Plataforma de Supervisão de Alarmes Baseada em CORBA

Fábio Studyny Higa
fabio.higa@siemens.com.br

Rafael Romualdo Wandresen
rafaelw@ppgia.pucpr.br

Manoel Camillo Penna
penna@ppgia.pucpr.br

PUC-PR - Pontifícia Universidade Católica do Paraná
Laboratórios de Engenharia Elétrica e Informática
R. Imaculada Conceição, 1155 - Prado Velho - Curitiba - Paraná.
CEP 80215-901

Resumo

Este trabalho tem como objetivo a descrição de uma plataforma de supervisão de alarmes baseada em CORBA. Dois aspectos principais são investigados: o mapeamento lógico dos alarmes de telecomunicações em uma estrutura de eventos bem definida, compatível com a arquitetura, assim como, a sua distribuição através dos mecanismos propostos nos serviços CORBA, com o uso dos Canais de Evento e de Notificação. O trabalho também descreve os experimentos realizados para validar as idéias propostas, o que inclui a implementação de um componente de adaptação, assim como, os testes realizados para comprovar a adequabilidade do Canal de Evento, como mecanismo básico para a propagação de alarmes.

Abstract

The objective of this work is the description of a platform for alarm supervision, based on CORBA. Two main aspects are investigated: the logic mapping of telecommunication alarms into a well defined structure, compatible with the architecture and, as well, their distribution by means of the mechanisms introduced by the CORBA Services, with the use of the Event and of the Notification Channels. This work also describes the experiments performed for validation of the proposed ideas, which includes the implementation of an adaption component and, as well, the tests performed for validation of the Event Channel, acting as the basic mechanism used for alarm propagation.

1 Introdução

As operadoras de telecomunicações têm intensificado esforços para reduzir os custos de desenvolvimento e manutenção dos sistemas de gerenciamento. Uma das estratégias importantes neste sentido é a padronização de interfaces entre estes sistemas, para permitir a integração entre soluções de várias empresas. A tecnologia de objetos distribuídos é considerada como uma das melhores opções para se integrar componentes de software distribuídos. Dentre estas tecnologias, a tecnologia CORBA (*Common Object Request Broker Architecture)* definida pelo OMG *(Object Management Group)* [1] tem se destacado como tecnologia para integração. Por ser uma tecnologia flexível, a CORBA pode ser aplicada em praticamente qualquer domínio, e a investigação da sua utilização na área de gerência de redes de telecomunicações tem despertado grande interesse.

Este trabalho propõe uma plataforma de supervisão de alarmes baseada na tecnologia CORBA, e discute a implementação de dois de seus componentes mais importantes, o Adaptador de Objetos e o Distribuidor de Eventos.

2 Plataforma de Supervisão de Alarmes baseada em CORBA

A plataforma de supervisão de alarmes apresentada nesta seção foi concebida a partir do levantamento de funcionalidades das plataformas de gerência (Digital-TeMIP e HP

OEMF) [2], a partir das quais, levantou-se os requisitos fundamentais.

Para descrever a sua arquitetura, introduziu-se o conceito de bloco construtivo, como a sua unidade básica funcional. Um bloco construtivo encapsula um conjunto de funcionalidades relacionadas e as disponibiliza através de uma interface bem definida. Como a premissa fundamental deste trabalho é a utilização de CORBA como arquitetura de distribuição, a interface de um bloco construtivo deve estar definida em IDL. Uma conseqüência deste princípio, é que se o bloco for implementado com uma ferramenta que segue os seus padrões de implementação, ele estará automaticamente disponível para uso através de um ORB *(Object Request Broker)* [1].

Para fins de classificação, identificou-se a existência de duas categorias de blocos construtivos na plataforma: bloco construtivo de propósito geral e bloco construtivo de propósito específico. Um bloco construtivo de propósito geral disponibiliza funcionalidades que podem ser compartilhadas por qualquer componente da plataforma. Esses blocos são básicos na construção de sistemas distribuídos, e os objetos de serviço *CORBAServices* [3] pertencem a esta categoria. O bloco construtivo de propósito específico tem funcionalidades próprias para um determinado domínio como, por exemplo, telecomunicações. Por exemplo, as Interfaces de Domínio *(Domain Interfaces)*[1]definidas pela OMG pertencem a esta categoria. Este trabalho sugere que as funcionalidades propostas pela TMN [4] sejam disponibilizadas por blocos construtivos de propósito específico do domínio de telecomunicações.

O escopo de abordagem deste trabalho é direcionado para as funções de supervisão de falhas. A figura 1 apresenta os componentes arquiteturais da plataforma de supervisão de falhas proposta, classificando-os como propósito geral ou específico.

figura 1: Plataforma para Supervisão de Alarmes

A função do Serviço de Nomes é permitir que um nome possa ser atribuído a qualquer componente, permitindo sua identificação, e localização. O Serviço de *Log* disponibiliza repositórios para o armazenamento dos eventos significativos da rede de telecomunicações [5]. A função do bloco construtivo Correlação e Filtragem de Alarmes é de diminuir a quantidade de alarmes que chegam às aplicações de gerência.

O Adaptador de Objetos, tem por função adaptar componentes não CORBA à plataforma de supervisão de alarmes. Ele é o responsável pela coleta de alarmes de sistemas implementados em outras tecnologias, e pela sua inserção na plataforma de supervisão através do Serviço de Distribuição de Eventos. O bloco construtivo Distribuição de Eventos é o responsável pela propagação de alarmes na plataforma. O Serviço de Eventos e o Serviço de Notificação definidos pelo OMG [1] são exemplos para sua implementação.

A principal contribuição deste trabalho é o detalhamento da concepção e implementação de um Adaptador de Objetos (seção 3), e a descrição de sua integração na

plataforma de supervisão de alarmes através de um Distribuidor de Eventos implementado por um Canal de Evento (ver seção 4).

3 Adaptador de Objetos

No processo de implantação de uma nova plataforma de gerenciamento, surge o problema de como integrar as arquiteturas de gerência existentes (e.g., SNMP, CMIP, RPC e protocolos proprietários), na nova plataforma. Este fato não é diferente com a plataforma de supervisão de alarmes baseada em CORBA. Por esse motivo foi incluído um componente capaz de adaptar ou mapear protocolos e modelos de informação para CORBA. O módulo proposto para resolver este problema denomina-se Ádaptador de Objetos, e sua função é a mediação, provendo acesso e adaptação aos recursos gerenciados definidos em outras plataformas. A figura *1* apresenta um exemplo de várias tecnologias sendo integradas a um ORB através do Ádaptador de Objetos.

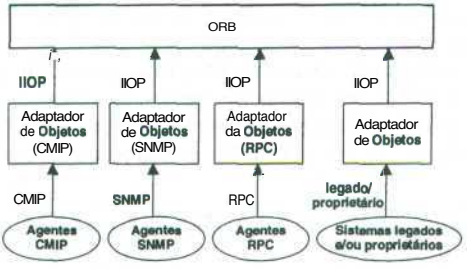

figura *2:* Adaptador de Blocos

O problema de adaptação de objetos já foi investigado em outros contextos. O *TeleManagementFórum* (antigo *Network Management Fórum)* e o *The Open Group* criaram um grupo de trabalho denominado JIDM *(Joint Inter-domain Management)* [6], cujo objetivo principal é prover ferramentas que permitam a integração entre sistemas de gerenciamento baseados em diferentes tecnologias. Este grupo tem trabalhado especialmente na interoperabilidade entre CMIP/CORBA e SNMP/CORBA. Em sua especificação eles definem como um *gateway*, a entidade responsável pela adaptação entre protocolos. Este *gateway é* um exemplo de Adaptador de Objetos.

O Adaptador de Objetos proposto neste trabalho difere entretanto da proposta do JIDM. A diferença fundamental é que não se pretende construir uma ferramenta automática para o processo de adaptação, mas sim definir um *framework* genérico para facilitar a construção de adaptadores.

Na plataforma de supervisão de alarmes, o Adaptador de Objetos é o responsável pela emissão de alarmes no formato padronizado. Ao receber alarmes oriundos de sistemas implementados em tecnologias diferentes, ele os formata e utiliza o Distribuidor de Eventos, para propagá-los para os demais componentes.

3.1 Concepção

Neste trabalho, a função de distribuição de eventos é realizada pelo Canal de Evento definido na especificação do Serviço de Eventos do OMG [1]. Um alarme de um Sistema de Gerenciamento Proprietário (S.G.P.), que chega ao Adaptador de Objetos (em um formato proprietário) é inserido em uma fila interna, mapeado para um evento estruturado (descrito na seção 4), sendo então encapsulado em um evento genérico (formato *any*), e encaminhado ao

Canal de Evento. A figura 3 ilustra esta seqüência.

figura 3: Adaptador de Objetos

O Adaptador de Objetos é composto de três módulos, um módulo de inicialização dos processos, um para conexão com o sistema de gerência não aderente à CORBA, e o outro para conexão com o Canal de Evento.

O processo de inicialização (figura 4) é responsável por iniciar o processo de Comunicação com o Canal de Evento (figura 6) e os processos de Comunicação com o Sistema Proprietário (figura 5).

Os processos de comunicação com o sistema proprietário são iniciados sob demanda, ou seja, o processo de inicialização aguarda um pedido de conexão para criar um novo processo, como ilustra a máquina de estados da figura 4. Este processo (figura 5) fica em um laço infinito de recepção de alarmes, e escrita na fila interna de mensagens.

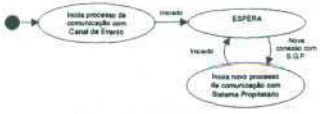

figura 4: Inicialização

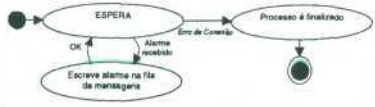

figura 5: Comunicação com o Sistema Proprietário

O processo de comunicação com o Canal de Evento (figura 6) lê os alarmes da fila de mensagens e os envia para o Canal de Evento. Esta estrutura permite independência entre os processos de comunicação com o sistema proprietário e de comunicação com o Canal de Evento. Se o Canal de Evento se desconectar, o Adaptador de Objetos não perderá alarmes, até um certo limite, que depende do tamanho da fila de mensagens interna. Este tamanho deve ser definido conforme o fluxo médio de alarmes oriundos dos equipamentos gerenciados.

figura 6: Comunicação com Canal de Evento

3.2 Implementação

Esta seção descreve uma implementação[7] do Adaptador de Objetos, realizada para integrar os alarmes da rede de gerenciamento EMOS-T da SIEMENS. Esta implementação permite várias conexões com a rede de gerenciamento proprietária, sendo que a comunicação entre o EMOS-T e o Adaptador de Objetos é feita via *socket*, através de uma protocolo definido pela SIEMENS, conforme ilustrado na figura 7.

O Adaptador de Objetos foi implementado na linguagem C++, utilizando o compilador Microsoft Visual C++ 6.0. Para interconexão com o EMOS-T foi utilizada uma biblioteca fornecida pela SIEMENS. A interconexão com o Canal de Evento foi implementada para dois ORBs distintos (TAO e ORBacus), de modo a possibilitar testes de interoperabilidade e de comparação de desempenho. Os ORBs TAO e ORBacus foram selecionados, porque têm sido usados com sucesso em diversos projetos relevantes, e porque têm o código aberto (fonte disponível) [8][9].

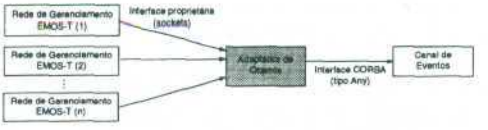

figura 7: Implementação do Adaptador de Objetos

4 Distribuição de Eventos

A distribuição de eventos é uma função fundamental de uma plataforma de supervisão de alarmes, pois um alarme contém a mensagem de falha em tempo real. Em sistemas de software, os alarmes são implementados por mensagens assíncronas denominadas eventos, cujo fluxo e o formato em um sistema de supervisão de falhas, são de especial relevância. Nesta seção discutimos a operacionalização da distribuição de eventos através do Canal de Evento ou de Notificação. Apresentamos uma proposição para o mapeamento do Relatório de Alarme definido pelo ITU-T, para o modelo de Evento Estruturado proposto pelo OMG no Serviço de Notificação.

4.1 Canal de Evento e de Notificação

A comunicação entre os componentes de software realizada através de um ORB é baseada no modelo de comunicação do tipo cliente-servidor [10]. Este tipo de comunicação é amplamente utilizado e é uma solução natural para a comunicação entre objetos. Este modelo exige que os parceiros estejam simultaneamente em execução (acoplamento forte). Entretanto, existem situações onde o acoplamento forte não é o modelo de interação mais adequado. Um exemplo típico de interação onde a comunicação não acoplada entre objetos é melhor, é a emissão de alarmes. Um objeto deve poder emitir espontaneamente um alarme, sem se preocupar se o objeto de destino está executando no momento da emissão.

Para suportar a comunicação assíncrona (não acoplada) entre objetos na plataforma de supervisão de alarmes, foi definido o bloco construtivo Distribuição de Eventos. Os requisitos básicos que este bloco deve atender são os seguintes: múltiplos objetos podem emitir eventos sem o conhecimento dos objetos que desejam receber estes eventos; e múltiplos objetos podem receber eventos sem o conhecimento dos objetos que emitem estes eventos. A figura 8 ilustra o conceito de distribuição de eventos.

figura 8: Distribuição de Eventos

Devido a importância da existência de um componente que permita a comunicação não acoplada entre objetos, o OMG definiu o Serviço de Evento e o Serviço de Notificação.

Estes dois serviços são exemplos do bloco construtivo de Distribuição de Eventos. Por serem padrões estabelecidos, e por já existirem implementações comerciais, o uso deles na plataforma de supervisão de alarmes é encorajado.

O Canal de Evento atende aos requisitos básicos do bloco de Distribuição de Eventos. O Canal de Notificação estende as funcionalidades do Canal de Evento, adicionando características tais como:

- habilidade de transmissão de eventos na forma de estruturas bem definidas;
- habilidade de clientes especificarem quais eventos desejam receber através de filtros;
- habilidade de fornecedores descobrirem quais tipos de eventos os consumidores estão interessados em receber podendo, desta forma, produzir eventos sob demanda;
- habilidade de consumidores descobrirem a existência de um novo tipo de evento oferecido no Canal de Evento e de se registrarem para o recebimento deste;
- habilidade de se configurar propriedades de qualidade de serviço, permitindo o estabelecimento de regras com relação a garantia de entrega, envelhecimento de eventos e prioridade de eventos; e
- existência de um repositório de tipo de eventos que disponibiliza informações sobre as estruturas de eventos disponíveis.

Devido a estas funcionalidades, o Canal de Notificação, é a melhor opção para a implementação do bloco de Distribuição de Eventos.Apesar de um Canal de Notificação se apresentar como o mecanismo ideal para a propagação de eventos em uma plataforma de supervisão de alarmes baseada em CORBA, o Canal de Evento também fornece a estrutura de comunicação adequada. A sua maior deficiência é entretanto o tratamento de eventos estruturados, não tendo sido identificada nenhuma implementação que incluísse os eventos estruturados.

Independentemente da implementação escolhida, o conceito de Evento Estruturado definido no Serviço de Notificação do OMG [1], apresenta-se como uma solução importante para o mapeamento de alarmes de telecomunicações em sistemas de gerência baseados em CORBA. A seguir é apresentada uma solução para o mapeamento da sintaxe utilizada do Relatório de Alarme definido pelo ITU-T, para o modelo de Evento Estruturado do Serviço de Notificação.

4.2 Eventos estruturados (*Structured events*)

O evento estruturado é um conceito definido no Serviço de Notificação, que possibilita a transmissão de eventos com estrutura de dados bem definida através de uma interface escrita em IDL. A grande vantagem do Evento Estruturado é que ele permite a adição de funcionalidades importantes, tais como filtragem e subscrição de eventos. O formato do Evento Estruturado é apresentado na figura 9.

O cabeçalho do Evento Estruturado contém uma porção denominada *fixedjieadpr* (cabeçalho fixo). Esta parte do cabeçalho é composta pelos seguintes campos:

- *domain_type* (tipo do domínio): este campo contém uma seqüência de caracteres que identifica o domínio do qual um evento faz parte. Como exemplo pode-se citar os domínios de Telecomunicações, Finanças e Saúde. A definição deste campo permite que cada domínio possua seu conjunto próprio de tipos de eventos sem haver a preocupação de que sejam definidos tipos de eventos com o mesmo nome em outros domínios.
- *type_name* (nome do tipo): este campo contém uma seqüência de caracteres que indica o tipo de evento contido dentro do evento estruturado. Este nome deve ser único dentro do conjunto de tipos de eventos definidos em um domínio.
- *event_name* (nome do evento): este campo contém uma seqüência de caracteres que

define o nome de uma determinada instância do evento estruturado.

O restante do cabeçalho de um evento estruturado está contido dentro do campo *variable_header* (cabeçalho variável). O tipo deste campo é uma seqüência de estruturas contendo os campos nome e valor. O campo nome é do tipo seqüência de caracteres enquanto o campo valor é do tipo *CORBA::Any*. O campo *variablejieader* pode conter qualquer estrutura nome-valor que o usuário julgue ser necessária.

O corpo do evento estruturado tem o propósito de conter os dados do evento, e é dividido em dois campos, *filterable_data* e *remainder_of_body*.O campo *filterable_data é uma* seqüência de estruturas nome-valor, onde o campo nome é do tipo seqüência de caracteres enquanto o campo valor é do tipo CORBA::Any. A finalidade deste campo é fornecer uma estrutura conveniente para o processo de filtragem. A finalidade do campo *remainder_of_body* é conter dados do evento para os quais não se deseje aplicar filtragem. O tipo deste campo é *CORBA::Any*.

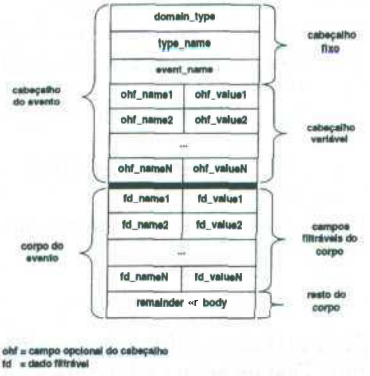

figura 9: Evento estruturado

4.3 Mapeamento de Relatórios de Alarme para Eventos Estruturados

Tanto o Serviço de Evento como o Serviço de Notificação foram projetados para serem utilizados por qualquer objeto CORBA em aplicações pertencentes a qualquer domínio. Desta forma, os tipos de eventos definidos para estes serviços (tipados, não-tipados e estruturados) podem conter qualquer tipo de informação. Desse fato surge a necessidade de se utilizar uma estrutura apropriada para o envio de alarmes pertencentes ao domínio das telecomunicações. Por isso, nesta seção é apresentada uma proposta de mapeamento dos alarmes definidos pela ITU-T para estruturas CORBA compatíveis com o Evento Estruturado.

Alarmes são notificações de eventos ocorridos em um sistema como, por exemplo, detecção de falhas e condições anormais. Um alarme pode ou não representar um erro. Um relatório de alarme é um tipo específico de relatório de eventos que traz informações a respeito do alarme. Assim, em um sistema de supervisão, é importante se utilizar uma estrutura apropriada para os alarmes, pois isto promove a interoperabilidade entre diferentes sistemas.

4.3.1 Serviço M-EVENT-REPORT

A recomendação X.710 [11] define o serviço M-EVENT-REPORT, cuja finalidade é informar a um usuário a ocorrência de um evento em um objeto gerenciado. No serviço M-EVENT-REPORT são definidos os seguintes parâmetros:

- Identificador de invocação (*Invoke identifier*):este parâmetro é utilizado para identificar uma determinada notificação. Sua presença é obrigatória na requisição e na resposta.
- Modo (*Mode*): este parâmetro indica se a operação deve ser confirmada ou não. Sua presença é obrigatória na requisição.
- Classe do objeto gerenciado (*Managed object class*): este parâmetro indica a classe do objeto gerenciado em que ocorreu o evento. Sua presença é obrigatória na requisição e depende da necessidade do usuário na resposta.
- Instância do objeto gerenciado *(Managed object instance)*: este parâmetro indica a instância do objeto gerenciado em que ocorreu o evento. Sua presença é obrigatória na requisição e depende da necessidade do usuário na resposta.
- Tipo do evento *(Event type)*: este parâmetro indica o tipo de evento. Sua presença é obrigatória na requisição e condicional na resposta.
- Instante do evento *(Event time):* este parâmetro indica o momento de geração do evento. Sua presença depende da necessidade do usuário na requisição.
- Informação do evento *(Event information)*: este parâmetro contém informação que se deseja passar. Sua presença depende da necessidade do usuário na requisição.
- Tempo corrente (*Current time):* este parâmetro indica o momento em que a resposta a um relatório de eventos foi gerada. Sua presença depende da necessidade do usuário na resposta.
- Resposta para o evento *(Event reply)*: este parâmetro contém a resposta ao relatório de evento enviado. Sua presença é condicional na resposta.
- Erros (*Errors*): este parâmetro é enviado na resposta indicando um erro na operação. Sua presença é condicional na resposta.

4.3.2 Relatório de Alarme

O conjunto de notificações e parâmetros definidos na recomendação X.733 [12] são usados para detalhar os parâmetros **Tipo do evento e Informação do evento** definidos no serviço M-EVENT-REPORT.

A recomendação X.733 define cinco categorias básicas de alarme, cujos tipos preenchem o parâmetro **Tipo do evento** do relatório de alarme. Estas categorias são:

- Alarme de Comunicação *(Communications alarm)*: um alarme deste tipo está relacionado aos procedimentos e/ou processos necessários para a transferência de informação.
- Alarme de Qualidade de Serviço (*Quality of service alarm):* um alarme deste tipo está relacionado à degradação da qualidade de serviço.
- Alarme de Erro de Processamento *(Processing error alarm):* um alarme deste tipo está relacionado às falhas de *software* ou de processamento.
- Alarme de Equipamento (*Equipment alarm):* um alarme deste tipo está relacionado às falhas de equipamento.
- Alarme Ambiental (*Environmental alarm):* um alarme deste tipo está relacionado às condições do ambiente em que se encontra o equipamento.

Os seguintes parâmetros constituem a informação específica de um alarme e são atribuídos ao parâmetro **Informação do evento:**

- Causa provável *(Probable cause):* este parâmetro define a causa provável do alarme. As normas ISO enumeram várias causas prováveis, como por exemplo perda de sinal, erro de enquadramento, erro de transmissão local, entre outros. A presença deste parâmetro no relatório de alarme é obrigatória.
- Problemas específicos *(Specific problems)*: este parâmetro fornece informações mais detalhadas sobre a causa provável do alarme. A presença deste parâmetro no relatório de

alarme depende da necessidade do usuário.

- Nível de severidade *(Perceived severity)*: este parâmetro, cuja presença é obrigatória, define seis níveis de severidade de alarme que indicam como a capacidade de funcionamento de um objeto gerenciado foi afetada. Quatro destes níveis são ordenados a partir do mais severo até o menos severo, são eles: Crítico, Maior, Menor e Atenção. Os outros dois níveis, limpo e indeterminado, não correspondem a uma ordem de severidade. Os níveis definidos para uso com este parâmetro obrigatório são:
  - limpo *(cleared)*: este nível indica a remoção de um ou um conjunto de alarmes anteriormente gerados. O conjunto de alarmes está associado a um mesmo objeto gerenciado e possuem o mesmo valor para os parâmetros Tipo de evento, Causa provável e Problemas específicos. Muitas notificações associadas podem ser removidas utilizando-se o parâmetro Notificações correlacionadas.
  - indeterminado *(indeterminate)*: este nível indica que não se pode determinar o quanto a capacidade do objeto gerenciado foi afetada.
  - crítico *(criticai)*: este nível indica que uma condição que afeta o serviço ocorreu e que uma ação corretiva imediata é requerida.
  - maior *(major)*: este nível indica que uma condição que afeta o serviço se desenvolveu e que uma ação corretiva urgente é requerida.
  - menor *(minor)*: este nível indica a existência de falhas que afetam o serviço e que uma ação corretiva deve ser tomada para se prevenir a ocorrência de falhas mais sérias.
  - atenção *(warning)*: este nível indica a detecção de uma potencial falha
- Estado de *back-up (Backed-upstatus)*: este parâmetro especifica se foi feito o *back-up* do objeto que emitiu o alarme. A presença deste parâmetro no relatório de alarme depende da necessidade do usuário.
- Objeto *back-up (Back-up object)*: este parâmetro deve estar presente no relatório de alarme quando o parâmetro Estado de *backed-up* estiver presente e tiver o seu valor igual a *true*. Este parâmetro indica a instância do objeto gerenciado que deve ser usada como *back-up* do objeto gerenciado que emitiu o alarme.
- Indicação de tendência *(Trend indication)*: este parâmetro, cuja presença depende da necessidade do usuário, é utilizado para indicar a tendência de severidade de um alarme com relação a outros alarmes pendentes *(outstanding alarms)*, que ainda não foram tratados *(cleared)*. Os possíveis valores para este parâmetro são:
  - mais severo *(more severe)*: o nível de severidade deste alarme é maior do que dos alarmes pendentes anteriormente notificados;
  - nenhuma mudança *(no change)*: o nível de severidade deste alarme é igual ao dos alarmes pendentes com maior nível de severidade; e
  - menos severo *(less severe)*: indica que há pelo menos um alarme pendente com nível de severidade superior ao deste alarme.
- Informação de valor-limite *(Threshold information)*: este parâmetro deve estar presente quando o alarme é resultado da transposição de um valor-limite *(threshold)*. Ele consiste de quatro sub-parâmetros:
  - limiar de disparo *(triggered threshold)*: o identificador do atributo de valor-limite que causou a notificação;
  - nível-limite *(threshold levei)*: valor-limite que quando ultrapassado gera a emissão de uma notificação;
  - valor observado *(observed value)*: valor observado que ultrapassou o valor-limite; e
  - instante de transposição *(arm time)*: instante de ocorrência da ultrapassagem do valor-limite.

- Identificador de notificação (*Notification identifier*): este parâmetro é utilizado para identificar uma determinada notificação. Este parâmetro deve ser único entre todas as outras notificações do objeto gerenciado. A presença deste parâmetro no relatório de alarme depende da necessidade do usuário.
- Notificações correlacionadas (*Correlated notifications*): este parâmetro contém um conjunto de identificadores de notificação e, se necessário, os nomes das instâncias dos objetos gerenciados relacionados a estas notificações. Este conjunto indica todas as notificações com as quais esta notificação está correlacionada. A presença deste parâmetro no relatório de alarme depende da necessidade do usuário.
- Definição de mudança de estado (*State change definition*):este parâmetro é usado para indicar uma transição de estado associado ao alarme. A presença deste parâmetro no relatório de alarme depende da necessidade do usuário.
- Atributos monitorados (*Monitoreaattributes*): este parâmetro define um ou mais atributos de um objeto gerenciado que devem ter seus valores monitorados. A presença deste parâmetro no relatório de alarme depende da necessidade do usuário.
- Ações de reparo propostas (*Proposed repair actions*): este parâmetro é utilizado para sugerir soluções para um problema cuja causa é conhecida. A presença deste parâmetro no relatório de alarme depende da necessidade do usuário.
- Texto adicional (*Additional text*): este parâmetro permite que um texto possa ser adicionado à notificação. A presença deste parâmetro no relatório de alarme depende da necessidade do usuário.
- Informação adicional (*Addtional information*): este parâmetro permite a inclusão no relatório de evento de um conjunto de estruturas de dados contendo informações. Cada estrutura de dados contém três itens: um identificador, um indicador de significado e a informação do problema. A presença deste parâmetro no relatório de alarme depende da necessidade do usuário.

4.3.3 Mapeamento

Nesta seção é apresentada uma proposta de mapeamento entre os relatórios de alarme especificados na recomendação X.733 e os eventos estruturados especificados no Serviço de Notificação. Nesta proposta, cada parâmetro de alarme especificado na recomendação X.733 e que preenche o parâmetro **Informação do evento** do serviço M-EVENT-REPORT é mapeado para uma estrutura nome-valor da porção de campos filtráveis do Evento Estruturado. O campo nome é preenchido com o nome do atributo, enquanto o campo valor é preenchido com um valor adequado ao tipo de atributo especificado no campo nome.

Para se realizar o mapeamento, inicialmente deve se preencher os campos de cabeçalho do Evento Estruturado. O primeiro campo, denominado de *domain_name*, deve conter uma seqüência de caracteres indicando o domínio ao qual o alarme pertence. Este campo é preenchido com o valor "Telecom", indicando que o alarme pertence ao domínio das telecomunicações.

Em seguida deve-se preencher o campo *type_name* do cabeçalho, que corresponde ao parâmetro **Tipo do evento** do serviço M-EVENT-REPORT, com uma seqüência de caracteres para indicar o tipo de alarme contido dentro do evento estruturado. De acordo com as categorias de alarme definidas na recomendação X.733, o campo *type_name* deve conter os seguintes valores definidos na Tabela 1.

Tipo do Alarme	*type_name*
Alarme de Comunicação	CommunicationsAlarm
Alarme de Qualidade de Serviço	QualityofServiceAlarm

Alarme de Erro de Processamento	ProcessingErrorAlarm
Alarme de Equipamento	EquipmentAlarm
Alarme Ambiental	EnvironmentalAlarm

Tabela 1: Tipos de alarmes

Após os campos do cabeçalho terem sido preenchidos, parte-se para a inserção dos parâmetros de alarme na porção de campos filtráveis do Evento Estruturado. A porção de campos filtráveis do evento estruturado é uma seqüência de estruturas nome-valor (*name-value*). Como não existe regras definidas para o mapeamento de alarmes em eventos estruturados, optou-se por mapear os parâmetros do alarme em estruturas nome-valor da porção de campos filtráveis. Isto permite uma posterior filtragem de acordo com os valores dos parâmetros do alarme. Para se mapear cada parâmetro do alarme em estruturas nome-valor, deve-se inicialmente preencher o campo nome (*name*) da estrutura nome-valor com o nome do atributo, cujo valor é apresentado na Tabela 2.

Parâmetro	Nome do atributo
Classe do objeto gerenciado	ManagedObjectClass
Instância do objeto gerenciado	ManagedObjectInstance
Instante do evento	EventTime
Causa provável	ProbableCause
Problemas específicos	SpecificProblems
Nível de severidade	PerceivedSeverity
Estado de *back-up*	BackedUpStatus
Objeto *back-up*	BackeupObject
Indicação de tendência	TrendIndication
Informação de valor-limite	ThresholdInfo
Identificador de notificação	NotificationIdentifier
Notificações correlacionadas	CorrelatedNotifications
Definição de mudança de estado	StateChangeDefinition
Atributos monitorados	MonitoredAttributes
Ações de reparo propostas	ProposedRepairActions
Texto adicional	AdditionalText
Informação adicional	AdditionalInformation

Tabela 2: Parâmetros do alarme

Em seguida, o campo valor (*value*) da estrutura nome-valor deve ser preenchido. Este campo valor é do tipo *CORBA::Any* e, portanto, permite a inserção de valores representados em qualquer tipo definido em CORBA ou qualquer tipo definido pelo usuário, através de IDLs.

Neste trabalho define-se apenas os tipos dos atributos de alarme cuja presença é obrigatória ou que foram consideradas importantes para a supervisão de alarmes. Os atributos eventTime e additionalText, cuja presença não é obrigatória, foram definidos pelo fato de serem utilizados nas plataformas de gerência estudadas. A Tabela 3 apresenta o nome do atributo e o tipo em IDL correspondente.

Nome do atributo (campo *name*)	Tipo IDL
ManagedObjectClass (obrigatório)	TelecomNotification::ObjectClassType
ManagedObjectInstance (obrigatório)	TelecomNotification::ObjectInstanceType
EventTime	TelecomNotification::EventTimeType
ProbableCause (obrigatório)	TelecomNotification::ProbableCauseType
PerceivedSeverity (obrigatório)	TelecomNotification::PerceivedSeverityType
AdditionalText	TelecomNotification::AdditionalTextType

Tabela 3: Tipos dos atributos do alarme

A definição dos tipos em IDL é apresentada em seguida:

```
module TelecomNotification {
   typedef string ObjectClassType;
   typedef string ObjectInstanceType;
   typedef string EventTimeType;
   typedef long ProbableCauseType;

   enum PerceivedSeverityType (
      indeterminate,
      criticai,
      major,
      minor,
      warning,
      cleared
   );
   typedef string AdditionalTextType;
);
```

Para ilustrar o resultado do processo de mapeamento, é apresentado um exemplo de Alarme de Comunicação mapeado para evento estruturado:

```
event.header.fixed_header.event_type.domain_name = "Telecom"
event.header.fixed_header.event_type.type_name = "communicationsAlarm"
event.filterable_data[0].name = "managedObjectClass"
event.filterable_data[0].value = "equipment"
event.filterable_data[1].name = "managedObjectInstance"
event.filterable_data[1].value = "equipment01"
event.filterable_data[2].name = "eventTime"
event.filterable_data[2].value = "19991231235959"
event.filterable_data[3].name = "probableCause"
event.filterable_data[3].value = "powerProblem (58)"   // definido na M.3100
event.filterable_data[4].name = "perceivedSeverity"
event.filterable_data[4].value = "criticai (2)"
event.filterable_data[5].name = "additionalText"
event.filterable_data[5].value = "Power supply failure"
```

5 Experimentos Realizados com o Bloco Construtivo Distribuição de Eventos

Com o objetivo de se validar as idéias apresentadas, esta seção apresenta experimentos realizados com a utilização do Canal de Evento como bloco construtivo para Distribuição de Eventos. Optou-se por utilizar o Canal de Evento, e não o Canal de Notificação, devido ao fato de que o primeiro está disponível em diferentes implementações de ORB, o que não ocorre com o segundo. Além disso, em alguns casos, o Canal de Notificação pode oferecer mais funcionalidade do que o necessário para uma aplicação específica. Ao se validar o conceito para o Canal de Evento, que é um subconjunto do Canal de Notificação, o conceito é validado para ambos.

5.1 Testes realizados com o Canal de Evento

Para avaliar a adequabilidade do Canal de Evento em uma aplicação de supervisão de alarmes, foram concebidos inúmeros procedimentos de teste [13] descritos a seguir. Os principais objetivos destes testes foram:

- verificar o funcionamento e a interoperabilidade das aplicações criadas com relação ao Serviço de Eventos dos dois ORBs;
- testar em diferentes situações o desempenho e a confiabilidade dos dois Serviços de Eventos, bem como o consumo de recursos (CPU e memória) para o envio e recepção de eventos; e
- realizar um teste de *throughput* dos dois Serviços de Eventos.

O ambiente utilizado para a realização dos testes tinha a seguinte configuração:

- máquina a: Pentium 200, 80MB de memória, Windows NT 4.0 Workstation.

- máquina b: Pentium II 450, 120MB de memória, Windows NT 4.0 Server.
- máquina c: Pentium 166, 64MB de memória, Windows NT 4.0 Workstation.
- máquina d: Pentium 166, 64MB de memória, Windows NT 4.0 Workstation.
- computadores conectados através de um segmento Ethernet de 10 mbit/s, isolado de outros segmentos.

Para a execução dos testes foram implementadas duas aplicações, uma em C++ utilizando o ORB TAO, com a função de produzir e enviar eventos para o Canal de Evento (fornecedor), e outra implementada em Java utilizando o ORB ORBacus, com a finalidade de receber os eventos do Canal de Evento (consumidor).

5.1.1 Teste de funcionamento e interoperabilidade

O primeiro teste realizado teve como objetivo verificar se haveria problemas de compatibilidade na utilização das duas implementações do Canal de Evento. Este teste consistiu em iniciar o Canal de Evento, o consumidor e o fornecedor de eventos e verificar se estes se comunicavam corretamente. Este procedimento foi repetido para as duas implementações de Canal de Evento, não tendo sido observado nenhum problema de compatibilidade.

5.1.2 Teste de desempenho, confiabilidade e consumo de recursos

O segundo teste teve como objetivo verificar a confiabilidade e o desempenho dos dois Canais de Evento, comparando-os em relação ao consumo de recursos de *hardware* (memória e CPU) e em relação aos tempos de envio e recebimento de eventos em situações diversas, ou seja, modificando-se o número de fornecedores e consumidores de eventos.

Este teste consistiu em iniciar o Canal de Evento e então iniciar quantidades diferentes de fornecedores e consumidores de eventos, sempre monitorando o comportamento das aplicações, bem como o uso dos recursos. Estes procedimentos foram repetidos para as duas implementações do Canal de Evento.

A Tabela 4 mostra a configuração do ambiente em 8 configurações de teste. As letras em maiúsculo indicam a máquina, seguido do número de consumidores ou fornecedores que estava em execução em cada uma delas. Entre parênteses, ao lado do número de fornecedores, é indicado a velocidade com que os eventos eram enviados por eles. Na última linha de cada célula aparece o número de eventos enviados em cada teste.

Com base nos resultados obtidos através destas 8 configurações, dois gráficos foram traçados. O primeiro, apresentado na figura 10, indica o tempo médio de envio de eventos, enquanto que segundo, apresentado na figura 11, indica o tempo médio de recebimento de eventos. Através destes gráficos pode-se avaliar comparativamente o desempenho dos dois Canais de Evento.

1	A: 50 fornecedores. (1 evento/s) B: 1 C.E 1 consumidor. (25.000 eventos)	2	A: 2 fornecedores (máximo) B: 1 CE. 1 consumidor (3.000 eventos)	3	A: 2 fornecedores (máximo) B: 1 C.E. C: 1 consumidor (25.000 eventos)	4	A: 2 fornecedores (máximo) B: 1 C.E. C: 1 consumidor (3.000 eventos)
5	A: 1 fornecedor (máximo) B: 1 CE. C: IO consumidores D: 10 consumidores (3.000 eventos)	6	A: 1 fornecedor (máximo) B: 1 CE C: 3 consumidores D: 3 consumidores (3.000 eventos)	7	A: 30 fornecedores (1 evento/s) B: 1 CE. C: 1 consumidor (9.000 eventos)	8	A: 3 fornecedores (10 evento/s) B: 1CE C: 1 consumidor (9.000 eventos)

Tabela 4: Configuração do ambiente

Ao final dos testes chegou-se aos seguintes resultados qualitativos:

- Em nenhum dos testes realizados com o Canal de Evento do TAO e do ORBacus houve perda de eventos.
- A gerência de memória feita pelo TAO é boa, apesar dele não desalocar toda a memória utilizada.
- Foram identificados problemas com relação ao uso de memória no Canal de Evento do ORBacus. A memória previamente alocada jamais era liberada, mesmo quando esta não estava mais sendo utilizada.
- O tempo de recebimento de um evento por um consumidor no Serviço de Eventos do TAO é menor do que o do ORBacus, para as configurações avaliadas.
- O tempo de envio de um evento por um fornecedor para apenas um consumidor no Canal de Evento do TAO é menor do que o do ORBacus.
- O tempo de envio de um evento por um fornecedor para vários consumidores no Canal de Evento do TAO é maior do que o do ORBacus.
- O Canal de Evento do TAO apresentou um problema grave quando havia vários consumidores de eventos. Se um dos consumidores fica bloqueado, todos os outros param de receber eventos.

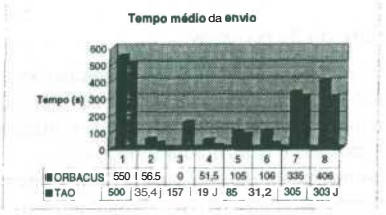

figura 10: Tempo médio de envio de eventos

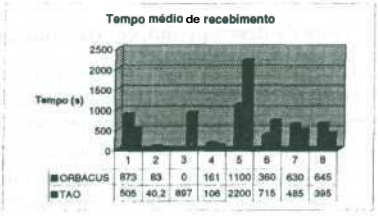

figura 11: Tempo médio de recebimento de eventos

5.1.3 Teste de Capacidade

No terceiro teste, realizado em duas etapas, foi avaliada a capacidade das duas implementações de Canal de Evento. Na primeira etapa, com o intuito de se verificar a capacidade de recebimento de eventos, é medido o tempo médio necessário para que um elemento de um conjunto consumidores receba 500 eventos de um único fornecedor. Na segunda etapa, cujo objetivo era verificar a capacidade de envio de eventos, é medido o tempo médio necessário para que um elemento de um conjunto de fornecedores envie 500 eventos para um único consumidor.

O Canal de Evento foi executado na máquina b. Um conjunto de consumidores ou fornecedores foi executado na máquina a, respectivamente o fornecedor ou o consumidor foi executados na máquina d.

A figura 12 e a figura 13 apresentam respectivamente o tempo médio de recebimento e envio de eventos, em relação ao número de consumidores e fornecedores.

Com a realização dos testes de capacidade, chegou-se aos seguintes resultados:

- O tempo médio de recebimento de eventos para um consumidor aumenta de forma mais expressiva no TAO do que no ORBacus com o aumento do número de consumidores conectados ao Canal de Evento.
- O tempo médio de envio de eventos para um fornecedor é maior e aumenta de forma mais expressiva no ORBacus do que no TAO com o aumento do número de fornecedores conectados ao Canal de Evento.

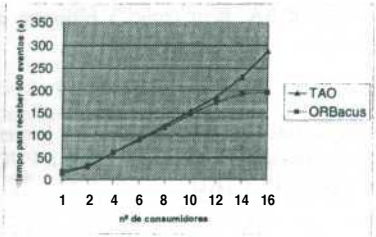

figura 12: Tempo médio de recebimento de eventos versus número de consumidores

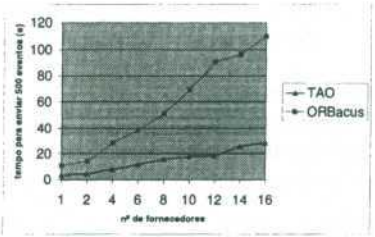

figura 13: Tempo médio de envio de eventos versus número de fornecedores

5.2 Observações Finais

Através dos dados observados nas três categorias de testes realizados pode-se chegar às seguintes conclusões finais:

- não houve problemas de interoperabilidade entre fornecedores e consumidores de eventos e os Canais de Evento disponibilizado pelos dois ORBs;
- o Canal de Evento disponibilizado pelo TAO mostrou-se ineficiente nos testes em que foram utilizados mais de um consumidor. Notou-se que na ocorrência de problemas em qualquer um dos consumidores, como por exemplo no caso da aplicação não se desconectar corretamente do Canal de Evento, o envio de eventos para os outros consumidores é prejudicado. O Canal de Evento do ORBacus não apresentou este tipo de problema;
- com relação ao desempenho, no caso onde se tem apenas um consumidor, a eficiência do Canal de Evento do TAO é superior ao do ORBacus em todos os itens observados. No caso onde há vários consumidores, o Canal de Evento do ORBacus mostra-se superior ao do TAO;
- o Canal de Evento do ORBacus apresentou problemas quando existe um grande fluxo de eventos. Quando há um grande número de eventos sendo transmitidos, a fila interna de eventos existente nesta implementação do Serviço de Eventos chega a um nível de utilização muito elevado, fazendo com que a aplicação pare de funcionar. Este problema foi identificado no teste 3 (Tabela 4).
- O teste de capacidade foi considerado satisfatório para a escala desejada.

6 Conclusão

Este trabalho demonstrou como a arquitetura de distribuição de objetos CORBA, pode ser incorporado à Gerência de Rede de Telecomunicações. Inicialmente, apresentamos uma plataforma para supervisão de alarmes, que foi concebida a partir do estudo de diferentes sistemas de gerência de falhas. A partir da definição de requisitos necessários para a supervisão de alarmes, determinou-se quais serviços CORBA podem ser aplicados e quais novos componentes devem ser criados.

A solução para a distribuição de eventos na plataforma supervisão de alarmes foi alcançada através do estudo da aplicabilidade do Canal de Evento e Canal de Notificação para a distribuição de eventos, e da proposta de mapeamento de relatórios de alarmes definidos pela ITU-T em eventos estruturados definidos no Serviço de Notificação. Outra contribuição relevante, foram os testes de compatibilidade e desempenho de duas implementações do Canal de Evento.

O problema de adaptação dos alarmes para a infra-estrutura proposta também foi abordado, através do bloco construtivo Adaptador de Objetos, que comprovou na prática uma

integração de um protocolo proprietário ao ambiente CORBA, bem como a proposta de mapeamento de relatório de alarmes descrita no item 4.3.3.

7 Agradecimentos

Agradecemos a Siemens pelo suporte dado a este trabalho que faz parte do Projeto GIR (Gerência Integrada de Redes) desenvolvido em cooperação entre a Siemens e a PUC-PR. Este projeto é coordenado na Siemens pelo departamento ICN TRD.

8 Referências Bibliográficas

[1] OBJECT MANAGEMENT GROUP. **A Discussion of the Object Management Architecture**. 1997.

[2] HIGA, Fábio Studyny. **Estudo do uso de CORBA para supervisão de alarmes em sistemas de suporte à operação de telecomunicações.** Dissertação de mestrado, CEFET-PR, Dezembro 1999.

[3] OBJECT MANAGEMENT GROUP. **CORBAservices: Common Object Services Specification**. 1997.

[4] INTERNATIONAL TELECOMMUNICATION UNION. **Overview of** TMN **Recommendations**, M.3000. [S.I.], 1994.

[5] PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ. **Gerência Integrada de Rede, Sub-projeto PUC-PR - Artigo submetido ao SBRC: Serviço Genérico de LOG Estruturado para Gerência de Telecomunicações.** Curitiba, 2000.

[6] OPEN GROUP. **Inter-domain Management: Specification Translation,** P509. 1997.

[7] PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ. **Gerência Integrada de Rede, Sub-projeto PUC-PR - Documento de Análise e Projeto dos Módulos Acesso ao NE e Log Genérico.** Curitiba, 1999.

[8J TAO.http://www.cs.wustl.edu/~schmidt/TAO.html.

(9] ORBacus. http://www.orbacus.com.

[10J COULORIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Distributed Systems, Concepts and Design.** 2. ed. England: Addison Wesley, 1996.

[11] INTERNATIONAL TELECOMMUNICATION UNION. **Common management information service definition for CCITT applications**, X.710. [S.I.], 1991.

[12] INTERNATIONAL TELECOMMUNICATION UNION. **Information technology - Open Systems Interconnection - Sytems management: Alarm reporting function, X.733.** [S.I.], 1992.

[13] PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ. **Gerência Integrada de Rede, Sub-projeto PUC-PR - Relatório Técnico: Testes dos Canais de Eventos dos ORBs Orbacus e TAO.** Curitiba, 1999.