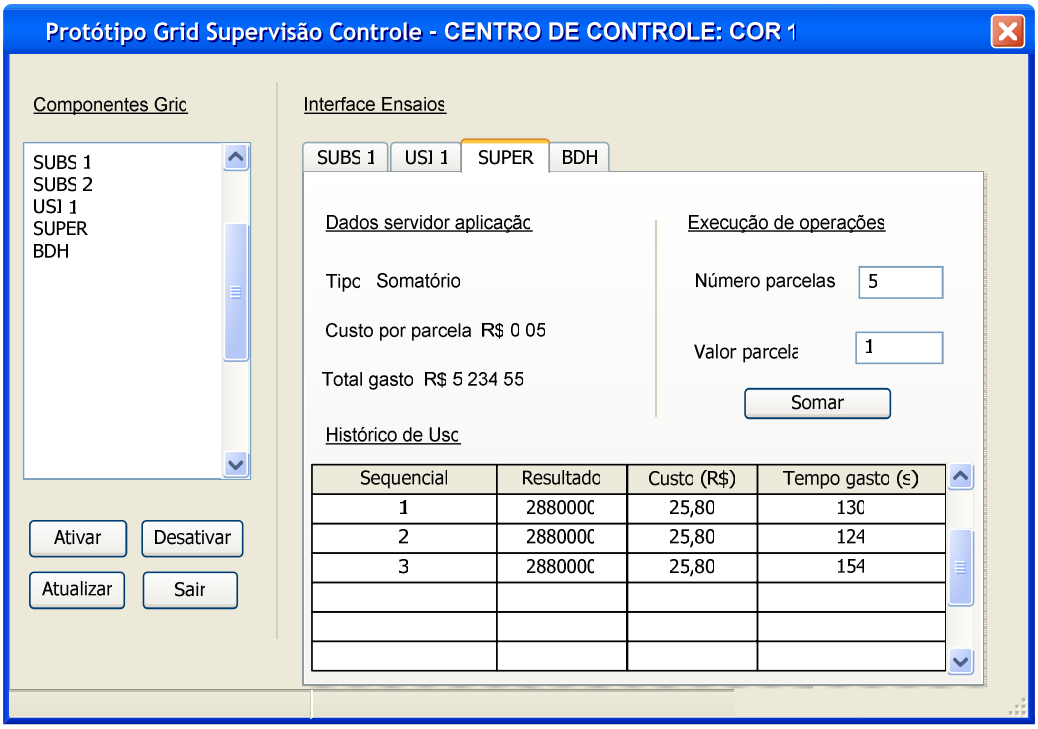


Figura 4-8: Interface de ensaio sobre servidor de aplicação

Para testar o serviço de processamento remoto de cálculos foi desenvolvido um componente que simula um supercomputador que realiza a operação de soma das parcelas de um vetor que lhe é passado como parâmetro de entrada. A interface mostrada acima permite a geração e envio de vetores, cujo número de parcelas é definido no campo **NumeroParcelas** No campo **Valor Parcela** é definido o único valor para todas as parcelas do vetor. A título de ilustração foi estabelecido um custo de utilização do serviço baseado no número de parcelas somadas. A interface mostra no painel de dados do servidor de aplicação, além da informação do tipo de cálculo provido o custo por parcela e o total gasto pelo cliente.

4.2.1.5 Diretório do projeto

A Figura 4-9 mostra a estrutura de diretórios utilizada no desenvolvimento do componente. Os principais arquivos são descritos a seguir e seus conteúdos estão listados no apêndice ao final desse trabalho :

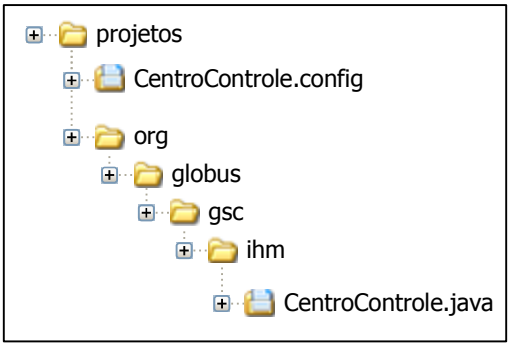


Figura 4-9: Árvore de diretórios do projeto do componente centro de controle

CentroControle.config

Arquivo texto contendo parâmetros iniciais para execução do programa.

CentroControle.java

Arquivo do programa fonte Java com a implementação do centro de controle.

4.2.1.6 Procedimentos de construção do componente

Abaixo são mostrados os procedimentos de construção e ativação do componente

```
> javac  -classpath ./build/classes/:$CLASSPATH \
         org/globus/gsc/ihm/CentroControle.java

> java -classpath ./build/classes/:$CLASSPATH \
         org.globus.gsc.ihm.CentroControle
```

4.2.2 Serviço de aquisição e controle

Componente responsável pelo sensoriamento do sistema elétrico. Através de dispositivos de medição, disponibiliza para a comunidade informações em tempo real

sobre a operação do sistema. E permitem também a manipulação do sistema através telecomandos. Na empresa fictícia proposta, esse tipo de componente é representado pelas usinas e subestações.

Sua implementação é feita através de uma aplicação Java integrada ao contêiner de serviços do Globus Toolkit, descrita na Figura 4-10 que implementa um servidor de ligações de supervisão. Através das ligações são enviadas periodicamente varreduras das medidas monitoradas, que na simulação são geradas aleatoriamente.

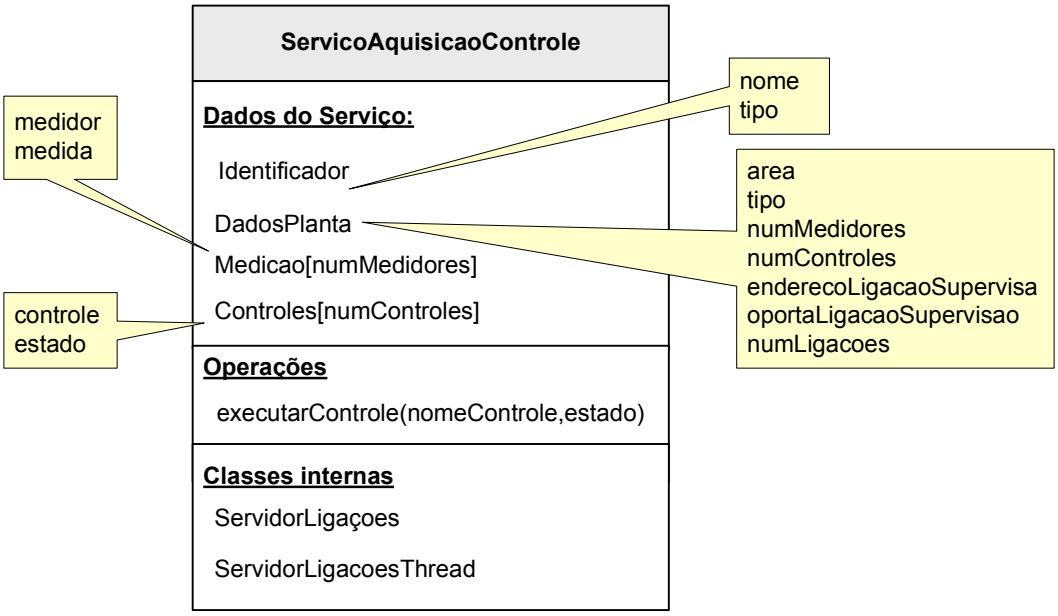


Figura 4-10: Descrição do sistema de aquisição e controle

4.2.2.1 Dados do serviço

Os dados do serviço estão organizados em 4 SDE's que contêm informações sobre a identificação do componente, as características da planta elétrica simulada e sobre o seu estado operativo.

Identificador

Esse SDE identifica o componente e é utilizado nas pesquisas por recursos disponíveis no grid de supervisão e controle. Todos os componentes do grid possuem esse SDE, que é publicado no serviço de pesquisa e indexação durante a ativação, onde fica disponível para pesquisa pelos demais componentes.

A cardinalidade do SDE é 1 e seus atributos estão descritos na Tabela 4-3.

Nome	Tipo	Descrição
nome	String	Nome componente.
tipo	String	Tipo do componente (sac,sap,bdh)

Tabela 4-3: Atributos SDE Identificador

DadosPlanta

Os atributos deste SDE guardam as características da planta elétrica simulada e informações para estabelecimento de ligações de supervisão e controle. Durante ativação do serviço os dados são inicializados a partir do um arquivo texto com as características da planta elétrica simulada. A cardinalidade do SDE é 1 e seus atributos estão descritos na Tabela 4-4.

Nome	Tipo	Descrição
tipo	String	Tipo da planta (usina ou subestação).
area	String	Área elétrica.
numMedidores	Inteiro	Número de medidores.
numControles	Inteiro	Número de pontos de controle.
endereçoLigacaoSupervisao	String	Endereço IP para estabelecimento de ligação de supervisão.
portaLigacaoSupervisao	Inteiro	Porta de serviços utilizada para estabelecimento de ligação de supervisão.
numLigacoes	Inteiro	Numero de ligações estabelecidas com centros de controle.

Tabela 4-4: Atributos da SDE DadosPlanta

Medição

Este SDE é responsável por manter a lista dos pontos de medição da planta elétrica e os valores medidos na última varredura. A ligação de supervisão prepara o buffer descrito na Figura 4-11 a partir dos valores do atributo **medidas** e transmite aos clientes a cada varredura.

A cardinalidade deste SDE é definida pelo atributo **numMedidores** definido na SDE **DadosPlanta** e seus atributos estão descritos na Tabela 4-5.

Nome	Tipo	Descrição
medidor	String	Nome do ponto de medição.
medida	Real	Últimos valores medidos.

Tabela 4-5:Atributos da SDE Medição

A lista é gerada durante o processo de inicialização do serviço, a partir do algoritmo de formação de identificadores de pontos, que concatena a sigla MED com o identificador da planta, a área, e um serial iniciado em 1 até **numMedidores** que é mostrado no trecho de código abaixo. Os valores medidos são preenchidos periodicamente com valores aleatórios.

```
for (int i=0; i< dadosPanta.getNumMedidores(); i++)
{
    medicao[i].setMedidor("MED_"+ dadosPanta.getNome()+ "_" +
                          dadosPanta.getArea()+ Integer.toString(i+1);
    medicao[i].setMedida = (0.0);
}
```

Controle

A SDE controle é utilizada na simulação de controles operativos, que são acionados através da operação **executaControle**. Ao ser acionada, a operação troca o valor do atributo estado simulando um controle operativo de abertura ou fechamento de, por exemplo, um disjuntor. A cardinalidade deste SDE é definida pelo atributo **numControles** declarado na SDE **DadosPanta** e seus atributos estão descritos na Tabela 4-6.

Nome	Tipo	Descrição
controle	String	Nome do ponto de controle.
estado	Booleano	Estado atual do ponto de controle.

Tabela 4-6: Atributos da SDE Controle

A lista é gerada durante o processo de inicialização do serviço, a partir do algoritmo de formação de identificadores de pontos, que concatena a sigla CTR com o identificador da planta, a área, e um serial iniciado em 1 até **numControles** conforme é mostrado no trecho de código abaixo. Os estados são inicializados com o valor **false** podem ser alterados através da operação. **executaControle** que será descrita mais adiante.

```
......
for (int i=0; i< dadosPanta.getNumControles(); i++)
{
    controle[i].setControle("CTR_"+ dadosPanta.getNome()+ "_" +
                            dadosPanta.getArea()+Integer.toString(i+1);
    controle [i]. setEstado = (0.0);
}
....
```

4.2.2.2 Operações do serviço

O componente disponibiliza somente uma operação tendo em vista que a ligação de envio de medições é feita através de um protocolo à parte e que para o seu estabelecimento é suficiente o acesso aos SDE do serviço. A operação disponibilizada permite o envio de controles operativos através de chamadas à função descrita abaixo.

```
public boolean executaControle(String nomeControle, boolean novoEstado)
                                                    throws RemoteException
{
   int i;
   for ( i=0; i< dadosPlanta.getNumControles() &&
        !controle[i].getNomeControle().equalsIgnoreCase(nomeControle);
           ;i++);
   if (i >= dadosPlanta.getNumControles())
   {
      throw new RemoteExcetpion("Ponto de controle não encontrado.");
   }
   controle[i].estado=novoEstado;
   return  Controle[i].estado;
}
```

A operação recebe como parâmetros o nome do ponto de controle a ser acionado e o novo estado desejado. No caso de sucesso na execução da operação, a função retorna o estado do controle que deve ser igual ao solicitado via parâmetro. No caso de ocorrer algum erro durante a execução do comando, como por exemplo o nome do controle selecionado não existir, é disparada uma exceção do tipo **RemoteException** que deve ser tratada pela aplicação responsável pela execução da operação.

4.2.2.3 Servidor de Ligações

O sistema traz embutida a implementação de um servidor ligações de supervisão baseado em *socket* que executa um protocolo de envio periódico de medidas que é ativado durante a inicialização do serviço. O servidor é acessado através do endereço IP e porta, disponíveis nos dados do serviço pelos atributos **enderecoLigacaoSupervisao** e **portaLigacaoSupervisao** que fazem parte da SDE **DadosPlanta**. Uma aplicação cliente utiliza o endereço e porta para criar uma ligação de supervisão onde é enviado periodicamente um buffer contendo uma varredura das medidas da planta elétrica simulada. O buffer contém a data e hora da varredura e um valor para cada medidor, ordenados conforme o SDE **Medição**.

```
<data/hora em ms>#<valor medida 1>#<valor medida 2># ...
```

Figura 4-11: Buffer de transmissão de valores medidos

O servidor de ligações de supervisão é implementado através das classes Java **ServidorLigacoes** e **ServidorLigacoesThread** internas ao provedor de serviços. A primeira é instanciada durante a ativação do serviço e passa a aguardar requisições de conexão através da porta definida pelo atributo **portaLigacaoSupervisao**. No momento em que uma aplicação cliente do grid solicita a criação de uma ligação, é criada uma instância de **ServidorLigacoesThread** que passa a transmitir varreduras dos valores dos medidos, via *socket*, no formato especificado na Figura 4-11.

4.2.2.4 Diretório do projeto

A Figura 4-12 mostra a estrutura de diretórios utilizada no desenvolvimento do componente.

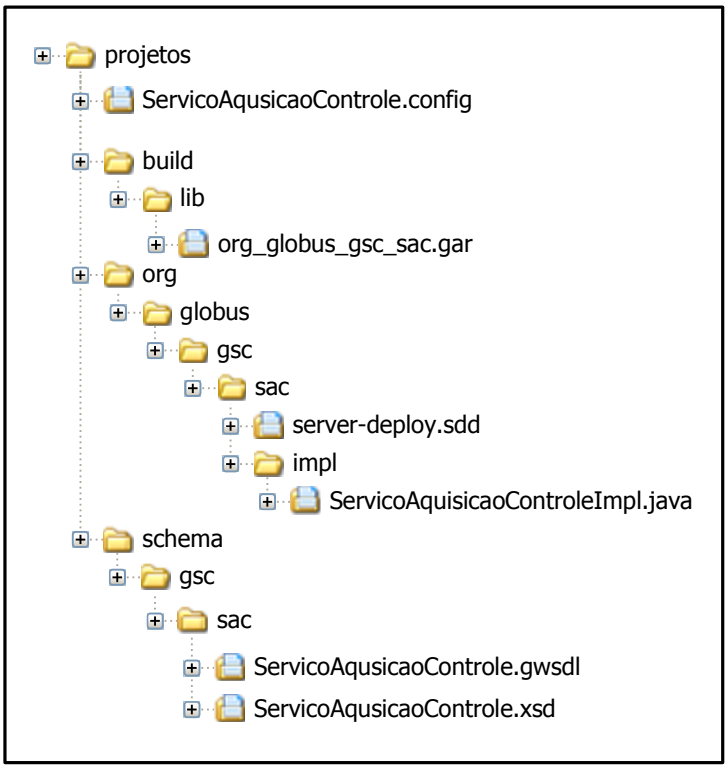


Figura 4-12: Árvore de diretórios do projeto serviço de aquisição e controle

Os principais arquivos são descritos a seguir e seus conteúdos estão listados no apêndice ao final desse trabalho :

server-deploy.sdd

Esse arquivo contém as configurações necessárias para integração do serviço ao contêiner de serviços do Globus Toolkit.

ServicoAquisicaoControle.config

Nesse arquivo são configuradas as características do serviço de aquisição e controle. Nele estão descritas o nome da planta, área elétrica, número de pontos de medição e de controle, endereço IP e a porta para estabelecimento de ligações de supervisão. O arquivo é lido durante a inicialização do serviço e armazenado no SDE **DadosPanta**. Abaixo é mostrado o exemplo de um arquivo de configuração do serviço de aquisição e controle. O caractere # no inicio da linha indica comentário.

```
# Planta : Área  : Tipo : Medidores : Controles : Host :        : Porta
  SUBS6  : AREA2 : subs : 1000      : 15        : 161.79.56.19 : 4568
```

org_globus_sac.gar

Arquivo compactado gerado pelo procedimento de construção do serviço que contém todos os arquivos utilizados na integração do serviço no contêiner do Globus Toolkit. É o produto final do processo de construção do sistema.

ServicoAquisicaoControle.gwsdl

Arquivo XML onde é descrita a interface do serviço. Nela são descritas as operações e os dados de serviço. Esse arquivo é utilizado no protocolo SOAP durante o procedimento de estabelecimento de um relacionamento cliente-servidor entre dois componentes para informar ao cliente como acessar os recursos disponíveis no provedor de serviços.

ServicoAquisicaoControle.xsd

Arquivo XML contém a definição dos SDE do serviço, é incluído pelo arquivo de definição da interface **ServicoAquisicaoControle.gwsdl** no processo de construção do sistema.

ServicoAquisicaoControleImpl.java

Arquivo Java onde está codificada a classe que implementa as operações do serviço. As classes ServidorLigacoes e ServidorLigacoesThread que simulam as ligações de supervisão dinâmicas estão codificadas neste arquivo.

4.2.2.5 Procedimentos de construção do serviço

O processo de construção de um provedor de serviços grid é executado através ferramenta ANT que gera a partir dos arquivos de definição da interface de serviços, de configuração do serviço no contêiner do globus e do código de implementação do serviço um arquivo com extensão GAR,pronto para ser instalado no contêiner do globus. A figura abaixo ilustra o processo.

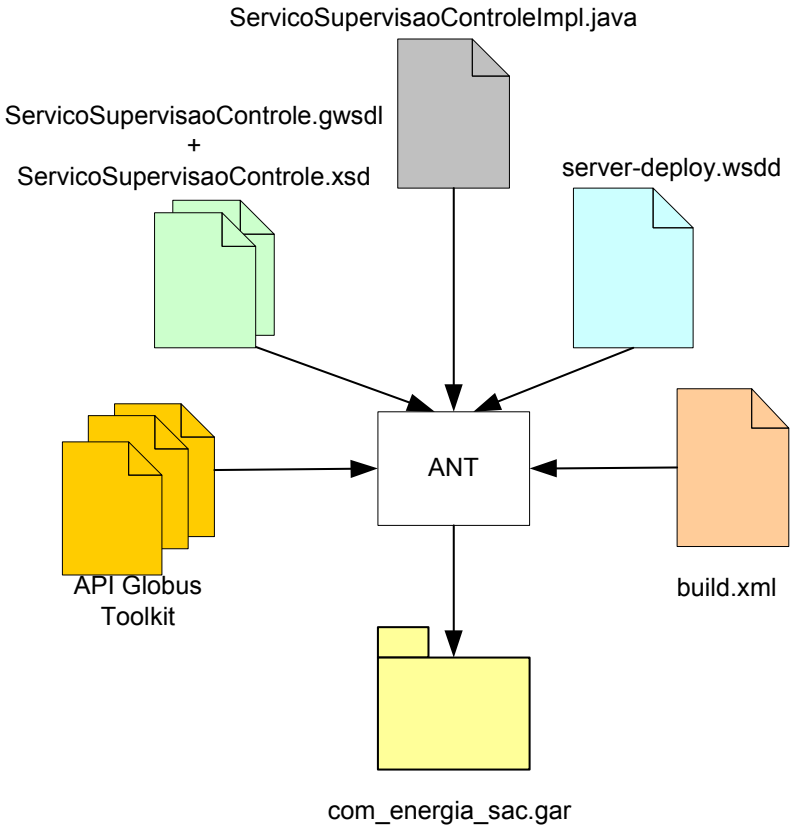


Figura 4.7: Processo de geração do arquivo GAR

O processo de construção do serviço e integração ao contêiner Globus é descrito passo a passo a seguir.

```
> cd ~/projeto

> monta_servico org/globus/gsc/sac
schema/sac/ServicoAquisicaoControle.gar

> cd $GLOBUS_LOCATION

> ant deploy -Dgar.name=~/projetos/buid/lib/org_globus_sac.gar
```

A ativação do serviço é feita através dos seguintes comandos:

```
> cd $GLOBUS_LOCATION

> globus-start-conteiner  -p 8080
```

Caso tudo tenha sido feito corretamente, deverão aparecer várias linhas mostrando os serviços disponíveis no contêiner e entre elas uma referente ao serviço de aquisição e controle semelhante à mostrada abaixo.

```
http://161.79.56.19:8080/ogsa/services/gsc/sac
```

Nesse ponto o serviço está pronto para ser acessado pelos demais componentes do grid.

4.2.3 Provedor de armazenamento de dados históricos

Este componente é responsável por manter um histórico da operação do sistema elétrico. Ele funciona como um grande armazém de dados que disponibiliza aos outros componentes, espaço para armazenamento do histórico da operação. O processo de armazenamento de dados históricos baseia-se no envio de arquivos contendo medidas relativas a um determinado período através do protocolo Gridftp e após o envio o conteúdo do arquivo é armazenado em um banco de dados Postgresql.

Sua implementação é feita através de uma aplicação Java integrada ao contêiner de serviços do Globus Toolkit, descrita na Figura 4-13.

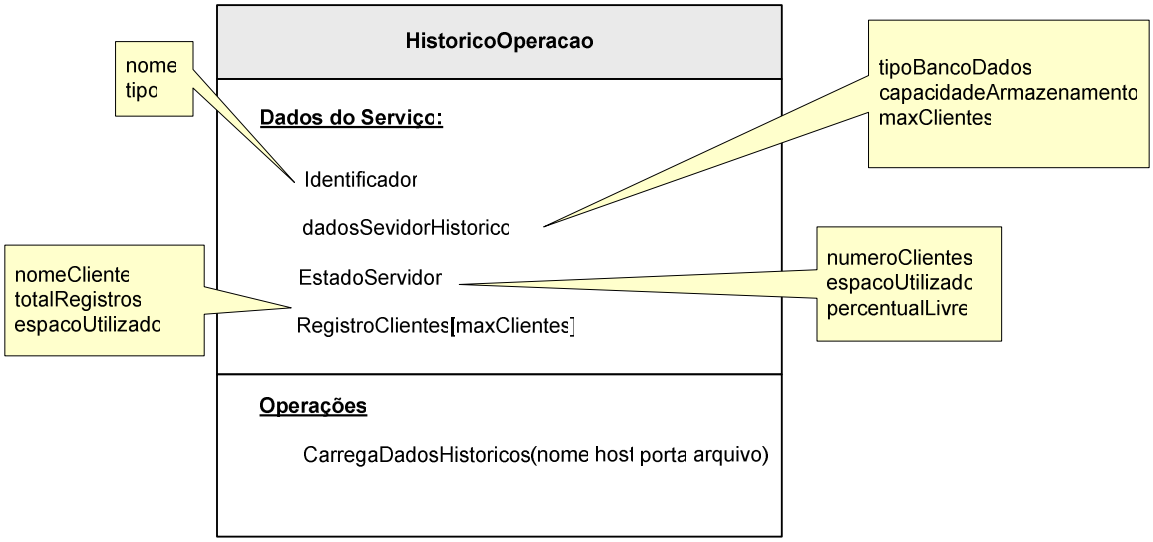


Figura 4-13: Interface do serviço histórico operaçào

4.2.3.1 Dados do serviço

Os dados do serviço estão organizados em 4 SDE's que contêm informações sobre as características do servidor, seu estado atual e lista de estatísticas do serviço por cliente. Devido às características de persistência de um banco de dados histórico, os SDE's desse componente são armazenados em tabelas do banco de dados que são lidas no processo de inicialização e atualizadas durante a operação do serviço.

Identificador

Esse SDE é o identificador do componente, e é utilizado nas pesquisas por recursos disponíveis no grid de supervisão e controle. Todos os componentes do grid possuem esse SDE que é publicado no componente DIR durante a ativação do serviço para pesquisa pelos demais componentes. A cardinalidade do SDE é 1 e seus atributos estão descritos na Tabela 4-7.

Nome	Tipo	Descrição
nome	String	Nome componente.
tipo	String	Tipo do componente (sac,sap,bdh)

Tabela 4-7: Atributos SDE Identificador

DadosServidorHistórico

Os atributos deste SDE guardam as características do serviço de histórico simulado. Durante ativação do serviço os dados são inicializados a partir da tabela **DADOS_SERVIDOR_HISTORICO** localizada no banco de dados Postgresql. A cardinalidade do SDE é 1 e seus atributos estão descritos na **Tabela 4-8.**

Nome	Tipo	Descrição
tipoBancoDados	String	Nome fabricante do banco de dados.
capacidadeArmazenamento	Inteiro	Espaço total em disco em GB.
maxClientes	Inteiro	Número máximo de clientes.

Tabela 4-8: Atributos SDE DadosServidorHistórico

EstadoServidor

Este SDE mantém informações sobre o estado do servidor, sua persistência é feita através da tabela ESTADO_SERVIDOR do banco de dados Postgresql que é lida durante o processo de inicialização e atualizada durante a operação do serviço. A cardinalidade deste SDE é 1 e seus atributos estão descritos na Tabela 4-9.

Nome	Tipo	Descrição
numeroClientes	String	Nome do ponto de medição.
espacoUtilizado	Real	Espaço em disco ocupado em MB
percentualLivre	Real	Percentual do disco livre

Tabela 4-9:Atributos da SDE EstadoServidor

RegistroClientes

A cardinalidade deste SDE é definida pelo atributo **maxClientes** definido na SDE **IdentificadorServidorHistorico** e seus atributos estão descritos na tabela 4.5. As informações são persistidas no banco de dados através da tabela REGISTRO_CLIENTES que é um espelho deste SDE.

Nome	Tipo	Descrição
nomeCliente	String	Nome do cliente
totalRegistros	Inteiro	Total de registros armazenados pelo cliente
espacoUtilizado	Inteiro	Espaço em disco ocupado pelo histórico do cliente

Tabela 4-10: Atributos da SDE RegistroClientes

4.2.3.2 Operações do serviço

O componente oferece uma operação que permite o envio e gravação, em um banco de dados PostgreSql, dos dados históricos que é descrita abaixo.

```
public void carregaDadosHistoricos(String nome, String host,
                                   String porta, String arquivo)
                                               throws RemoteException
```

O envio dos dados é feito através do envio do arquivo especificado no parâmetro **arquivo** através do servidor GridFtp localizado na estação indicada pelos parâmetros **host** e **porta**. Durante a execução é verificado se o cliente especificado pelo parâmetro **nome** já possui dados armazenados no banco de dados. Caso não tenha, o serviço cria uma tabela no banco para gravação das medidas e atualiza o SDE **RegistroClientes** com o novo usuário que identificado pelo parâmetro **nome**. Detalhes sobre a implementação do banco de dados histórico serão apresentados no tópico a seguir.

4.2.3.3 Banco de dados histórico

O banco de dados histórico consiste de, para cada cliente atendido, manter uma tabela com o nome igual ao do cliente que é descrita abaixo. A cada chamada da operação **carregaDadosHistoricos** são inseridos nesta tabela os registros contidos no arquivo de transporte definido na Figura 4-7.

Nome	Tipo	Descrição
identificadorPonto	String	Identificador do ponto de medicao
dataHora	Data	Data hora da medida
Valor medido	Real	Valor medido

Tabela 4-11: Tabela com histórico de medidas

Além das tabelas históricas, existem também as tabelas de armazenamento dos conteúdos das SDE's do componente. A estrutura destas tabelas é idêntica à estrutura das respectivas SDE's e são criadas através do script **HistoricoOperacao.sql** que é listado no apêndice no final deste trabalho.

4.2.3.4 Diretório do projeto

A Figura 4-14 mostra a estrutura de diretórios utilizada no desenvolvimento do componente. Os principais arquivos estão destacados e são descritos a seguir e seus conteúdos estão listados no apêndice ao final desse trabalho :

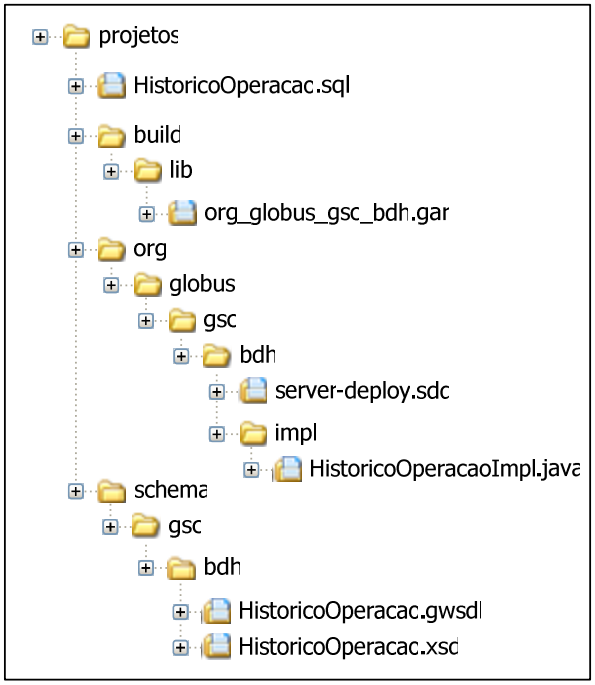


Figura 4-14: Árvore de diretórios do projeto serviço de histórico da operação

HistoricoOperacao.sql

Script SQL com os comandos para criação das tabelas do serviço no banco de dados Postgresql.

server-deploy.sdd

Esse arquivo contém as configurações necessárias para integrar no serviço ao contêiner de serviços do Globus Toolkit.

org_globus_bdh.gar

Arquivo compactado gerado pelo procedimento de construção que contém todos os arquivos utilizados na integração do serviço ao contêiner do Globus Toolkit. É o produto final do procedimento de construção.

HistoricoOperacao.gwsdl

Arquivo XML onde é descrita a interface do serviço. Nela são descritas as operações e os dados de serviço. Esse arquivo é utilizado no protocolo SOAP durante o procedimento de estabelecimento de um relacionamento cliente-servidor entre dois componentes para informar ao cliente como acessar os recursos disponíveis no provedor de serviços.

HistoricoOperacao.xsd

Arquivo XML contém a definição dos SDE do serviço, é incluído pelo arquivo de definição da interface **HistoricoOperacao.gwsdl** no processo de construção do sistema.

HistoricoOperacaoImpl.java

Arquivo Java onde está codificada a classe que implementa as operações do serviço.

4.2.3.5 Procedimentos de construção do serviço

O processo de construção do serviço e integração ao contêiner Globus é descrito passo a passo a seguir.

```
> cd ~/projeto
>createdb historico_operacao
>psql -d histórico_operacao -f HistoricoOperacao.sql
>monta_servico org/globus/gsc/bdh  schema/bdh/HistoricoOperacao.gar
>cd $GLOBUS_LOCATION
>ant deploy -Dgar.name=~\projetos\HistoricoOperacao \
                \buid\lib\org_globus_bdh.gar
```

A ativação do serviço é feita através dos seguintes comandos:

```
>cd $GLOBUS_LOCATION

>globus-start-conteiner  -p 8080
```

Caso tudo tenha sido feito corretamente deverão aparecer várias linhas mostrando os serviços disponíveis no contêiner, e entre elas uma referente ao serviço conforme é destacado abaixo.

```
http://161.79.56.19:8080/ogsa/services/gsc/bdh
```

Nesse ponto o serviço está pronto para ser acessado pelos demais componentes do grid.

4.2.4 Provedor de aplicações de suporte à operação.

Este componente simula um supercomputador que provê serviços de cálculos avançados de apoio à operação do sistema. Para a simulação foi desenvolvida uma aplicação Java integrada ao contêiner de serviços do Globus Toolkit. O serviço desenvolvido disponibiliza uma operação que soma todos os membros de um vetor de valores que é passado como parâmetro e retorna o resultado da soma. A interface do serviço é mostrada na figura abaixo.

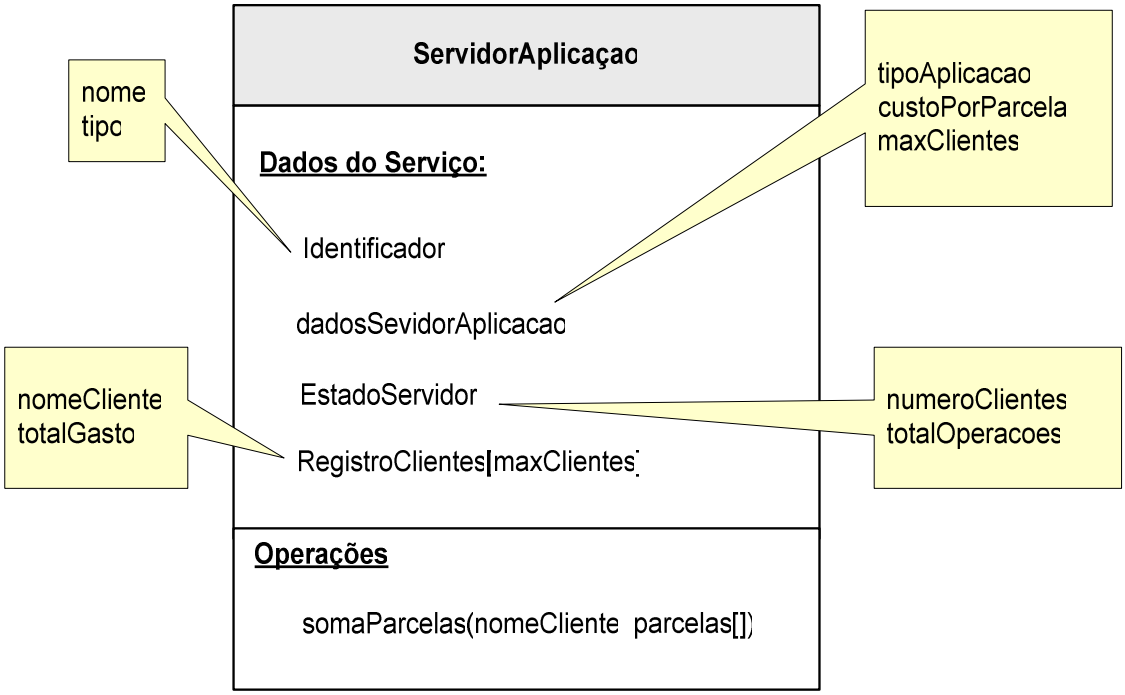


Figura 4-15: Interface do servidor de aplicações

A título de ilustração foram associados custos ao processamento do cálculo que são cobrados conforme o número de parcelas do vetor. O valor a ser cobrado de cada cliente fica armazenado em um SDE específico para esse fim.

4.2.4.1 Dados do serviço

Os dados do serviço estão organizados em 4 SDE's que contêm informações sobre as características do servidor, seu estado atual e lista de estatísticas por cliente do serviço.

Identificador

Esse SDE é o identificador do componente, e é utilizado nas pesquisas por recursos disponíveis no grid de supervisão e controle. Todos os componentes do grid possuem esse SDE que é publicado no componente DIR durante a ativação do serviço ficando disponível para pesquisa pelos demais componentes. A cardinalidade do SDE é 1 e seus atributos estão descritos na Tabela 4-12.

Nome	Tipo	Descrição
nome	String	Nome componente.
tipo	String	Tipo do componente (sac,sap,bdh)

Tabela 4-12: Atributos SDE identificador do servidor de aplicações

DadosServidorAplicacao

Os atributos deste SDE guardam as características do servidor de aplicações simulado. Durante ativação do serviço os dados são inicializados a partir do arquivo de configuração **ServidorAplicaçao.config.** A cardinalidade do SDE é 1 e seus atributos estão descritos na Tabela 4-13**.**

Nome	Tipo	Descrição
tipoAplicacao	String	Tipo de calculo efetuado.
custoPorParcela	Inteiro	Custo de utilização por parcela
maxClientes	Inteiro	Número máximo de clientes.

Tabela 4-13: Atributos SDE IdentificadorServidorAplicacao

EstadoServidor

Este SDE mantém informações sobre o estado do servidor, é zerado o processo de inicialização e atualizado durante a operação do serviço.A cardinalidade deste SDE é 1 e seus atributos estão descritos na Tabela 4-14

Nome	Tipo	Descrição
numeroClientes	Inteiro	Numero de clientes utilizando o serviço.
totalOperacoes	Inteiro	Total de operações executadas

Tabela 4-14:Atributos da SDE EstadoServidor

RegistroClientes

A cardinalidade deste SDE é definida pelo atributo **maxClientes** definido na SDE **DadosServidorAplicacao** e seus atributos estão descritos na Tabela 4-15.

Nome	Tipo	Descrição
nomeCliente	String	Nome do cliente
totalGasto	Inteiro	Valor a ser cobrado do cliente

Tabela 4-15: Atributos da SDE RegistroClientes

4.2.4.2 Operações do serviço

A interface do componente provê uma operação que soma as parcelas do vetor enviado como parâmetro e retorna o resultado. A chamada da operação é descrita abaixo e mostra dois parâmetros. O primeiro identifica o cliente do serviço e o segundo é um vetor cujas parcelas devem ser somadas.

```
public double somaParcelas(String nome, double parcelas[]) throws
                                                RemoteException
```

4.2.4.3 Diretório do projeto

A Figura 4-14 mostra a estrutura de diretórios utilizada no desenvolvimento do componente.

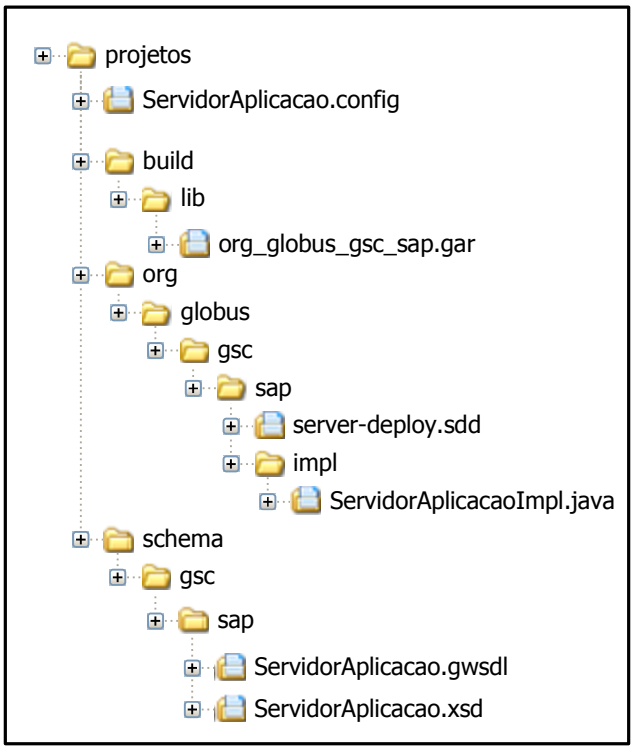


Figura 4-16: Árvore de diretórios do projeto servidor de aplicação

Os principais arquivos estão destacados e são descritos a seguir e seus conteúdos estão listados no apêndice ao final desse trabalho :

server-deploy.sdd

Esse arquivo contém as configurações necessárias para integrar no serviço ao contêiner de serviços do Globus Toolkit.

ServicoAquisicaoControle.config

Nesse arquivo são configuradas as características do servidor de aplicação. O arquivo é lido durante a inicialização do serviço e armazenado no SDE

IdentificadorServidorAplicacao.

Abaixo é mostrado o exemplo de um arquivo de configuração. O caractere # no inicio da linha indica comentário.

```
# Nome : TipoAplicacao.: CustoPorParcela : MaxClientes
  Super : Somatorio : subs :         0.5 :        15
```

org_globus_sap.gar

Arquivo compactado gerado pelo procedimento de construção do serviço, que contém todos os arquivos utilizados na integração ao contêiner do Globus Toolkit. É o produto final do processo de construção do sistema.

ServidorAplicacao.gwsdl

Arquivo XML onde é descrita a interface do serviço. Nela são descritas as operações e os dados de serviço. Esse arquivo é utilizado no protocolo SOAP durante o procedimento de estabelecimento de um relacionamento cliente-servidor entre dois componentes para informar ao cliente como acessar os recursos disponíveis no provedor de serviços.

ServidorAplicacao.xsd

Arquivo XML contém a definição dos SDE do serviço, é incluído pelo arquivo de definição da interface **ServidorAplicacao.gwsdl** no processo de construção do sistema.

ServidorAplicacao Impl.java

Arquivo Java onde está codificada a classe que implementa as operações do serviço.

4.2.4.4 Procedimentos de construção do serviço

O processo de construção do serviço e integração ao contêiner Globus é descrito passo a passo a seguir.

```
> cd ~/projeto

> monta_servico org/globus/gsc/sap  schema/sap/ServidorAplicacao.gar

>cd $GLOBUS_LOCATION

> ant deploy \
  -Dgar.name=~\projetos\ServidorAplicacao\buid\lib\org_globus_sap.gar
```

A ativação do serviço é feita através dos seguintes comandos:

```
>cd $GLOBUS_LOCATION

>globus-start-conteiner  -p 8080
```

Caso tudo tenha sido feito corretamente deverão aparecer várias linhas mostrando os serviços disponíveis no contêiner e entre elas uma referente ao serviço conforme é destacado na lista abaixo.

```
http://161.79.56.19:8080/ogsa/services/gsc/sap
```

Nesse ponto o serviço está pronto para ser acessado pelos demais componentes do grid.

4.3 Preparação do ambiente

Para execução dos ensaios, os componentes da empresa fictícia foram distribuídos pela rede do LASC conforme descrito na Figura 4-17.

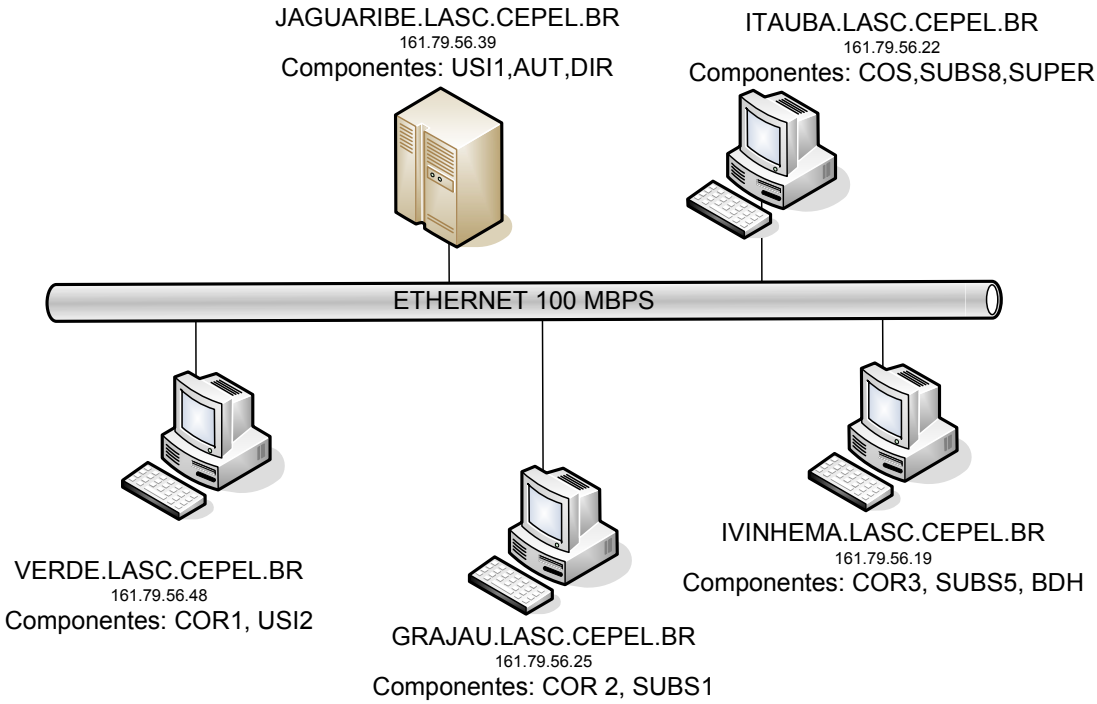


Figura 4-17 Localização dos componentes na rede do LASC

Nesse ponto é considerado que o Globus Toolkit e os demais softwares descritos no primeiro tópico já estejam instalados e configurados em todas as estações. E os componentes descritos no segundo tópico já tenham sido compilados e estão localizados no diretório ~/**projetos** da conta globus na estação **jaguaribe** que faz o papel de servidor de infra-estrutura. O transporte dos componentes para as demais estações é feito copiando o diretório de projetos da estação jaguaribe para a raiz da conta globus das demais estações.

Uma vez copiada a implementação dos componentes cabe agora configurá-los e integrá-los ao contêiner de serviços do Globus Toolkit para que simulem seus papéis no grid de supervisão e controle da empresa fictícia. A seguir são mostrados os passos para configuração de cada estação. Todos os passos serão executados na conta globus.

4.3.1 Estação Jaguaribe

Os componentes AUT e DIR foram implementados a partir dos componentes CAS e MDS do Globus Toolkit. O procedimento de configuração desses serviços é detalhadamente descrito em [5] e [45], e está fora do escopo desse trabalho. Resta o procedimento de configuração do componente USI1 que faz parte do grupo de serviços de aquisição e controle.

4.3.1.1 Configuração de USI 1

Editar o arquivo ~/projetos/ServicosAquisicaoControle.config de forma a ficar igual ao listado abaixo.

```
# Planta : Área  : Tipo  : Medidores : Controles : Host :       : Porta
  USI1   : AREA2 : Usina : 1000      : 15        : 161.79.56.39 : 4568
```

Publicar o serviço no contêiner do Globus Toolkit:

```
> cd $GLOBUS_LOCATION

> ant deploy -Dgar.name=~/projetos/buid/lib/org_globus_sac.gar
```

A ativação do contêiner é feita através dos seguintes comandos:

```
> cd $GLOBUS_LOCATION

> globus-start-conteiner  -p 8080
```

Após a execução do comando de ativação devem aparecer várias linhas mostrando os serviços disponíveis no contêiner e entre elas os serviços especialmente configurados nessa estação destacados pelas linhas em negrito abaixo. O terminal deve ficar preso na aplicação e para desativá-la basta teclar Ctrl-C que o contêiner será desativado.

```
> globus-start-conteiner  -p 8080

[01/25/2005 10:19:34:780 ] org.globus.ogsa.server.ServiceContainer
INFO: Starting SOAP server at: http://161.79.56.39:8080/ogsa/services/
With the following services:

http://161.79.56.39:8080/ogsa/services/core/admin/AdminService
http://161.79.56.39:8080/ogsa/services/core/management/OgsiManagementSe
rvice
http://161.79.56.39:8080/ogsa/services/core/registry/ContainerRegistryS
ervice
                .
http://161.79.56.39:8080/ogsa/services/gsc/sac
                .
http:// 161.79.56.39:8080/ogsa/services/ogsi/HandleResolverService
http://161.79.56.39:8080/ogsa/services/base/cas/CASService
http://161.79.56.39:8080/ogsa/services/base/index/IndexService
                .
```

4.3.2 Estação Verde

Os passos para configuração dos componentes COR1 e USI2 são descritos a seguir.

4.3.2.1 Configuração de COR1

Editar o arquivo ~/projetos/CentroControle.config de forma a ficar igual ao listado abaixo.

```
# Nome : Servidor GridFTP  : Porta GridFtp
  COR1 : 161.79.56.48      : 4568
```

Para ativar a interface execute os comandos:

```
>cd ~/projetos

>java -classpath ./build/classes/:$CLASSPATH \
         org.globus.gsc.ihm.CentroControle &
```

4.3.2.2 Configuração USI2

Editar o arquivo ~/projetos/ServicosAquisicaoControle.config de forma a ficar igual ao listado abaixo.

```
# Planta : Área  : Tipo  : Medidores : Controles : Host          : Porta
  USI2   : AREA2 : Usina : 1500      : 19        : 161.79.56.48 : 4568
```

Publicar o serviço no contêiner do Globus Toolkit:

```
> cd $GLOBUS_LOCATION
> ant deploy -Dgar.name=~/projetos/buid/lib/org_globus_sac.gar
```

A ativação do contêiner é feita através dos seguintes comandos:

```
> cd $GLOBUS_LOCATION
> globus-start-conteiner  -p 8080
```

Após a execução do comando de ativação devem aparecer várias linhas mostrando os serviços disponíveis no contêiner e entre elas os serviços especialmente configurados nessa estação destacados pelas linhas em negrito abaixo. O terminal deve ficar preso na aplicação e para desativá-la basta teclar Ctrl-C que o contêiner será desativado.

```
http://161.79.56.48:8080/ogsa/services/gsc/sac

> globus-start-conteiner  -p 8080

[01/25/2005 10:19:34:780 ] org.globus.ogsa.server.ServiceCon...
INFO: Starting SOAP server at: http://161.79.56.48:8080/....
With the following services:

http://161.79.56.48:8080/ogsa/services/core/admin/AdminService
http://161.79.56.48:8080/ogsa/services/core/management/Og...
                .
http://161.79.56.48:8080/ogsa/services/gsc/sac
                    .
```

4.3.3 Estação Grajaú

Os passos para configuração dos componentes COR2 e SUBS1 são descritos a seguir.

4.3.3.1 Configuração de COR2

Editar o arquivo ~/projetos/CentroControle.config de forma a ficar igual ao listado abaixo.

```
# Nome : Servidor GridFTP  : Porta GridFtp
  COR2 : 161.79.56.25      : 4568
```

Para ativar a interface execute os comandos:

```
>cd ~/projetos

>java –classpath ./build/classes/:$CLASSPATH \
         org.globus.gsc.ihm.CentroControle &
```

4.3.3.2 Configuração SUBS1

Editar o arquivo ~/projetos/ServicosAquisicaoControle.config de forma a ficar igual ao listado abaixo.

```
# Planta: Área : Tipo      : Medidores: Controles :Host          : Porta
  SUBS1 : AREA1: Subestacao: 2300     : 25        :161.79.56.25 : 4568
```

Publicar o serviço no contêiner do Globus Toolkit:

```
> cd $GLOBUS_LOCATION

> ant deploy –Dgar.name=~/projetos/buid/lib/org_globus_sac.gar
```

A ativação do contêiner é feita através dos seguintes comandos:

```
> cd $GLOBUS_LOCATION

> globus-start-conteiner  -p 8080
```

Após a execução do comando de ativação, devem aparecer várias linhas mostrando os serviços disponíveis no contêiner e entre elas os serviços especialmente configurados nessa estação destacados pelas linhas em negrito abaixo. O terminal deve ficar preso na aplicação e para desativá-la basta teclar Ctrl-C que o contêiner será desativado.

```
> globus-start-conteiner  -p 8080

[01/25/2005 10:19:34:780 ] org.globus.ogsa.server.ServiceContainer
INFO: Starting SOAP server at: http://161.79.56.25:8080/ogsa/services/
With the following services:

              .
http://161.79.56.25:8080/ogsa/services/gsc/sac
              .
```

4.3.4 Estação Ivinhema

Os passos para configuração dos componentes BDH, SUBS5 e COR3 são descritos a seguir.

4.3.4.1 Configuração de BDH

Execute os passos a seguir para criar a base de dados histórica e integrar o serviço ao contêiner do Globus.

```
> cd ~/projeto

>createdb histórico_operacao

>psql -d histórico_operacao -f HistoricoOperacao.sql

>cd $GLOBUS_LOCATION

>ant deploy -Dgar.name=~\projetos\HistoricoOperacao \
               \buid\lib\org_globus_bdh.gar
```

4.3.4.2 Configuração de COR3

Editar o arquivo ~/projetos/CentroControle.config de forma a ficar igual ao listado abaixo.

```
# Nome : Servidor GridFTP  : Porta GridFtp
  COR3 : 161.79.56.19      : 4568
```

Para ativar a interface execute os comandos:

```
>cd ~/projetos

>java -classpath ./build/classes/:$CLASSPATH \
         org.globus.gsc.ihm.CentroControle &
```

4.3.4.3 Configuração SUBS5

Editar o arquivo ~/projetos/ServicosAquisicaoControle.config de forma a ficar igual ao listado abaixo.

```
# Planta: Área : Tipo      : Medidores: Controles :Host          : Porta
  SUBS5 : AREA2: Subestacao: 500      : 10        :161.79.56.19 : 4568
```

Publicar o serviço no contêiner do Globus Toolkit:

```
> cd $GLOBUS_LOCATION

> ant deploy -Dgar.name=~/projetos/buid/lib/org_globus_sac.gar
```

A ativação do contêiner é feita através dos seguintes comandos:

```
> cd $GLOBUS_LOCATION

> globus-start-conteiner  -p 8080
```

Após a execução do comando de ativação devem aparecer várias linhas mostrando os serviços disponíveis no contêiner e entre elas os serviços especialmente configurados nessa estação destacados pelas linhas em negrito abaixo. O terminal deve ficar preso na aplicação e para desativá-la basta teclar Ctrl-C que o contêiner será desativado.

```
> globus-start-conteiner  -p 8080

[01/25/2005 10:19:34:780 ] org.globus.ogsa.server.ServiceContainer
INFO: Starting SOAP server at: http://161.79.56.19:8080/ogsa/services/
With the following services:


              .
http://161.79.56.19:8080/ogsa/services/gsc/sac
http://161.79.56.19:8080/ogsa/services/gsc/bdh
              .
```

4.3.5 Estação Itaúba

Os passos para configuração dos componentes SUPER, SUBS8 e COS são descritos a seguir.

4.3.5.1 Configuração de SUPER

Editar o arquivo ~/projetos/ServidorAplicacao.config de forma a ficar igual ao listado abaixo.

```
# Nome  : TipoAplicacao.: CustoPorParcela : MaxClientes
  Super : Somatorio     : 0.5  :          : 15
```

Publicar o serviço no contêiner do Globus Toolkit:

```
> cd $GLOBUS_LOCATION

> ant deploy -Dgar.name=~/projetos/buid/lib/org_globus_sap.gar
```

4.3.5.2 Configuração de COS

Editar o arquivo ~/projetos/CentroControle.config de forma a ficar igual ao listado abaixo.

```
# Nome : Servidor GridFTP  : Porta GridFtp
  COS  : 161.79.56.22      : 4568
```

Para ativar a interface execute os comandos:

```
>cd ~/projetos

>java -classpath ./build/classes/:$CLASSPATH \
         org.globus.gsc.ihm.CentroControle &
```

4.3.5.3 Configuração SUBS8

Editar o arquivo ~/projetos/ServicosAquisicaoControle.config de forma a ficar igual ao listado abaixo.

```
# Planta: Área : Tipo      : Medidores: Controles :Host          : Porta
  SUBS8 : AREA3: Subestacao: 200      : 7         :161.79.56.22 : 4568
```

Publicar o serviço no contêiner do Globus Toolkit:

```
> cd $GLOBUS_LOCATION

> ant deploy -Dgar.name=~/projetos/buid/lib/org_globus_sac.gar
```

A ativação do contêiner é feita através dos seguintes comandos:

```
> cd $GLOBUS_LOCATION
> globus-start-conteiner  -p 8080
```

Após a execução do comando de ativação, devem aparecer várias linhas mostrando os serviços disponíveis no contêiner e entre elas os serviços especialmente configurados nessa estação destacados pelas linhas em negrito abaixo. O terminal deve ficar preso na aplicação e para desativá-la basta teclar Ctrl-C que o contêiner será desativado.

```
> globus-start-conteiner  -p 8080

[01/25/2005 10:19:34:780 ] org.globus.ogsa.server.ServiceContainer
INFO: Starting SOAP server at: http://161.79.56.22:8080/ogsa/services/
With the following services:


              .
http://161.79.56.22:8080/ogsa/services/gsc/sac
http://161.79.56.22:8080/ogsa/services/gsc/sap
              .
```

4.4 Ensaios

Uma vez implementados todos os componentes do protótipo de grid de supervisão e controle, é chegada a hora de vê-lo em ação. Nesse tópico são apresentados os procedimentos para execução de ensaios que visam analisar o comportamento dos mecanismos da computação grid implementada pela OGSI. Apesar de nos ensaios existirem medições de tempo de processamento de tarefas esses valores não são conclusivos devido às condições de execução do ensaio onde não são levados em consideração vários detalhes de uma implementação de um grid de supervisão e controle real. Porém servem para fazer uma boa estimativa do desempenho das ferramentas do Globus Toolkit.

Devido a problemas encontrados na configuração do serviço CAS do Globus Toolkit não puderam ser executados procedimentos de autenticação e autorização durante os ensaios. A avaliação do serviço foi postergada para estudos futuros.

Os ensaios foram organizados em grupos de simulações que visam demonstrar os procedimentos listados abaixo.

- Estabelecimento ligações de supervisão e controle.
- Histórico da operação
- Processamento remoto

A seguir estão descritos os procedimentos passo a passo para execução dos ensaios e os resultados esperados. Os ensaios serão encadeados e representam um caminho mínimo para demonstração das principais características do grid de supervisão e controle apresentadas nesse trabalho. Não obstante o ambiente pode ser utilizado para vários outros tipos de ensaios não cobertos nesse trabalho.

4.4.1 Estabelecimento de ligações de supervisão e controle

O objetivo deste ensaio é verificar a flexibilidade obtida com o estabelecimento de ligações dinâmicas. De uma forma geral o ensaio consiste no estabelecimento de ligações entre os centros e as plantas elétricas. O centro faz uma pesquisa no grid por recursos e monta uma lista com os componentes que encontrou. A partir desta lista é

possível selecionar um componente do tipo usina ou subestação e estabelecer uma ligação com o componente. A ligação é composta por um canal de dados de supervisão na qual são enviados periodicamente varreduras com uma medida para cada medidor. E também é possível enviar controles operativos via protocolo SOAP.

Para inicio dos testes nenhum componente do grid de supervisão e controle deve estar ativado, sendo o primeiro passo ativar o servidor de infra-estrutura. Executar os comandos abaixo na conta globus da estação jaguaribe.

```
> cd $GLOBUS_LOCATION

> globus-start-conteiner  -p 8080

[01/25/2005 10:19:34:780 ] org.globus.ogsa.server.ServiceContainer
INFO: Starting SOAP server at:
http://161.79.56.39:8080/ogsa/services/
With the following services:

http://161.79.56.39:8080/ogsa/services/core/admin/AdminService
http://161.79.56.39:8080/ogsa/services/core/management/OgsiManag...
http://161.79.56.39:8080/ogsa/services/core/registry/Container...
                .
http://161.79.56.39:8080/ogsa/services/gsc/sac
                .
http:// 161.79.56.39:8080/ogsa/services/ogsi/HandleResolverService
http://161.79.56.39:8080/ogsa/services/base/cas/CASService
http://161.79.56.39:8080/ogsa/services/base/index/IndexService
                .
```

Verificar se os serviços destacados em negrito acima aparecem na ativação do contêiner. O terminal fica preso na aplicação e qualquer problema durante execução dos serviços será mostrado nesta janela.

Uma vez ativado o servidor de infra-estrutura será ativado um centro de controle para interagir com o grid de supervisão e controle. Executar os seguintes comandos, na estação verde usuário globus.

```
>cd ~/projetos

>java -classpath ./build/classes/:$CLASSPATH \
         org.globus.gsc.ihm.CentroControle &
```

Quando é a ativado o centro de controle, COR 1, faz uma pesquisa inicial no grid e descobre que o componente USI1 que reside no contêiner de serviços da estação jaguaribe está ativo, e o mostra na lista de componentes do grid da interface gráfica. Como USI1 não pertence à área elétrica monitorada pelo COR1 somente aparece listada no painel de componentes do grid.

Figura 4-18: Tela inicial de COR1

Para estabelecer uma ligação dinâmica com USI1 basta clicar no botão **Ativar** que está destacado na figura anterior. Quando estabelecer a ligação com USI1 aparecerá na janela de COR 1 uma interface com todas as informações sobre a usina com uma lista refrescada periodicamente com informações sobre as varreduras recebidas e um formulário que permite o envio de controles operativos. Conforme .Figura 4-19

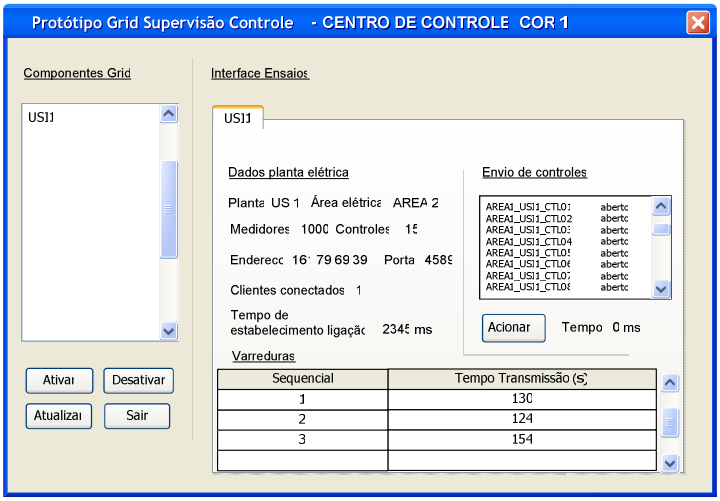


Figura 4-19: COR1 estabelece ligação com USI1

Ativar o contêiner na estação verde através dos seguintes comandos:

```
> cd $GLOBUS_LOCATION

> globus-start-conteiner  -p 8080

[01/25/2005 10:19:34:780 ] org.globus.ogsa.server.Service;;;
INFO: Starting SOAP server at: http://161.79.56.48:8080/....
With the following services:

http://161.79.56.48:8080/ogsa/services/core/admin/AdminService
http://161.79.56.48:8080/ogsa/services/core/management/O....
                .
http://161.79.56.48:8080/ogsa/services/gsc/sac
                .
```

Ao ser ativado o contêiner de serviços na estação verde o servidor de pesquisa e indexação notifica COR1 da entrada em operação da subestação SUBS1 que pertence à área elétrica sobe sua responsabilidade. Automaticamente é criada uma ligação entre o centro e a planta que é mostrada através de uma nova ficha no fichário de ligações, conforme a Figura 4-20. Nesse momento existem duas ligações ativas no centro de controle que podem ser acessadas clicando na aba do fichário.

Figura 4-20; COR1 estabelece ligações com SUBS1 e USI1

Para desativar uma ligação basta selecionar na lista um componente ativo e clicar em desativar. Nesse momento a interface da ligação será apagada.

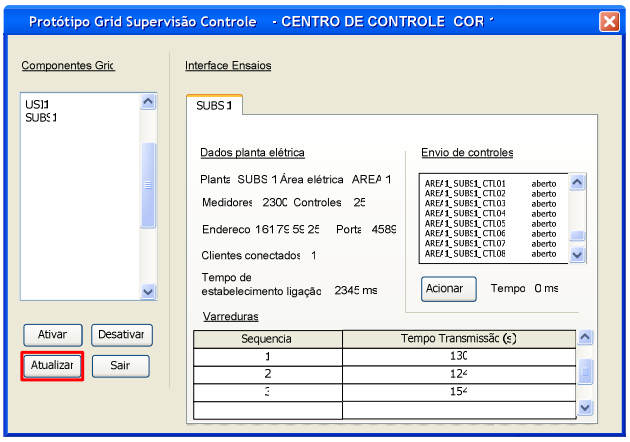


Figura 4-21 COR1 desativa ligação com USI1

O passo agora é ativar todos os demais componentes do protótipo. Para isso basta executar os comandos abaixo na conta globus das estações **grajaú, ivinhema e itaúba**.

Ativar o contêiner na estação verde através dos seguintes comandos:

```
> cd $GLOBUS_LOCATION

> globus-start-conteiner  -p 8080
```

Após a ativação dos demais componentes fazer uma pesquisa em COR1 clicando no botão Atualizar. Após a consulta serão listados todos os componentes configurados do grid de supervisão e controle. Conforme a figura adiante.

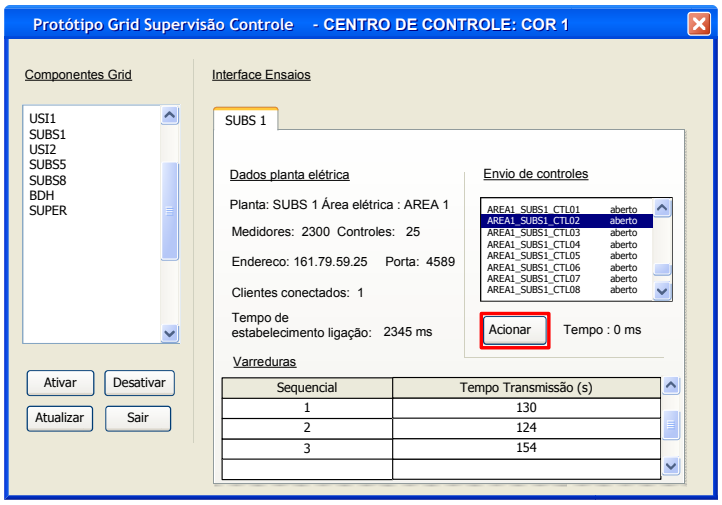


Figura 4-22: Os demais componentes do grid entram em operação

Nesse passo será executado um controle operativo na tela de COR1 selecionando um ponto de controle e clicando em **Acionar.** Após o envio e execução do comando a lista de pontos é atualizada com o novo estado e o tempo de execução do comando fica registrado ao lado do botão. Ver figura abaixo.

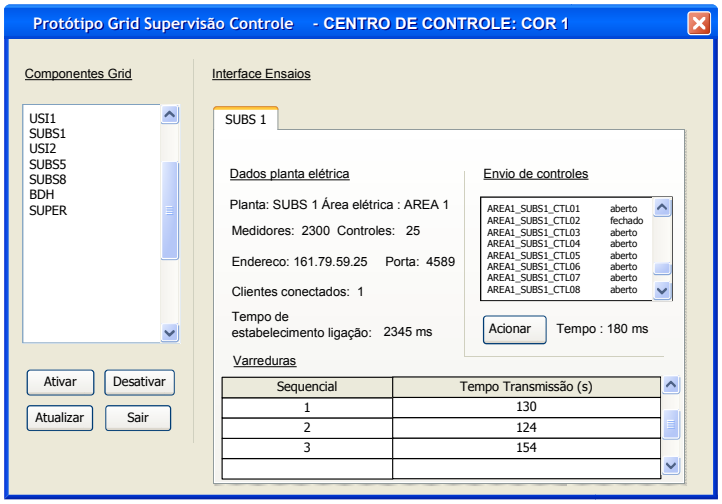


Figura 4-23: COR1 envia comando operativo à SUBS1

O passo seguinte é ativar um segundo centro de controle. Para isso deve ser ativado o centro COR2 localizado na estação grajaú. Executando os seguintes comandos como cliente globus.

```
>cd ~/projetos

>java -classpath ./build/classes/:$CLASSPATH \
         org.globus.gsc.ihm.CentroControle &
```

Uma vez ativada a interface serão criadas automaticamente ligações para todas as plantas elétricas pertencentes á área elétrica 2 sob responsabilidade de COR2. Conforme Figura 4-24.

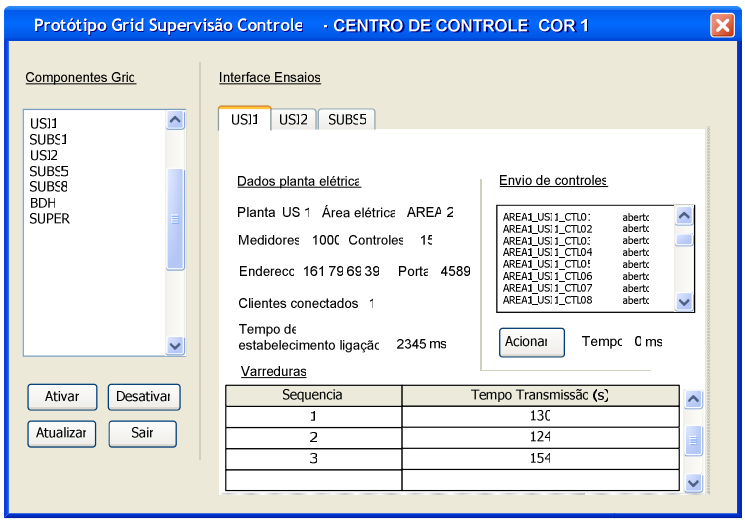


Figura 4-24: COR2 é ativado

Caso seja selecionada a SUBS1 e clicado o botão **Ativar** será estabelecida uma ligação entre COR2 e SUBS1. O que caracteriza uma situação de dois centros de controle monitorando uma mesma planta elétrica. O que permite que um centro execute todas ou parte das funções de outro centro quando necessário.

A partir desse ponto seria possível estender os ensaios para a ativação dos demais centros e estabelecer uma infinidade de combinações de ligações entre centros de controle e plantas elétricas demonstrando a grande flexibilidade do estabelecimento de ligações de supervisão e controle, trazida pela computação grid.

4.4.2 Histórico da operação

Nesse ensaio são verificados os mecanismos do Globus Toolkit para manipulação de grandes massas de dados. O ensaio consiste no envio de um arquivo contendo medidas elétricas em um determinado período e gravação desses dados em um banco de dados relacional.

Para executar esse ensaio voltar ao COR1 e selecionar BDH na lista de componentes ativos e clicar em **Ativar**. Conforme figura abaixo.

Figura 4-25: Ativando serviço de histórico da operação

Uma vez conectada com o componente BDH é criada uma ficha com varias informações e estatísticas sobre o servidor de armazenamento e que permite gerarmos uma massa de dados históricos a serem enviados a partir da seleção do número de medidores, período de varredura das medidas em minutos e número de dias a transmitir. Para o envio preencher após campos com os valores desejados e clicar em **Transmitir**.

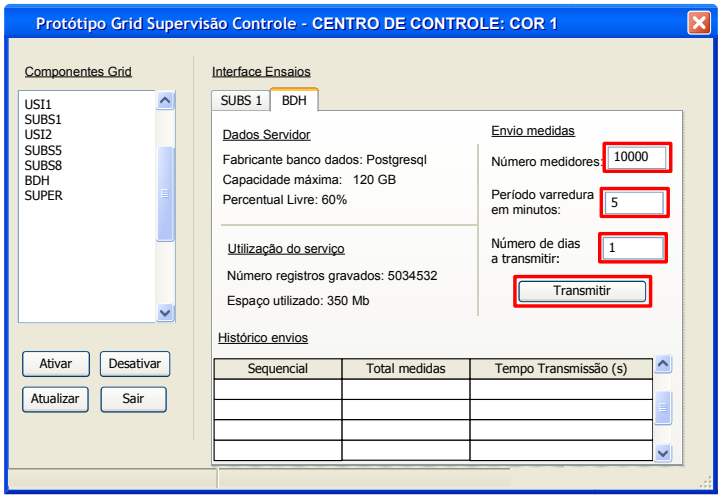


Figura 4-26: Interface de envio de histórico da operação

Após a geração, envio e armazenamento a tabela **Histórico envios** é atualizada com o número de medidas transmitidas e o tempo gasto para a transmissão e armazenamento no banco de dados relacional. Conforme mostra figura abaixo.

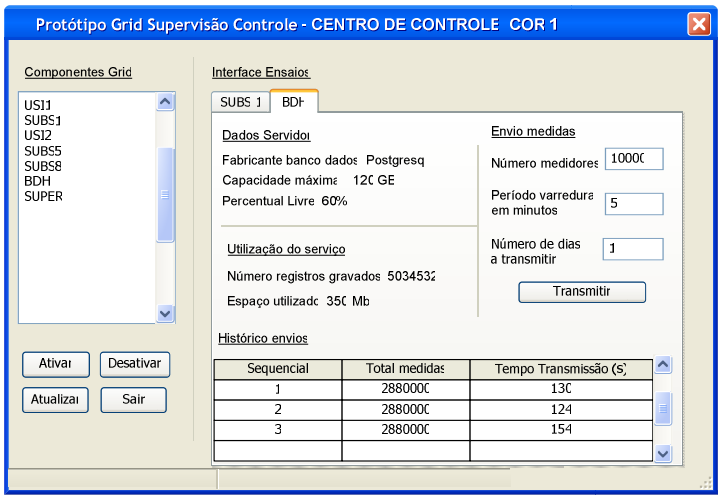


Figura 4-27: Estatísticas dos envios de arquivos

4.4.3 Servidor de aplicações

Para esse ensaio foi desenvolvido um componente que simula um supercomputador localizado na sede da empresa, capaz de cálculos sofisticados em grande velocidade, e que será compartilhado com os demais centros de controle do grid. O componente SUPER criado para a simulação é capaz de executar uma operação de somatório dos valores de todas as parcelas de um vetor que lhe é passado pelo cliente e retornar o resultado. A titulo de ilustração foi adicionado um custo associado a cada parcela do cálculo que será cobrado do usuário. A interface do cliente mostrará o total gasto por cliente conectado.

Para iniciar o ensaio, em COR1 selecione SUPER na lista de componentes ativos conforme figura abaixo.

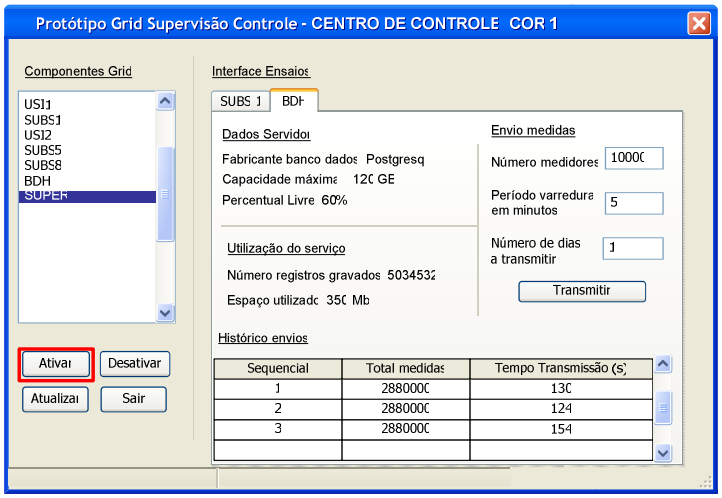


Figura 4-28: Ativando ligação com servidor de aplicações

Uma vez ativada a ligação entre o centro e o supercomputador, aparece na interface gráfica do centro uma nova ficha com informações sobre o servidor de aplicação. Conforme Figura 4-29

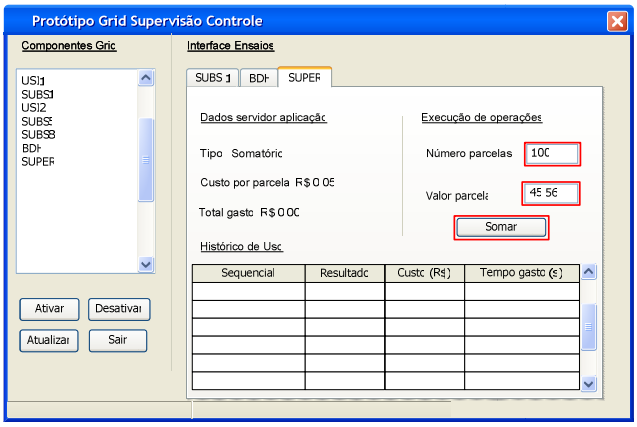


Figura 4-29: Interface com servidor de aplicações

A interface para SUPER implementada em COR1 permite definir um único valor para todas as parcelas e o número de parcelas do somatório. Ao se clicar em somar o vetor é enviado ao supercomputador e processado. O resultado é retornado é disponibilizado na tabela de **Histórico de Uso** junto com o custo do processamento e o tempo gasto na execução do calculo. Conforme é mostrada na Figura 4-30

Figura 4-30: Estatísticas de uso do servidor de aplicações

4.5 Comentários

Os ensaios executados confirmaram a flexibilidade oferecida pela nova arquitetura no estabelecimento de ligações de supervisão e controle dinâmicas. A forma de envio de dados adotada nos ensaios isenta a tecnologia grid de eventuais problemas de performance na transmissão já que os recursos da computação grid foram utilizados somente para o estabelecimento da ligação. A transmissão de medidas fica sob a responsabilidade da aplicação SCADA no caso uma simulação implementada através de classes internas ao serviço de aquisição e controle.

Na simulação de histórico da operação, o protocolo GridFtp mostrou-se uma ferramenta poderosa para a transferência de grandes massas de dados, utilizando de várias técnicas para aceleração do processo e recuperação de falhas tornando o serviço bem robusto e com uma performance compatível com as necessidades.

No caso dos experimentos com servidor de aplicações os testes não são conclusivos tendo em vista que o comportamento do serviço depende diretamente do tipo de aplicação utilizada. Porém as ferramentas do Globus Toolkit apresentam um bom ambiente para desenvolvimento desse tipo de aplicação incluindo ferramentas que não foram avaliadas nesse trabalho como, por exemplo, o GRAM (*Grid Resource Allocation and Management*) [3,5,8,14] que oferece uma interface muito poderosa para o desenvolvimento de aplicações que utilizam recursos distribuídos em rede.

O objetivo principal dos ensaios realizados foi fazer uma primeira incursão na programação de aplicativos baseados em computação grid para sistemas de supervisão e controle em tempo real. A partir desse trabalho espera-se uma continuidade avaliando aspectos ainda não cobertos e integração com aplicativos de supervisão e controle reais, para que seja feita uma análise mais profunda das soluções adotadas.

No próximo capitulo serão feitas observações sobre o impacto da adoção da tecnologia na operação de sistemas elétricos e finalmente serão feitas considerações sobre o futuro das redes de supervisão e controle e onde a computação grid se encaixa como solução para os novos desafios a serem enfrentados.

5 - Conclusão

Esse capítulo está reservado para observações finais e conclusões sobre o trabalho. Inicialmente é feita uma avaliação dos principais aspectos da aplicação da computação grid em sistemas de supervisão e controle para em seguida serem feitos os comentários finais.

5.1 Principais aspectos

Dentro das funcionalidades e características da computação Grid implementada através do Globus Toolkit algumas se destacam no âmbito da supervisão e controle.

5.1.1 Ligações de supervisão e controle dinâmicas

Esse certamente é a funcionalidade que trará o primeiro grande impacto na forma de se encarar as redes de supervisão e controle. O que atualmente é configurado de forma estática e obedece a uma rígida estrutura hierárquica, com a computação grid será dinâmico e estará disponível, de forma segura e flexível. Esse aspecto trará uma grande versatilidade às redes de supervisão e controle, permitindo que um centro assuma as funções de outro centro em eventuais indisponibilidades. Um outro exemplo da versatilidade seria, por exemplo, no caso de uma crise em uma determinada parte do sistema elétrico seria possível criar uma força tarefa com a participação de engenheiros dos vários centros de controle da empresa e disponibilizar a todos, de forma trivial, uma visão uniforme do foco do problema.

5.1.2 Integração com sistema legado

Uma das questões chave para o sucesso da computação grid é a capacidade deste integrar em suas soluções ao sistema legado. No âmbito da supervisão e controle a capacidade de integrar os protocolos de comunicação já existentes é fundamental para o processo de migração para o novo ambiente. Para atender a essa questão o Globus Toolkit oferece um pacote de serviços denominado Globus XIO. O pacote oferece uma API baseada em diretivas básicas do tipo *open/close/read/write* que permite a implementação de protocolos de comunicação com relativa facilidade. Com ele será

possível criar implementações de protocolos de tais como o ICCP/TASE., DNP3 ou IEC 61850 estendidos com novas funcionalidades disponibilizadas com computação grid.

5.1.3 Segurança

A segurança é um dos aspectos mais importantes na computação moderna e devido à natureza fechada das redes de supervisão e controle elétricas nunca foi uma grande questão para o setor. Porém, com a crescente interligação dos sistemas e como já vemos, atualmente em algumas empresas, o centro de controle conectado à rede corporativa, a questão se torna fundamental. O Globus já traz nativa uma solução o GSI – *Grid Security Infra-estructure*, que implementa funções para autenticação, autorização e criptografia o que garante um nível satisfatório de segurança para os sistemas desenvolvidos.

5.1.4 Facilidade de implementação de novos serviços, expansão.

Um dos principais aspectos quando vamos avaliar uma nova tecnologia e o grau de dificuldade para implementação de novos sistemas. Nesse ponto o Globus é uma solução ainda em desenvolvimento. O processo de aprendizagem das ferramentas do ambiente é muito dificultado pela falta de uma documentação completa, o que existe disponível no momento está em fase de acabamento. A codificação se novos serviços também é um processo trabalhoso, pois além da codificação da aplicação propriamente dita é preciso preencher manualmente arquivos no formato XML para implantação do serviço no servidor de aplicação. Esse problema deverá ser resolvido por iniciativas iguais à da IBM que está desenvolvendo para a ferramenta de desenvolvimento Eclipse 3.0 um *plug-in* que permitirá a geração desses arquivos de forma automática a partir de definições feitas através de uma interface gráfica. Apesar dessas dificuldades iniciais a API do Globus é composta por pacotes de serviços bem organizados e que trazem um excelente grau de abstração simplificando muito o código dos serviços.

5.2 Considerações finais

A adoção da computação grid trará grandes benefícios na implantação de sistemas automatizados para apoio a várias áreas de atuação do setor elétrico. Sua capacidade de compartilhamento de recursos otimizará os investimentos em hardware e software permitindo um maior retorno do capital investido. Por exemplo, os centros de controle da empresa poderão compartilhar o uso de um novo sistema para recomposição de redes elétricas que foi instalado no supercomputador localizado na sede da empresa. Uma concessionária de energia elétrica poderá criar centros especializados em tarefas como armazenar histórico da operação ou processamento de cálculos complexos e disponibilizá-los para todos os setores da empresa interessados.

A forma dinâmica como se estabelecem os relacionamentos entre os participantes do grid possibilitará redes de supervisão mais robustas, tornando trivial a tarefa de um centro substituir outro em eventualidades. Também solucionará antigos problemas na reconfiguração da rede elétrica monitorada, onde a entrada ou saída de uma área elétrica interfere no funcionamento geral de um centro de supervisão e controle, criando perigosos períodos de ausência de monitoramento no sistema elétrico.

Os históricos da operação do sistema serão mais precisos. Atualmente os custos de investimento e manutenção em bancos de dados leva as empresas a adotarem uma política de manter uma base de dados histórica no centro de controle que encabeça a hierarquia de supervisão e controle onde somente são armazenados os dados monitorados por este centro, vimos anteriormente que ao subir de nível em uma hierarquia há perda de informação. Com a técnica sugerida neste trabalho de implementação de um servidor de armazenamento de dados históricos da operação será possível gerar uma base de dados com todas as informações sobre a operação de todas as plantas elétricas monitoradas independentemente de quem as monitora. Com dados históricos mais precisos será possível uma programação da operação também mais precisa trazendo um melhor aproveitamento dos recursos disponíveis. Outro aspecto importante da melhoria da precisão do histórico da operação é que quanto mais precisos o dados utilizados, mais precisas serão as previsões para o futuro. O setor de planejamento da expansão do sistema elétrico contará com previsões mais confiáveis que auxiliarão na tomada de decisão sobre novos investimentos.

Fazendo um pequeno exercício de futurologia, podemos perceber que o grid pode ser chave na solução dos novos desafios do setor. Por exemplo, atualmente verifica-se uma tendência de que cada vez mais, pequenos geradores entrem o mercado de energia utilizando fontes alternativas de energia mais adaptadas às condições locais, renováveis e menos poluentes, formando redes de micro-geração. A tendência é de que nas próximas décadas existam centenas ou até milhares desses micro-geradores conectados na rede elétrica, acarretando redes de supervisão e controle complexas. A tecnologia atual não é escalável o suficiente para atender a esse grande número de micro-geradores. A computação grid oferece uma solução relativamente barata mais uma vez graças à natureza dinâmica dos relacionamentos entre seus componentes.

Tal como foi um dia a *web* fez uma revolução na forma de se usar o computador, o grid está chegando levar a informática para um novo patamar que transformará a forma de usarmos o computador. Como todos outros setores de atividade, o setor elétrico encontrará no grid solução para problemas atuais e futuros

6 - Bibliografia

[1] The Grid: Blueprint for a Future Computing Infrastructure.I. Foster and C. Kesselman, editors. Morgan Kaufmann Publishers, 1998.

[2] The Anatomy of the Grid: Enabling Scalable Virtual Organizations. I. Foster, C. Kesselman, S. Tuecke. International J. Supercomputer Applications, 15(3), 2001. http://www.globus.org/research/papers/anatomy.pdf

[3] The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. I. Foster, C. Kesselman, J. Nick, S. Tuecke, Open Grid Service Infrastructure WG, Global Grid Forum, Junho, 2002 http://www.globus.org/research/papers/ogsa.pdf

[4] Globus Toolkit 3.2 Documentation, The Globus Alliance -http://www-unix.globus.org/toolkit/docs/3.2/index.html -2004

[5] Index grid services using Globus Toolkit 3.0, Cai Jun Jie, IBM Globalization Certification Lab., 2004. http://www-106.ibm.com/developerworks/grid/library/gr-indexgrid

[6] The Globus Toolkit 3 Programmer's Tutorial, Borja Sotomayor, 2004. http://www.casa-sotomayor.net/gt3-tutorial.

[7] Deploy a C application as a grid service, Jean-Yves Girard, Ioane Muni Toke, Jean-Luc Lepesant, Jean-Pierre Prost. IBM – Developer Works, 2004. http://www-128.ibm.com/developerworks/grid/library/gr-deploy/index.html

[8] Grid Infrastructure to support science portals for large scale instruments, Gregor Von Laszwski and Ian Foster. Argonne National Laboratory, 1999. www.mcs.anl.gov/~gregor/ papers/vonLaszewski--rostock.pdf

[9] The WS-Resource Framework version 1.0, Karl Czajkowski, Donald F Ferguson, Ian Foster, Jeffrey Frey, Steve Graham, Igor Sedukhin, David Snelling, Steve Tuecke, William Vambenepe. GGF – Global Grid Forum, 2004. www-106.ibm.com/developerworks/ library/ws-resource/ws-wsrf.pdf

[10] Enabling Applications for Grid Computing with Globus, Bart Jacob, Luis Ferreira, Norbert Bieberstein, Candice Gilzean, Jean-Yves Girard, Roman Strachowski, Seong (Steve) Yu. IBM Redbooks, 2003.
http://www.redbooks.ibm.com/redbooks/pdfs/sg246936.pdf

[11] An overview of the Web Services Inspection Language, Peter Brittenham. IBM DeveloperWorks, 2002.
http://xml.coverpages.org/IBM-WS-Inspection-Overview.pdf

[12] A developer's overview of OGSI and OGSI-based Grid computing, Joshy Joseph. IBM DeveloperWorks, 2002.
http://www-106.ibm.com/developerworks/grid/library/gr-gt3/

[13] Open Grid Service Infrastructure Primer. GGF – Global Grid Forum, 2004.
http://pcbunn.cacr.caltech.edu/GAE/workshop/draft-ggf-ogsi-gridserviceprimer-1.pdf

[14] Open Grid Services Infrastructure (OGSI), S. Tuecke, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt. GGF – Global Grid Forum, 2003.
http://www.globus.org/research/papers/Final_OGSI_Specification_V1.0.pdf

[15] A Grid Application Framework based on Web Services Specifications and Practices, Savas Parastatidis, Jim Webber, Paul Watson, Thomas Rischbeck. North East Regional e-Science Centre, 2003.
http://www.cs.ncl.ac.uk/research/pubs/trs/papers/825.pdf

[16] Globus Toolkit 3 Core – A Grid Service Container Framework, Thomas Sandholm, Jarek Gawor. GGF – Global Grid Forum, 2003.
http://www-unix.globus.org/toolkit/3.0/ogsa/docs/gt3_core.pdf

[17] Global Grid Connectivity Using Globus Toolkit With Solaris Operating System, Chong-Wee Simon See, Gabriel Ghinita. Sun BluePrints, 2004.
http://www.sun.com/blueprints/0504/817-6228.pdf

[18] Globus Toolkit 3.0 0 Quick Start, Luis Ferreira, Bart Jacob, Sean Slevin, Michael Brown, Srikrishnan Sundararajan, Jean Lepesant, Judi Bank. IBM Redbooks, 2003. http://www.redbooks.ibm.com/redpapers/pdfs/redp3697.pdf

[19] EZ-Grid: Integrated Resource Brokerage Services for Computational Grids, Barbara M. Chapman, Babu Sundaram, Kiran K. Thyagaraja. Department of Computer Science, University of Houston.
http://www2.cs.uh.edu/~hpctools/EZ-Grid-Abstract.pdf

[20] The Grid: past, present, future, in Berman, F., Fox, G. C. and Hey, A. J. G., Eds. Grid Computing - Making the Global Infrastructure a Reality, chapter 1, pages pp. 9-50. Wiley and Sons.
http://eprints.ecs.soton.ac.uk/9098/01/The%5FGrid%5FPast%5FPresent%5FFuture.pdf

[21] Application Experiences with the Globus Toolkit. S. Brunett, K. Czajkowski, S. Fitzgerald, I. Foster, A. Johnson, C. Kesselman, J. Leigh, S. Tuecke. Proceedings of 7th IEEE Symp. on High Performance Distributed Computing, July 1998. ftp://ftp.globus.org/pub/globus/papers/globus-apps.pdf

[22] Grid Service Specification, Steven Tuecke, Karl Czajkowski, Ian Foster, Jeffrey Frey, Steve Graham, Carl Kesselman. . GGF – Global Grid Forum, 2002.
http://www.globus.org/research/papers/gsspec.pdf

[23] Globus: A Metacomputing Infrastructure Toolkit. I. Foster, C. Kesselman. Intl J. Supercomputer Applications, 11(2):115-128, 1997.
ftp://ftp.globus.org/pub/globus/papers/globus.pdf

[24] What is SCADA? Axel Daneels, Wayne Salter, IT/CO, 7th International Conference on Accelerator and Large Experimental Physics Control Systems, held in Trieste, Italy, 4 - 8 Oct. 1999.
http://ref.web.cern.ch/ref/CERN/CNL/2000/003/scada/Pr/

[25] DNP3 Overview, Triangle MicroWorks, Inc. http://www.trianglemicroworks.com/documents/DNP3_Overview.pdf

[26] Communication Standards in Power Control, Andrew C. West B.E., B.Sc., B.A., P.Eng, Grad. I.E., Aust, MIEEE. Triangle MicroWorks Inc.

[27] IEEE Recommended Practice for Master / Remote Supervisory Control and Data Acquisition (SCADA) Communications. IEEE Power Engineering Society, 1993.

[28] IEEE Recommended Practice for Data Communications Between Remote Terminal Units and Intelligent Electronic Devices in a Substation. IEEE Power Engineering Society, 2000.

[29] IEC 61850, IEC 61400-25 and IEC 61970: Information Models and information exchange for electric power systems. Karlheinz Scharz. Schwarz Consulting Company. 2002.

[30] Comparison of IEC 60870-5-101/-102/-104, DNP3 and IEC 60870-6-TASE.2 with IEC 6850. Karlheinz Scharz. Schwarz Consulting Company. 2002.

[31] Plug in to Grid Computing, Malcom Irving, Gareth Taylor and Peter Hobsom. Power and Energy Magazine, IEEE, 2004.

[32] Apache Ant 1.6.2 Manual, The Apache Software Foundation- http://ant.apache.org/manual/index.html – 2004

[33] JUnit Cookbook, Kent Beck, Erich Gamma- http://junit.sourceforge.net/doc/cookbook/cookbook.htm,2004

[34] JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification, Sun Microsystems, Inc. http://java.sun.com/j2se/1.4.2/docs/api/index.html

[35] PostgreSQL 7.4.5 Documentation, The PostgreSQL Global Development Group, 2004 – http://postgresql.org

7 - Glossário

ANT Apache Ant, ferramenta para construção de software baseada em Java e XML.

API *Application Program Interface*, conjunto de bibliotecas de funções que permitem a integração entre sistemas computacionais.

CA *Certificate Authority*, componente em um grid responsável por emitir certificados digitais identificadores dos demais componentes do grid.

CAS *Community Authorization Service,* componente do Globus Toolkit responsável pela manutenção de regas de acesso aos recursos disponíveis no grid.

CEPEL Centro de Pesquisas de Energia Elétrica.

CIM Common Information Model, modelo de dados proposto pelo EPRI para descrição de sistemas elétricos.

COR Centro de Operação Regional, centro de controle responsável pela operação de uma área elétrica pertencente a uma empresa concessionária de energia elétrica.

COS Centro de Operação do Sistema, centro de controle responsável pela operação do sistema elétrico interligado de uma empresa concessionária de energia elétrica.

CSV Comma Separated Values, formato de arquivo texto em que as informações estruturadas são organizadas em colunas separadas por um caractere especial, normalmente vírgula.

DNP3.0 *Distributed Network Protocol* 3.0, protocolo de supervisão e controle de domínio publico criado pela GE Harris em 1990.

EPRI *Eletric Power Research Institute*, centro de pesquisas de energia elétrica norte americano.

FTP File Transfer Protocol, protocolo para transmissão de arquivos em redes TCP/IP.

GAR *Grid Archive,* arquivo que empacota todos os arquivos necessários para publicação de um determinado serviço em um contêiner de serviços grid.

Globus Toolkit Software de código aberto para desenvolvimento de grids computacionais conforme a definição feita pela OGSA, *Open Grid Services Architecture*, que implementa a infra-estrutura de serviços especificada pela OGSI, *Open Grid Services Infrastructure*.

GRAM *Grid Resource Allocation and Management,* componente do Globus Toolkit responsável pelo gerenciamento do acesso de recursos computacionais remotos durante a execução de aplicações.

GRIDFTP Protocolo de transferência de arquivos de alta performance especialmente desenvolvido para computação Grid.

GridService Classe contendo as funcionalidades básicas de um provedor de serviços em um grid computacional.

GSDD *Grid Services Deployment Descriptor*, tipo de arquivo texto no formato XML contendo informações para integração do serviço em um container de serviços do Globus Toolkit.

GSH *Grid Service Handler*, endereço, no formato de uma URI, de um determinado serviço no grid computacional.

GSI Grid Security Infrastructure, componente do Globus Toolkit responsável pela infra-estrutura de segurança do grid computacional.

GSR Grid Service Reference, documento XML no formato GWSDL descrevendo os serviços disponibilizados por um determinado componente e como acessá-los.

GWSDL *Grid Web Services Description Language*, linguagem baseada em XML para descrição de serviços grid.

ICCP *Inter Control Center Protocol*, protocolo de comunicação entre centros de controle proposto pelo EPRI.

LASC Laboratório Avançado de Supervisão e Controle, laboratório localizado no CEPEL voltado ao desenvolvimento de soluções computacionais para supervisão e controle de sistemas elétricos.

MDS *Monitoring and Discovery System*, implementa o serviço de informações sobre o grid computacional. Através dele é possível determinar os recursos disponíveis no grid e em que estado se encontram.

OGSA *Open Grid Services Architecture,* conjunto de definições desenvolvidas pelo Global Grid Fórum com o objetivo de definir um padrão arquitetônico para aplicações baseada em computação grid.

OGSI *Open Grid Services Infrastructure*, infra-estrutura de serviços, definida a partir da OGSA, para a implementação de um ambiente de desenvolvimento de aplicações baseada em computação grid.

RFT *Reliable File Transfer*, componente do Globus Toolkit que formence uma API para utilização do protocolo GridFtp a partir de aplicações habilitadas para computação grid.

SCADA Supervisory Control And Data Acquisition, tipo de sistema computacional utilizado para operar redes eétricas em tempo real.

SDE *Service Data Element*, estrutura de dados associados aos serviços grid, utilizados na descrição das características dos serviços e seus estados. Também são utilizados na implementação de mecanismos de notificação.

SOAP *Simple Object Access Protocol*, protocolo baseado em troca de arquivos XML para acesso a serviços remotos em sistemas distribuídos.

SSL *Secure Socket Layer*, protocolo desenvolvido pela Netscape para transmissão de documentos criptografados pela Internet.

URI *Uniform Resource Identifier*, termo genérico para especificar todos os tipos de nome e endereços que referenciam objetos na Internet.

URL *Uniform Resource Locator*, especifica o endereço de um recurso ou um documento na *World Wide Web*.

WSDL *Web Services Description Language*, linguagem baseada em XML para descrição de serviços web.

X-509 Padrão utilizado da confecção de certificados digitais.

XIO Componente do Globus Toolkit que permite do desenvolvimento de protocolos de comunicação para aplicações baseada em computação grid.

XML *Extensible Markup Language,* linguagem de marcação de dados que provê um formato para descrever dados estruturados.

Apêndice A Scripts e Fontes

A seguir são listados os conteúdos dos arquivos fontes de programas, scripts e configurações utilizadas neste trabalho.

A.1 Arquivos comuns

A.1.1 namespace2package.mappings

Localização: ~globus/projetos/ namespace2package.mappings

```
http://www.globus.org/namespaces/2004/02/gsc/ServicoAquisicaoControle=
        org.globus.gsc.stubs.sac

http://www.globus.org/namespaces/2004/02/gsc/ServicoAquisicaoControle/
        bindings= org.globus.gsc.stubs.ServicoAquisicaoControle.bindings

http://www.globus.org/namespaces/2004/02/gsc/ServicoAquisicaoControle/ service=
        org.globus.gsc.stubs.ServicoAquisicaoControle.service

http://www.globus.org/namespaces/2004/02/gsc/ServicoAquisicaoControle/
        ServicoAquisicaoControleSDE=
        org.globus.gsc.stubs.ServicoAquisicaoControle.servicedata
http://www.globus.org/namespaces/2004/02/gsc/HistoricoOperacao=
        org.globus.gsc.stubs.sac

http://www.globus.org/namespaces/2004/02/gsc/HistoricoOperacao/bindings=
        org.globus.gsc.stubs.HistoricoOperacao.bindings

http://www.globus.org/namespaces/2004/02/gsc/HistoricoOperacao/service=
        org.globus.gsc.stubs.HistoricoOperacao.service

http://www.globus.org/namespaces/2004/02/gsc/HistoricoOperacao/HistoricoOperacaoSDE=
        org.globus.gsc.stubs.HistoricoOperacao.servicedata

http://www.globus.org/namespaces/2004/02/gsc/ServidorAplicacao=
        org.globus.gsc.stubs.sac

http\://www.globus.org/namespaces/2004/02/gsc/ServidorAplicacao/bindings=
        org.globus.gsc.stubs.ServidorAplicacao.bindings

http\://www.globus.org/namespaces/2004/02/gsc/ServidorAplicacao/service=
        org.globus.gsc.stubs.ServidorAplicacao.service

http\://www.globus.org/namespaces/2004/02/gsc/ServidorAplicacao/ServidorAplicacaoSDE=
        org.globus.gsc.stubs.ServidorAplicacao.servicedata
```

A.1.2 build.properties

Localização: ~globus/projetos/build. properties

```
ogsa.root=/usr/globus/gt321
```

A.1.3 monta_servico.sh

Localização: ~globus/projetos/monta_servico.sh

```
#!/bin/bash


printUsage() {
echo ""
echo "Usage:"
echo "$0 <service_dir> <schema_file>"
echo ""
echo "<service_dir> is the directory that contains all the grid service files:"
echo -e "\t<service_dir>/server-deploy.wsdd    Deployment file (mandatory)"
echo  -e  "\t<service_dir>/impl/*.java                Java implementation files
(mandatory)"
echo -e "\t<service_dir>/config/*.xml          Configuration files (optional)"
echo ""
echo "<schema_file> is the GWSDL file with the service's interface description"
}

SERVICE_DIR=$1
SCHEMA_PATH=$2


if [ ! $# -eq 2 ]
then
    echo "ERROR: Wrong number of parameters."
    printUsage
    exit 1;
fi

if [ ! -e $SERVICE_DIR -o ! -d $SERVICE_DIR ]
then
    echo "$SERVICE_DIR does not exist or is not a directory."
    printUsage
    exit 1;
fi

if [ ! -e $SCHEMA_PATH -o ! -f $SCHEMA_PATH ]
then
    echo "$SCHEMA_PATH does not exist or is not a file."
    printUsage
    exit 1;
fi


if [ $(echo $SERVICE_DIR | rev | cut -c1) == "/" ]
```

```
then
    SERVICE_DIR=$(echo $SERVICE_DIR | rev | cut -c2- | rev)
fi

PACKAGE=$(echo $SERVICE_DIR | sed "s/\//\./g")
SCHEMA_DIR=$(echo $SCHEMA_PATH | cut -d/ -f2- | rev | cut -d/ -f2- | rev)
SERVICE_NAME=$(echo $SCHEMA_DIR | rev | cut -d/ -f1 | rev)
SCHEMA_FILE=$(echo $SCHEMA_PATH | rev | cut -d/ -f1 | rev)
INTERFACE=$(echo $SCHEMA_FILE | rev | cut -d. -f2 | rev)
GAR_FILENAME=$(echo $PACKAGE | sed "s/\./_/g")

eval       ant       -Dpackage=$PACKAGE       -Dinterface.name=$INTERFACE       -
Dpackage.dir=$SERVICE_DIR              -Dschema.path=$SCHEMA_DIR              -
Dservice.name=$SERVICE_NAME -Dgar.filename=$GAR_FILENAME
return_code=$?

exit $return_code;
```

A.1.4 build.xml

Localização: ~globus/projetos/build.xml

```
<?xml version="1.0"?>
<!--
This file is licensed under the terms of the Globus Toolkit Public
License, found at http://www.globus.org/toolkit/download/license.html.
-->
<!--
Written by Borja Sotomayor, Sebastien Barre
-->
<project default="all" basedir=".">

  <!--
    Give user a chance to override without editing this file (and
    without typing -D each time it compiles it) by loading in a
    project-specific build.properties file or a user-specific
    ${user.home}/build.properties file.
  -->

  <property file="build.properties"/>
  <property file="${user.home}/build.properties"/>

  <!--
    Set some properties for the build and lib directory.
    Note that it is definitely safer to use absolute paths here,
    since we will be calling GT3 build scripts.
  -->

  <property name="build.dir" value="${basedir}/build"/>
  <property name="build.bin" value="${build.dir}/bin"/>
  <property name="build.lib" value="${build.dir}/lib"/>
  <property name="build.stubs" value="${build.dir}/stubs"/>
  <property name="stubs.dest" value="../stubs"/>
  <property name="build.dest" value="${build.dir}/classes"/>
  <property name="build.schema" value="${build.dir}/schema"/>
  <property name="lib.dir" value="${basedir}/lib"/>
  <property name="schema.dir" value="${basedir}/schema/${schema.path}"/>
```

```
<!--
  Set some properties related to the GT3 installation directory.
-->

<property name="build.client" value="${ogsa.root}/build-client.xml"/>
<property name="build.services" value="${ogsa.root}/build-services.xml"/>
<property name="build.packages" value="${ogsa.root}/build-packages.xml"/>
<property name="build.deploy" value="${ogsa.root}/build.xml"/>
<property name="schema.origin" value="${ogsa.root}/schema"/>
<property file="${ogsa.root}/ogsa.properties"/>

<!--
  The namespace mappings.
-->

<property name="mapping.file" value="namespace2package.mappings"/>

<!-- SEBASTIEN
     Set a useful CLASSPATH.
     Explicitly exclude the jar files matching the service being built, so
     that an old version is not picked before the current one.
 -->

<path id="classpath">
  <pathelement location="${java.home}/../lib/tools.jar"/>
  <pathelement location="./"/>
  <pathelement location="${build.dest}"/>
  <fileset dir="${lib.dir}">
    <include name="*.jar"/>
  </fileset>
  <fileset dir="${ogsa.root}/lib">
   <!-- BORJA
        The following line should fix the "why don't my stubs get generated??"
        problem
        when you try to recompile a service that is already deployed -->
    <include name="*.jar"/>
    <exclude name="progtutorial_${service.name}-stub.jar"/>
  </fileset>
  <!-- SEBASTIEN
       IMPORTANT: the following line is commented-out. It would have
       added the current CLASSPATH value to our classpath ref. If this
       CLASSPATH actually includes a reference to this service (for
       example, if it has been deployed already in OGSA/GT), this
       outdated reference *will* cause a lot of trouble (part of the
       stubs won't be generated, the client will pick the ref, etc).
       So we decided here not to include it, and we make sure this
       safe value is propagated to the OGSA/GT build scripts we call
       by using <reference refid="classpath"/> inside <ant>
  -->
  <!--
  <pathelement path="${java.class.path}"/>
  -->
</path>

<!-- SEBASTIEN
     Echo info about the package being built.
-->
<target name="echoPackageInfo">
  <echo message="package: ${package}"/>
  <echo message="interface.name: ${interface.name}"/>
  <echo message="package.dir: ${package.dir}"/>
  <echo message="schema.dir: ${schema.dir}"/>
```

```
    <echo message="service.name: ${service.name}"/>
    <echo message="gar.filename: ${gar.filename}"/>
    <echo message="java.home: ${java.home}"/>
  </target>

  <!-- SEBASTIEN
       Create build tree and copy all OGSA/GT3 schema files over so that
       they can be accessed from our GWSDL files.
       Explicitly exclude the schema matching the service being built, so
       that an old version is not copied over.
  -->

  <target name="createBuildTree">
    <mkdir dir="${build.dest}"/>
    <mkdir dir="${build.bin}"/>
    <!-- BORJA
         Start out with a new "lib" directory everytime. This avoids unwanted
         JARs ending up in the GAR file.
         TODO: Find a better way of making sure that only the service's JARs
               plus any extra JARs (included in $TUTORIAL_DIR/lib) are included,
               but not the JARs from previously compiled services.
      -->
    <delete dir="${build.lib}"/>
    <mkdir dir="${build.lib}"/>
    <mkdir dir="${build.schema}"/>
    <mkdir dir="${build.stubs}"/>
  </target>

  <target name="copySchemaFiles" depends="createBuildTree">
    <copy toDir="${build.schema}">
      <fileset dir="${schema.origin}">
        <exclude name="**/progtutorial/${service.name}/**"/>
      </fileset>
    </copy>
  </target>

  <!--
    Copy the service config directory contents to the build tree.
  -->

  <target name="configDirAvailable">
    <available file="${package.dir}/config" property="config.dir.present"/>
  </target>
  <target name="copyConfigFiles"
          depends="configDirAvailable,createBuildTree"
          if="config.dir.present">
    <mkdir dir="${build.dest}/${package.dir}/config"/>
    <copy toDir="${build.dest}/${package.dir}/config">
      <fileset dir="${package.dir}/config"/>
    </copy>
  </target>

  <!--
    Merge the service mappings.
  -->

  <target name="mappingFileAvailable">
    <available file="${mapping.file}" property="mapping.file.present"/>
  </target>
  <target name="mergeMapping"
          depends="mappingFileAvailable"
          if="mapping.file.present">
    <ant antfile="${build.services}"
```

```
        target="mergePackageMapping">
        <reference refid="classpath"/>
    </ant>
  </target>

  <!--
    Set the environment (i.e., execute the above targets).
  -->

  <target name="setenv"
      depends="echoPackageInfo,createBuildTree,copySchemaFiles,
              copyConfigFiles,mergeMapping"/>

  <!--
    Generate WSDL files from service GWSDL file.
  -->

  <target name="checkSchemaUptodate">
    <uptodate property="schema.uptodate">
      <srcfiles dir="${schema.dir}"/>
      <mapper type="glob"
              from="*"
              to="${build.schema}${file.separator}progtutorial${file.separator}$
                  {service.name}${file.separator}*"/>
    </uptodate>
  </target>

  <target name="generateWSDLfromGWSDL" depends="setenv,checkSchemaUptodate">
    <property name="schema.file" value="${interface.name}_service.wsdl"/>

    <copy todir="${build.schema}/progtutorial/${service.name}"
          overwrite="false">
      <fileset dir="${schema.dir}"/>
    </copy>

    <ant antfile="${build.services}" target="GWSDL2WSDL">
      <property name="build.schema.dir" value="${schema.path}"/>
      <property name="wsdl.root" value="${interface.name}"/>
      <reference refid="classpath"/>
    </ant>

    <ant antfile="${build.services}" target="generateBinding">
      <property name="binding.root" value="${interface.name}"/>
      <property name="build.schema.dir" value="${schema.path}"/>
      <property name="porttype.wsdl" value="${interface.name}.wsdl"/>
      <reference refid="classpath"/>
    </ant>
  </target>

  <!--
    Generate, compile and jar stubs.
  -->

  <target name="generateStubs"
          depends="generateWSDLfromGWSDL"
          unless="schema.uptodate">
    <ant antfile="${build.services}"
         target="generateStubs">
      <property name="schema.file.dir" value="${schema.path}"/>
      <property name="schema.file" value="${schema.file}"/>
      <reference refid="classpath"/>
    </ant>
  </target>
```

```
  <target name="compileStubs" depends="generateStubs">
    <javac srcdir="${build.stubs}"
           destdir="${build.dest}"
           debug="${debug}"
           deprecation="${deprecation}"
           classpathref="classpath">
    </javac>
  </target>

  <target name="jarStubs" depends="compileStubs">
    <jarjarfile="${build.lib}/progtutorial_${service.name}-stub.jar"
basedir="${build.dest}" >
      <include name="**/stubs/${service.name}/**" />
    </jar>
  </target>

  <!--
    Compile and jar service.
  -->

  <target name="compileService" depends="compileStubs">
    <javac srcdir="./"
           includes="${package.dir}/impl/**"
           destdir="${build.dest}"
           debug="${debug}"
           deprecation="${deprecation}"
           classpathref="classpath">
    </javac>
  </target>

  <target name="jarService" depends="compileService">
    <jar jarfile="${build.lib}/${package}.jar" basedir="${build.dest}" >
      <include name="**/${package.dir}/impl/**" />
      <include name="**/${package.dir}/config/**" />
    </jar>
  </target>

  <!--
    Gar stubs and service, deploy.
  -->

  <target name="gar" depends="jarStubs,jarService">
    <copy todir="${build.lib}">
      <fileset dir="${lib.dir}">
        <include name="**/*.jar"/>
      </fileset>
    </copy>
    <!-- IMPORTANT: make gar.name is an absolute path -->
    <ant antfile="${build.packages}" target="makeGar">
      <property name="gar.name" value="${build.lib}/${gar.filename}.gar"/>
      <property name="garlib.dir" value="${build.lib}"/>
      <property   name="garserverdeployment.file"   value="${package.dir}/server-
deploy.wsdd"/>
      <property name="garschema.origin" value="${build.schema}/${schema.path}"/>
      <property name="garschema.path" value="${schema.path}"/>
    </ant>
  </target>


  <!--
    Misc.
  -->
```

```
  <target name="clean">
    <delete dir="${build.dir}"/>
  </target>

  <target name="all" depends="gar"/>


</project>
```

A.2 Serviço de Aquisição e Controle

A.2.1 server-deploy.sdd

Localização: ~globus/projetos/org/globus/gsc/sac/server-deploy.sdd

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
       xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

       <service name="gsc/sac/ServicoAquisicaoControle" provider="Handler"
                style="wrapped">
              <parameter name="name" value="ServicoAquisicaoControle"/>
              <parameter name="baseClassName"
                         value="org.globus.gsc.sac.HistoricoOperacaoImpl"/>
              <parameter name="className"
                        value="org.globus.gsc.stubs.sac.HistoricoOperacaoType"/>
              <parameter name="schemaPath"
                         value="schema/gsc/sac/ServicoAquisicaoControle.wsdl"/>

              <!-- Start common parameters -->
              <parameter name="allowedMethods" value="*"/>
              <parameter name="persistent" value="true"/>
              <parameter name="handlerClass"
                         value="org.globus.ogsa.handlers.RPCURIProvider"/>
       </service>

</deployment>
```

A.2.2 ServicoAquisicaoControleImpl.java

Localização:

~globus/projetos/org/globus/gsc/sac/impl/ServicoAquisicaoControleImpl.java

A.2.3 ServicoAquisicaoControle.gwsdl

Localização: ~globus/projetos/schema/gcs/sac/ ServicoAquisicaoControle.gwsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="sac"
  targetNamespace="http://www.globus.org/namespaces/2004/02/gsc/sac"
  xmlns:tns="http://www.globus.org/namespaces/2004/02/gsc/sac"

xmlns:data="http://www.globus.org/namespaces/2004/02/gsc/sac/Servi..."
       xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
       xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gt..."
       xmlns:sd="http://www.gridforum.org/namespaces/2003/03/servi..."
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns="http://schemas.xmlsoap.org/wsdl/">

<import location="../../ogsi/ogsi.gwsdl"
       namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
<import location="ServicoAquisicaoControleSDE.xsd"
namespace="http://www.globus.org/namespaces/2004/02/gsc/sac/.../>

<types>
<xsd:schema targetNamespace="http://www.globus.org/namespaces/2004/02/gsc/sac"
       attributeFormDefault="qualified"
       elementFormDefault="qualified"
       xmlns="http://www.w3.org/2001/XMLSchema">

       <!--====================-->
       <!-- Carrega Historico -->
       <!--====================-->
       <xsd:element name="executarControle" type="tns:executarControle">
              <xsd:complexType name="executarControle">
                     <xsd:sequence>
                            <xsd:element name="nomeControle" type="xsd:string"/>
                            <xsd:element name="estado" type="xsd:boolean"/>
                     </xsd:sequence>
              </xsd:complexType>
       </xsd:element>

       <xsd:element name="executarControleaddResponse"
         type="tns:executarControleResponse">
              <xsd:complexType name="executarControleResponse"/>
       </xsd:element>

</xsd:schema>
</types>

<message name="executarControleInputMessage">
  <part name="parameters" element="tns:executarControle"/>
</message>
<message name="executarControleOutputMessage">
  <part name="parameters" element="tns:executarControleResponse"/>
</message>


<gwsdl:portType                          name="ServicoAquisicaoControlePortType"
extends="ogsi:GridService
                ogsi:NotificationSource">
  <operation name="executarControle">
    <input message="tns:executarControleInputMessage"/>
```

```
    <output message="tns:executarControleOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>

  <sd:serviceData name="Identificador"
                  type="data:IdentificadorType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>
  <sd:serviceData name="DadosPlanta"
                  type="data:DadosPlantaType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>
  <sd:serviceData name="Medicao"
                  type="data:MedicaoType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>
 <sd:serviceData name="Controle"
                  type="data:ControleType"
                  minOccurs="1"
                  maxOccurs="unbounded"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>


</gwsdl:portType>
</definitions>
```

A.2.4 ServicoAquisicaoControle.xsd

Localização: ~globus/projetos/schema/gcs/sac/ ServicoAquisicaoControle.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="ServicoAquisicaoControleData"
       targetNamespace="http://www.globus.org/namespaces/2004/02/gsc/sac/.....
       xmlns:tns="http://www.globus.org/namespaces/2004/02/gsc/sac/ServicoAquisi
caoControleSDE"
       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

<wsdl:types>
<schema
targetNamespace="http://www.globus.org/namespaces/2004/02/gsc/sac/ServicoAquisic
aoControleSDE"
```

```
      attributeFormDefault="qualified"
      elementFormDefault="qualified"
      xmlns="http://www.w3.org/2001/XMLSchema">
      <complexType name="Identificador">
            <sequence>
                  <element name="nome" type="string"/>
                  <element name="tipo" type="string"/>
            </sequence>
      </complexType>

      <complexType name="DadosPlanta">
            <sequence>
                  <element name="area" type="string"/>
                  <element name="tipo" type="string"/>
                  <element name="numMedidores" type="int"/>
                  <element name="numcontroles" type="int"/>
                  <element name="enderecoLigacaoSupervisao" type="string"/>
                  <element name="portaLigacaoSupervisao" type="string"/>
                  <element name="numLigacoes" type="int"/>
            </sequence>
      </complexType>

   <complexType name="Medicao">
            <sequence>
                  <element name="medidor" type="string"/>
                  <element name="medida" type="float"/>
            </sequence>
      </complexType>

   <complexType name="Controles">
            <sequence>
                  <element name="controle" type="string"/>
                  <element name="estado" type="boolean"/>
            </sequence>
      </complexType>
```

A.3 Histórico Operação

A.3.1 server-deploy.sdd

Localização: ~globus/projetos/org/globus/gsc/sac/server-deploy.sdd

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
      xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

      <service name="gsc/sac/ServicoAquisicaoControle" provider="Handler"
            style="wrapped">
            <parameter name="name" value="ServicoAquisicaoControle"/>
            <parameter name="baseClassName"
               value="org.globus.gsc.sac.ServicoAquisicaoControleImpl"/>
            <parameter name="className"
               value="org.globus.gsc.stubs.sac.ServicoAquisicaoControleType"/>
            <parameter name="schemaPath"
```

```
                value="schema/gsc/sac/ServicoAquisicaoControle.wsdl"/>

            <!-- Start common parameters -->
            <parameter name="allowedMethods" value="*"/>
            <parameter name="persistent" value="true"/>
            <parameter name="handlerClass"
               value="org.globus.ogsa.handlers.RPCURIProvider"/>
      </service>

</deployment>
```

A.3.2 HistoricoOperacaoImpl.java

Localização: ~globus/projetos/org/globus/gsc/bdh/impl/HistoricoOperacaoImpl.java

A.3.3 HistoricoOperacao.gwsdl

Localização: ~globus/projetos/schema/gcs/bdh/ HistoricoOperacao.gwsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="bdh"
      targetNamespace="http://www.globus.org/namespaces/2004/02/gsc/bdh
"
      xmlns:tns="http://www.globus.org/namespaces/2004/02/gsc/bdh"
      xmlns:data="http://www.globus.org/namespaces/2004/02/gsc/bdh/Hist
ori..."
      xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
      xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridW...
"
      xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData
"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns="http://schemas.xmlsoap.org/wsdl/">

<import location="../../ogsi/ogsi.gwsdl"
      namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
<import location="HistoricoOperacaoSDE.xsd"
      namespace="http://www.globus.org/namespaces/2004/02/gsc/bdh/Histo
ri...."/>

<types>
<xsd:schema
targetNamespace="http://www.globus.org/namespaces/2004/02/gsc/bdh"
      attributeFormDefault="qualified"
      elementFormDefault="qualified"
      xmlns="http://www.w3.org/2001/XMLSchema">

      <!--====================-->
      <!-- Carrega Historico -->
      <!--====================-->
      <xsd:element name="carregaDadosHistoricos"
```

```
type="tns:carregaDadosHistoricos">
            <xsd:complexType name="carregaDadosHistoricos">
                  <xsd:sequence>
                        <xsd:element name="nome" type="xsd:string"/>
                        <xsd:element name="host" type="xsd:string"/>
                        <xsd:element name="porta" type="xsd:string"/>
                        <xsd:element name="arquivo" type="xsd:string"/>
                  </xsd:sequence>
            </xsd:complexType>
      </xsd:element>

      <xsd:element name="carregaDadosHistoricosaddResponse"
         type="tns:carregaDadosHistoricosResponse">
            <xsd:complexType name="carregaDadosHistoricosResponse"/>
      </xsd:element>

</xsd:schema>
</types>

<message name="CarregaDadosHistoricosInputMessage">
  <part name="parameters" element="tns:carregaDadosHistoricos"/>
</message>
<message name="CarregaDadosHistoricosOutputMessage">
  <part name="parameters"
element="tns:carregaDadosHistoricosResponse"/>
</message>


<gwsdl:portType name="HistoricoOperacaoPortType"
extends="ogsi:GridService
       ogsi:NotificationSource">
  <operation name="carregaDadosHistoricos">
    <input message="tns:CarregaDadosHistoricosInputMessage"/>
    <output message="tns:CarregaDadosHistoricosOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>

  <sd:serviceData name="Identificador"
                  type="data:IdentificadorType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>
  <sd:serviceData name="DadosServidorHistorico"
                  type="data:DadosServidorHistoricoType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>
  <sd:serviceData name="EstadoServidor"
                  type="data:EstadoServidorType"
                  minOccurs="1"
                  maxOccurs="1"
```

```
                    mutability="mutable"
                    modifiable="false"
                    nillable="false">
  </sd:serviceData>
 <sd:serviceData name="RegistroCliente"
                    type="data:RegistroClienteType"
                    minOccurs="1"
                    maxOccurs="unbounded"
                    mutability="mutable"
                    modifiable="false"
                    nillable="false">
  </sd:serviceData>


</gwsdl:portType>
</definitions>
```

A.3.4 HistoricoOperacao.xsd

Localização: ~globus/projetos/schema/gcs/bdh/ HistoricoOperacao.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="HistoricoOperacaoData"
       targetNamespace=
          "http://www.globus.org/namespaces/2004/02/gsc/Hist...."
       xmlns:tns="http://www.globus.org/namespaces/2004/02/...."
       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

<wsdl:types>
<schema targetNamespace="http://www.globus.org/names....."
       attributeFormDefault="qualified"
       elementFormDefault="qualified"
       xmlns="http://www.w3.org/2001/XMLSchema">

       <complexType name="Identificador">
              <sequence>
                     <element name="nome" type="string"/>
                     <element name="tipo" type="string"/>
              </sequence>
       </complexType>

       <complexType name="DadosServidorHistorico">
              <sequence>
                     <element name="tipoBancoDados" type="string"/>
                     <element name="capacidadeArmazenamento" type="int"/>
                     <element name="maxClientes" type="int"/>
              </sequence>
       </complexType>

    <complexType name="EstadoServidor">
              <sequence>
                     <element name="numClientes" type="int"/>
                     <element name="espacoUtilizado" type="float"/>
                     <element name="percentualLivre" type="float/>
              </sequence>
       </complexType>
```

```
   <complexType name="RegistroCliente">
            <sequence>
                  <element name="nomeCliente" type="string"/>
                  <element name="totalRegistros" type="int"/>
                  <element name="espacoUtilizado" type="float"/>
            </sequence>
      </complexType>



</schema>
</wsdl:types>

</wsdl:definitions>
```

A.4 Servidor Aplicacão

A.4.1 server-deploy.sdd

Localização: ~globus/projetos/org/globus/gsc/sap/server-deploy.sdd

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig"
xmlns="http://xml.apache.org/axis/wsdd/"
      xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

      <service name="gsc/sap/ServidorAplicacao" provider="Handler"
         style="wrapped">
            <parameter name="name" value="ServidorAplicacao"/>
            <parameter name="baseClassName"
                value="org.globus.gsc.sap.ServidorAplicacaoImpl"/>
            <parameter name="className"

value="org.globus.gsc.stubs.sap.ServidorAplicacaoType"/>
            <parameter name="schemaPath"
                value="schema/gsc/sap/ServidorAplicacao.wsdl"/>

            <!-- Start common parameters -->
            <parameter name="allowedMethods" value="*"/>
            <parameter name="persistent" value="true"/>
            <parameter name="handlerClass"
                value="org.globus.ogsa.handlers.RPCURIProvider"/>
      </service>

</deployment>
```

A.4.2 ServidorAplicacaoImpl.java

Localização: ~globus/projetos/org/globus/gsc/sap/impl/ServidorAplicacaoImpl.java

A.4.3 ServidorAplicacao.gwsdl

Localização: ~globus/projetos/schema/gcs/sap/ ServidorAplicacao.gwsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="sap"
       targetNamespace="http://www.globus.org/namespaces/2004/02/gsc/sap"
       xmlns:tns="http://www.globus.org/namespaces/2004/02/gsc/sap"
       xmlns:data="http://www.globus.org/namespaces/2004/02/gsc/sap/...
       xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
       xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03...
       xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns="http://schemas.xmlsoap.org/wsdl/">

<import location="../../ogsi/ogsi.gwsdl"
       namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
<import location="ServidorAplicacaoSDE.xsd"
       namespace="http://www.globus.org/namespaces/2004/02/gsc/sap...

<types>
<xsd:schema targetNamespace="http://www.globus.org/namespaces/2004/02/gsc/sap"
       attributeFormDefault="qualified"
       elementFormDefault="qualified"
       xmlns="http://www.w3.org/2001/XMLSchema">

       <!--====================-->
       <!-- Carrega Historico -->
       <!--====================-->
       <xsd:element name="somarParcelas" type="tns:somarParcelas">
              <xsd:complexType name="somarParcelas">
                     <xsd:sequence>
                            <xsd:element name="nome" type="xsd:string"/>
                            <xsd:element name="parcelas" type="xsd:double"
                                   minOccurs="1" maxOccurs="unbounded"/>
                     </xsd:sequence>
              </xsd:complexType>
       </xsd:element>

       <xsd:element name="somarParcelasaddResponse"
          type="tns:somarParcelasResponse">
              <xsd:complexType name="somarParcelasResponse"/>
       </xsd:element>

</xsd:schema>
</types>

<message name="somarParcelasInputMessage">
  <part name="parameters" element="tns:somarParcelas"/>
</message>
<message name="somarParcelasOutputMessage">
  <part name="parameters" element="tns:somarParcelasResponse"/>
</message>
```

```
<gwsdl:portType name="ServidorAplicacaoPortType" extends="ogsi:GridService
               ogsi:NotificationSource">
  <operation name="somarParcelas">
    <input message="tns:somarParcelasInputMessage"/>
    <output message="tns:somarParcelasOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>

  <sd:serviceData name="Identificador"
                  type="data:IdentificadorType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>
  <sd:serviceData name="DadosServidorHistorico"
                  type="data:IdentificadorType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>
  <sd:serviceData name="EstadoServidor"
                  type="data:IdentificadorType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>
 <sd:serviceData name="RegistroCliente"
                  type="data:IdentificadorType"
                  minOccurs="1"
                  maxOccurs="unbounded"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>


</gwsdl:portType>
</definitions>
```

A.4.4 ServidorAplicacao.xsd

Localização: ~globus/projetos/schema/gcs/sap/ ServidorAplicacao.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="ServidorAplicacaoData"
       targetNamespace="http://www.globus.org/namespaces/2004/...
       xmlns:tns="http://www.globus.org/namespaces/2004/02/gsc/....
       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

<wsdl:types>
<schema targetNamespace="http://www.globus.org/namespaces/2004/02/gsc/sap...
       attributeFormDefault="qualified"
       elementFormDefault="qualified"
       xmlns="http://www.w3.org/2001/XMLSchema">

       <complexType name="Identificador">
              <sequence>
                     <element name="nome" type="string"/>
                     <element name="tipo" type="string"/>
              </sequence>
       </complexType>

       <complexType name="DadosServidorAplicacao">
              <sequence>
                     <element name="tipoAplicacao" type="string"/>
                     <element name="custoParcela" type="float"/>
                     <element name="maxClientes" type="int"/>
              </sequence>
       </complexType>

    <complexType name="EstadoServidor">
              <sequence>
                     <element name="numeroClientes" type="int"/>
                     <element name="totalOperacoes" type="int"/>
              </sequence>
       </complexType>

    <complexType name="RegistroClientes">
              <sequence>
                     <element name="nomeCliente" type="string"/>
                     <element name="totalGasto" type="boolean"/>
              </sequence>
       </complexType>




</schema>
</wsdl:types>

</wsdl:definitions>
```

A.5 Centro de Controle

A.5.1 CentroControle.java

Localização: ~globus/projetos/org/globus/gsc/ihm/CentroControle.java