Desenvolvimento de Agentes OSI para Gerência de Desempenho de Sistemas de Transmissão Digital PDH — *Plesiochronous Digital Hierarchy*

*José Félix Furtado de Mendonça** *José Marcos Silva Nogueira*

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
CP 702 - CEP 30161-970 - Belo Horizonte, MG
E-mail: {jfelix,jmarcos}@dcc.ufmg.br

Resumo

Nesse trabalho vamos apresentar uma solução para efetuar o supervisionamento de parâmetros de desempenho do Sistema de Transmissão Digital PDH — *Plesiochronous Digital Hierarchy*, desenvolvida de acordo com os padrões de gerenciamento do modelo OSI/ISO, no escopo da arquitetura TMN. O trabalho foi efetuado através da utilização dos recursos oferecidos pelo ambiente de desenvolvimento Q3ADE — *Q3 Agent Development Environment*. O sistema desenvolvido é composto por um sistema simulador do PDH, um agente de gerenciamento e um sistema de gerenciamento ou gerente. O Sistema de Transmissão Digital PDH é uma etapa da evolução dos sistemas de telecomunicações para atender a demanda por grandes volumes de tráfego de dados e de voz.

Abstract

In this work we present a solution to supervise PDH (Plesiochronous Digital Hierarchy) Transmission System performance parameters. It has been developed using a network management platform called Q3ADE — Q3 Agent Development Environment. The developed system is composed by three parts: a PDH system simulator, a management agent and a management system (manager). PDH is a digital transmission system that has been playing an important role in the efforts of telecommunication systems to support the raising demand for great volume of data and voice traffic.

1 Introdução

As redes de computadores, hoje em dia, estão se tornando uma exigência do mundo moderno. O tráfego de informações entre empresas, cidades, países está cada vez mais

*Licenciado pelo Banco Central para o Curso de Mestrado em Ciência da Computação no DCC/UFMG

demandado e se realizando a taxas cada vez mais elevadas. Existe, é verdade, um parque muito grande já instalado, e a cada dia, mais e mais equipamentos compondo teias menores de diversos ramos tecnológicos — baixa, média e alta velocidade, estão sendo agregados.

Neste panorama, boas soluções para o gerenciamento de redes proprietárias já existem. Todavia, a corrida tecnológica e a crescente necessidade de interoperar com outras instalações invariavelmente transformam as configurações proprietárias em configurações heterogêneas. A partir de então, apenas soluções proprietárias isoladas para o gerenciamento deixarão de ter efeitos. A solução passa por um gerenciamento integrado e rigorosamente padronizado, independente dos dispositivos, que sabemos não é uma tarefa trivial. Grandes esforços e recursos, centrados principalmente em padronizações, vêm sendo paulatinamente aplicados na solução desse problema. Conforme podemos perceber pela vasta e recente literatura, onde podemos citar como exemplo: [Black, 1992, BRISA, 1993, BRISA, 1994, Leinwand e Fang, 1993, Stallings, 1993, Rose, 1990, Perrow *et al.*, 1995, McCarthy *et al.*, 1995, Yemini, 1993, Klerer, 1993, Raman, 1993, Hayes, 1993].

Esta solução está gradativamente sendo alcançada. O panorama hoje é bastante mais promissor. Atualmente estão começando a surgir as primeiras plataformas não proprietárias, para permitir o desenvolvimento de aplicações de gerenciamento dentro dos padrões OSI — *Open System Interconnection* do ISO — *International Organization for Standardization*: são escritas em linguagem orientada por objeto, possuem bibliotecas de classes para suporte às operações padrões de gerenciamento, etc. tornando quase obrigatório que os códigos das aplicações sejam também escritos dentro do mesmo paradigma. As linguagens mais presentes no caso tem sido o ASN.1 — *Abstract Sintax Notation One* [Rose, 1990, Bapat, 1994], para descrever uma sintaxe abstrata para os tipos de dados e suas instâncias, e o C++ [Stroustrup, 1991, Bapat, 1994], usada na construção das plataformas. Como exemplo destas, tem-se o OSIMIS — *OSI Management Information Service* [Pavlou *et al.*, 1995, Souza *et al.*, 1995], a MACT — *Management Agent Creation Tool* [Perrow *et al.*, 1995] e o Q3ADE — *Q3 Agent Development Environment* [UH Consulting, 1994d, UH Consulting, 1994b, UH Consulting, 1994c, UH Consulting, 1994e, UH Consulting, 1994a]. Um ambiente de desenvolvimento de padrão OSI muito usado tem sido o ISODE — *The ISO Development Environment* [Rose *et al.*, 1991a, Rose *et al.*, 1991b, Rose *et al.*, 1991c, Rose *et al.*, 1991d], que representa um dos grandes símbolos desses esforços, e pode ser considerado o marco precursor dessas plataformas.

Esses padrões ainda não adquiriram a unanimidade da comunidade especializada da área, existem resistências. São padrões complexos de serem implantados, e a pilha OSI, com sua exigência de estabelecimento de conexão em diversos níveis pode resultar em um gerenciamento muito caro, uma vez que pode haver um grande aumento de tráfego em decorrência só do gerenciamento em si. O consenso, no entanto, é o de que o padrão OSI irá pouco a pouco sendo encampado. Entretanto, hoje, o padrão *de facto* é o gerenciamento SNMP — *Simple Network Management Protocol*. E para o futuro, já se trabalha com a previsão de que o grosso do gerenciamento ocorrerá de forma distribuída [Yemini, 1993], onde os dispositivos possuirão módulos inteligentes para controlar muitas daquelas tarefas que hoje requerem a interveniência do módulo gerente. Com isso, dois aspectos críticos da questão perderão o impacto que hoje podem causar: menor risco de um tráfego excessivo, uma vez que o diálogo gerente/agente ocorrerá numa freqüência bem menor. Com isso serão reduzidos os estabelecimentos de conexões, troca de mensagens, etc. Por fim, em conseqüência disso, o módulo gerente ficará liberado para cuidar apenas das tarefas mais críticas.

Cabe registrar, que um segmento pioneiro na adoção dos padrões do modelo OSI de gerenciamento foi o de TMN — *Telecommunications Management Network* [Klerer, 1993]. Se somarmos a isso o fato de que a evolução das redes de telecomunicações está tornando-as cada vez mais similares às redes de computadores, embora mais complexo, fica fácil perceber porque, com o surgimento das plataformas de gerenciamento integrados que seguem os mesmos padrões (e.g. Q3ADE), os problemas do gerenciamento de dispositivos, em ambos os casos, convergem para uma solução cuja estrutura pouco difere uma da outra. Dessa forma, para um trabalho de construção de Agentes OSI, a escolha do objeto gerenciado chega a não ser relevante, uma vez que a estrutura da solução diferirá pouco de um caso para outro.

A escolha do Sistema PDH, como objeto gerenciado tem algumas razões: o sistema PDH é relevante como meio de transmissão e tem base instalada grande no mundo; existem poucas informações disponíveis sobre o PDH na literatura, não tendo sido possível encontrar informações sobre a existência de MIB específica; e os sistemas instalados vão coexistir por muito tempo ainda com outros sistemas de transmissão, principalmente nos países em desenvolvimento, como o Brasil.

Este trabalho não mostra apenas implementação de mais um agente: mais do que isso, o objeto gerenciado (PDH) é importante e precisa ser integrado nos ambientes de gerenciamento das empresas, o padrão perseguido de gerenciamento é o TMN, as ferramentas e ambientes de desenvolvimento utilizados são todas atuais, complexas e aderentes aos padrões.

Esse artigo está dividido nas seguintes seções: na primeira seção fazemos a introdução. Nas próximas três seções descrevemos de forma muito resumida o Modelo OSI de gerenciamento, em seguida, o Sistema de Transmissão Digital PDH e, por fim, a Plataforma Q3ADE. Na quinta seção apresentamos a concepção do Sistema de Gerência, onde descrevemos os detalhes das especificações, do projeto e da implementação. Dedicamos a sexta seção para falar sobre o ambiente de desenvolvimento, a integração das partes e os testes. A sétima e última seção deixamos para as conclusões.

2 O gerenciamento OSI, plataformas e ferramentas

O Gerenciamento de Sistemas OSI é uma parte importante dos esforços que visam alcançar um tratamento padronizado para as soluções de gerenciamento de redes de computadores.

O ambiente de gerenciamento de redes OSI está centrado em três conceitos: gerente, agente e objeto gerenciado, conforme esquematizado na figura 1.

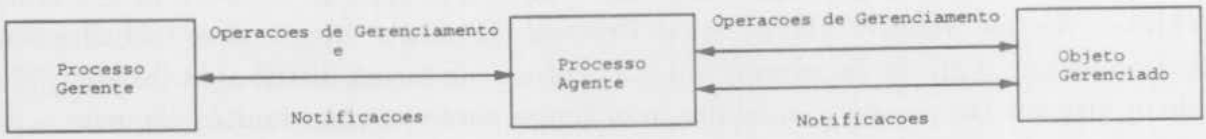

Figura 1: Processos do Gerenciamento OSI

Ao gerente cabe fazer solicitações ao agente para obter informações sobre os objetos gerenciados, ou para execução de alguma operação de gerenciamento sobre os mesmos.

O papel do agente é o de executar as solicitações do gerente. Pode, ainda, transmitir ao gerente as notificações emitidas pelos objetos gerenciados. O objeto gerenciado é a representação abstrata do objeto real, isto é, o dispositivo a ser gerenciado.

O conjunto de objetos gerenciados e de suas instâncias formam a Base de Informação de Gerenciamento (MIB — *Management Information Base*).

O Gerenciamento de Sistemas OSI segue um paradigma que é centrado em plataforma porque as aplicações ficam nela centralizadas, separadas dos dados gerenciados e das funções de controle dos dispositivos. Esse paradigma é atualmente amplamente utilizado pelos fabricantes para resolver o problema particular do gerenciamento de seus dispositivos. O grande desafio a ser vencido é o desenvolvimento de convenções para suportar o gerenciamento integrado de redes heterogêneas. O Modelo OSI de gerenciamento surgiu como uma opção para vencer esse desafio.

Para resolver o problema do gerenciamento integrado de redes heterogêneas, tanto o modelo de gerenciamento OSI, quanto o modelo usado na Internet propõem padrões que visam tornar as plataformas capazes de promoverem o gerenciamento com os mais diversificados tipos de dispositivos. Os problemas de padronizações a serem resolvidos para o gerenciamento centrado em plataformas podem ser separados em dois segmentos:

1. os acessos através de plataformas aos dispositivos de vários fabricantes devem ser unificados através de um protocolo de gerenciamento padrão; e

2. a estrutura da base de dados de gerenciamento dos agentes, manipulada pelo protocolo, deve ser padronizada.

O problema, contudo, não ficará de todo equacionado, conforme [Yemini, 1993] ainda subsistem questões que não foram contempladas no escopo dessas duas propostas. Dizem respeito à heterogeneidade de plataformas e de semântica. Existem algumas organizações em consórcio (e.g., OSF, XOPEN, POSIX) trabalhando com vistas a uma solução eficaz para a padronização de plataformas de gerenciamento.

A heterogeneidade de plataformas leva à necessidade de replicação para cada tipo diferente de plataforma. A heterogeneidade de semântica surge quando dispositivos diferentes utilizam informações diferentes para representar comportamentos similares da rede, ou seja não é levado em conta um modelo semântico uniforme para as informações gerenciadas. O IEEE — *Institute of Electrical and Electronics Engineers* e o ITU-T — *International Telecommunication Union, Telecommunication Standardization Sector* trabalham com o intuito de prover um modelo com padrões para informações gerenciadas para uso pelas entidades de protocolos.

Apesar disso após os esforços pioneiros representados pelo surgimento do ambiente de desenvolvimento ISODE e da plataforma OSIMIS começam a surgir plataformas que permitem a construção de Agentes de Gerenciamento conforme os padrões propostos pelo Sistema de Gerenciamento OSI. Uma que podemos citar é o Q3ADE descrita na seção 4.

3 O Sistema de Transmissão Digital PDH

O PDH — *Plesiochronous Digital Hierarchy* é um sistema hierarquizado de transmissão que surgiu no campo da telefonia digitial, para atender a demanda crescente por transferências de grandes volumes de tráfegos de informações.

O Sistema PDH era a tecnologia predominante quando surgiram, há mais ou menos dez anos atrás, as primeiras RDSI[1] (ISDN[2]), atualmente conhecidas como RDSI-F[3] (N-ISDN[4]), e precursoras das atuais RDSI-FL[5] (B-ISDN[6]) surgidas no final dos anos 80's.

Junto com as RDSI-FL surgiu uma nova proposta de estrutura de transmissão conhecida como SDH/SONET (SDH — *Syncronous Digital Hierarchy*)/(SONET — *Synchronous Optical NETwork*) que vem presentemente substituindo o PDH. O SDH/SONET possui várias vantagens sobre o PDH, sem deixar de ser compatível com os canais dessa hierarquia ainda existentes, e que são ainda o sistema de transmissão digital telecomunicações dominante no mundo. A RDSI-FL é também conhecida como a via da informação, que vem sendo construída sobre a base fornecida pelo SDH/SONET.

O primeiro padrão digital a entrar em operação foi o DS-1 (DS — *Digital Service*), desenvolvido pela AT&T no início dos anos 60's. O DS-1 reúne as amostragens de 24 canais de voz, e fornece uma banda passante de 1,544 Mbps. Um canal digital de voz tem uma velocidade de 64 Kbps, sendo referido como DS-0. Neste sistema, dois, três ou mesmo os 24 canais poderiam ser utilizados todos ao mesmo tempo, por um único usuário, para a transmissão de uma maior quantidade de informação de um só vez.

Nessa época outros padrões também começaram a surgir na Europa e no Japão. O primeiro a surgir na Europa foi o E-1, que reunia 32 canais.

Outro padrão largamente utilizado por empresas é o sistema DS-3, multiplexando 28 canais DS-1, ou 672 canais DS-0 (64 Kbps — a conexão telefônica padrão), permitindo taxas de transferência nominais da ordem de 44,763 Mbps. A figura 2 ilustra a formação dessas hierarquias de dados.

Com o advento das transmissões digitais o sincronismo passou a ser um problema crescente, porque até então os sistemas consistiam de dispositivos analógicos e não eram sincronizados com base em nenhum relógio. O sincronismo é importante porque permite a fácil localização e extração da informação correta, dentre muitos de outras, constante da massa multiplexada. Quando, por exemplo, pulsos PCM — *Pulse Code Modulation* são utilizados em transmissão digital, a sincronização deve ser realizada em três níveis diferentes: sincronismo de bit, de *time slots*[7], e de quadro (*frame*) — (os bits trafegam imersos em quadros).

A sincronização da rede pode ser completada pela sincronização dos relógios de todos os nodos intervenientes, de modo que as transmissões originárias desses nodos tenham todas a mesma taxa de bits na rede em média. São também instalados *buffers* estrategicamente, em alguns pontos, com vistas a absorver eventuais diferenças entre a taxa de bits real e aquele previsto em média. A sincronização dos relógios desempenha um importante papel em todo esse processo, porque na maioria das redes digitais o sincronismo é derivado desses relógios. A falta de sincronismo entre esses nodos associados impõe um prejuízo à rede denominado *slips*. *Slips* é a perda de sincronismo e a perda de detecção de bits.

A sincronização dos relógios, contudo, não põem um ponto final ao problema. Há problemas decorrentes de condições ambientais, meio físico, etc. que provocam desvio

[1] Redes Digitais de Serviços Integrados

[2] Integrated Services Digital Network

[3] Redes Digitais de Serviços Integrados de Faixa Estreita

[4] Narrowband Integrated Services Digital Network

[5] Redes Digitais de Serviços Integrados de Faixa Larga

[6] Broadband Integrated Services Digital Network

[7] é o intervalo de duração do conjunto de bits que representam o valor binário de uma amostra do quadro de amostras resultantes da multiplexação.

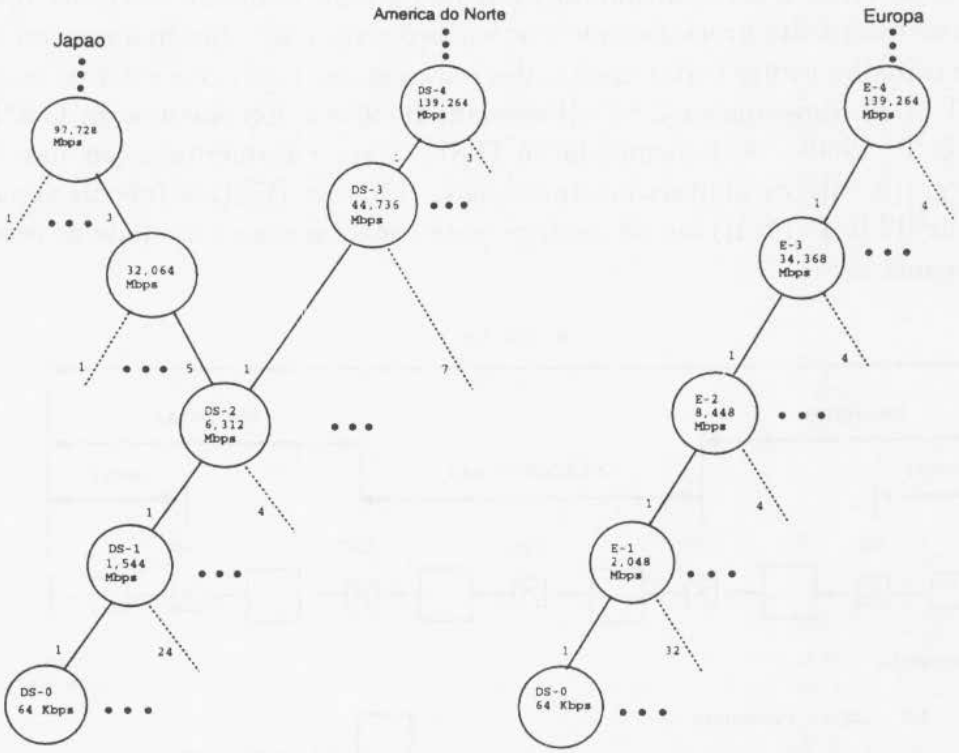

Figura 2: Esquema de formação das hierarquias de dados.

e vibração de fase (*phase wander and jitter*) na taxa de bits que provocam também a ocorrência de *slips*. " *"Jitter" é definido como uma variação de curta duração na fase de um sinal digital, que inclui as variações acima de 10 Hz. As causas de "jitter" incluem os ruídos comuns, o preenchimento de bits ("bit stuffing") na multiplexação ou as falhas de repetidores."* [Black, 1995].

Conforme esse mesmo autor, " *"Wander" é uma variação de mais longa duração da fase do sinal, e inclui todos as variações abaixo de 10 Hz. "Wander" também pode incluir os efeitos do desvio de freqüência ("frequency departure") que é uma diferença de freqüência constante entre os elementos da rede. "Wander" é quase inevitável em qualquer rede, devido às pequenas variações nas diferenças de freqüência dos relógios, aos atrasos de sinal no meio físico, bem como às operações de preenchimento de bits ("bit stuffing")."*

O problema de *phase wander and jitter* é superado por uma utilização adequada de *buffers*. Portanto, *buffers* de tamanho adequado nas interfaces do sistema de transmissão digital e o alinhamento dos relógios dos nodos intervenientes são os mecanismos básicos para se atingir os objetivos de redução da ocorrência de *slips* na rede.

Existem vários métodos para se fazer o sincronismo de uma rede. Dentre esses está a operação plesiócrona, que no início dos anos 60's foi um processo pioneiro.

Uma rede plesiócrona é aquela cujos elementos de rede são sincronizados com relógios distintos, todos com quase a mesma sincronização [Black, 1995].

Os sistemas de transmissão digital que foram sendo construídos com base nas hierarquias digitais de dados, e que tinham a operação plesiócrona como método de sincronização, ficaram conhecidos como sistema de transmissão digital PDH.

Rigorosamente uma modalidade de transmissão síncrona é aquela na qual o sincro-

nismo dos componentes dos nodos da rede é derivado de um relógio de referência. Sob esse ponto de vista, o sincronismo do PDH não é rigorosamente síncrono, mas é um sincronismo de muito alta precisão onde e as variações recaem sobre margens muito estreitas.

Nesse trabalho vamos tratar apenas dos parâmetros envolvidos em duas recomendações do ITU-T, a Recomendação G.811 [Freeman, 1989] e a Recomendação G.821 [Freeman, 1989, ITU-T, 1980]. A Recomendação G.811 trata da sincronização nas conexões de 64 Kbps entre enlaces digitais internacionais. Enlaces Digitais Internacionais (figura 3 extraída de [ITU-T, 1984]) são necessários para conectar uma variedade de redes nacionais e internacionais.

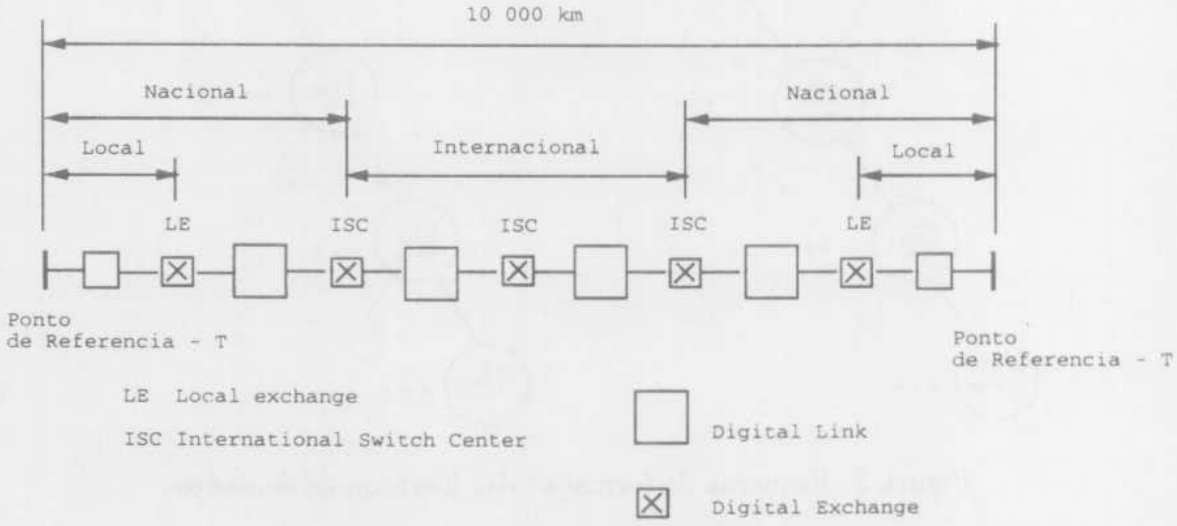

Figura 3: Esquematização de uma rede digital de transmissão.

Nesses enlaces pode ocorrer uma coexistência de operações plesiócronas e síncronas. Quando coexistências dessa natureza se manifestarem nas redes que processam tráfegos internacionais, é necessário que seus nodos atendam a ambos os tipos de operação. Nestes casos, devem ser observados alguns cuidados em vistas de se evitar que os controles de sincronização não ocasionem defasagens de curta duração nas freqüências dos relógios, uma vez que isso será inaceitável para a operação plesiócrona. Por isso, essas defasagens devem obedecer alguns requisitos que são especificados nessa Recomendação [Freeman, 1989]. Nesse documento são definidos e estabelecidos, por categoria de desempenho, valores limítrofes para os seguintes parâmetros: Erro de Intervalo de Tempo (TIE — *Time Interval Error*) e Desvio de Freqüência (FD — *Frequency Departure*). O parâmetro TIE é a magnitude da diferença entre a defasagem de tempo com relação a uma hora padrão ideal, como o UTC — *Universal Coordinated Time*, no início e no fim de um período de S segundos, ou seja

$$TIE(S) = | \Delta T(t+S) - \Delta T(t) | \qquad (1)$$

O parâmetro FD é o TIE dividido por S. A tabela 1 reúne informações, extraídas da tabela 10.2 de [Freeman, 1989], sobre valores limítrofes que devem ser obedecidos para esses parâmetros.

Em outro documento, a Recomendação G.821, o ITU-T aborda o problema do desempenho quanto a erros, cujos objetivos são baseados no canal digital de voz de 64 Kbps, denominado de DS-0. Os erros de bits surgem nas redes digitais, decorrentes das fontes

de conversão analógico/digital e vice-versa. Esta Recomendação define e estabelece valores limítrofes para os seguintes parâmetros: Minutos Degradados (*Degraded minutes*), Segundos Errados Gravemente (*Severely errored seconds*) e Segundos Errados (*Errored seconds*), que devem ser respeitados nas conexões internacionais de ISDN. A tabela 2 extraída de [Freeman, 1989, ITU-T, 1980], mostra os valores limítrofes que devem ser obedecidos para esses parâmetros.

Categoria de desempenho	**Nodo**	
Nominal	**Local**	**Trânsito**
	$(100S)ns + 1/8$ unit interval[8] para $S < 5$	idem
	$(5S + 500)ns$ para $5s < S < 500s$	idem
	$(10^{-2S} + 3000)ns$ para $S > 500$	idem
(a)	$> 10^{-11}S$ $\leq 10^{-8}S$	$> 10^{-11}S$ $\leq 2.0 \times 10^{-9}S$
(b)	$> 10^{-8}S$ $\leq 10^{-6}S$	$> 2.0 \times 10^{-9}S$ $\leq 5.0 \times 10^{-7}S$
(c)	$> 10^{-6}S$	$> 5.0 \times 10^{-7}S$

Tabela 1: Valores limítrofes para **TIE**

Classificação de Desempenho	**Objetivo**
Minutos Degradados (*Degraded Minutes*)	$< 0,1$
Segundos Errados Gravemente (*Severely errored seconds*)	$< 0,002$
Segundos Errados (*Errored seconds*)	$< 0,08$

Tabela 2: Valores limítrofes para os parâmetros de erros de bits

4 A Plataforma Q3ADE

A Plataforma Q3ADE — *Q3 Agent Development Environment*[9] foi implementada pela empresa UH *Consulting* da Dinamarca e está em início de comercialização. É uma plataforma de gerenciamento orientada por objeto, voltada para o modelo OSI de gerenciamento e implementada basicamente em C++. É parcialmente baseada no ambiente de desenvolvimento ISODE — *The ISO Development Environment* e na plataforma OSIMIS — *OSI Management Information Service*.

[8] *unit interval (UI)* - valor que depende do tipo da codificação do sinal.

[9] [UH Consulting, 1994d, UH Consulting, 1994b, UH Consulting, 1994c, UH Consulting, 1994e, UH Consulting, 1994a]

O Q3ADE objetiva primordialmente fornecer meios para a construção a custos reduzidos de Agentes de alto desempenho compatíveis com o CMIS/CMIP. É portátil para muitas versões de UNIX e Windows NT.

Possui uma infraestrutura baseada em multitarefa e *non-blocking* que permite a comunicação em tempo real com várias interfaces simultaneamente e, ainda de forma altamente paralela manter o processamento de transações e de notificações.

Traz um conjunto de bibliotecas de sintaxes e objetos pré-definidos. Permite imprimir persistência aos objetos. Tem algumas facilidades para testes e *debugging*, inclusive uma interface TTY, por onde se pode invocar todas as requisições CMIS. Esses comandos CMIS e alguns outros comandos que podem ser executados através dessa interface TTY fazem parte de uma interface de operação. É uma interface simples que na nomenclatura do Q3ADE é denominada de *Command Interpreter*.

5 Concepção do Sistema de Gerência PDH

5.1 Objetivos

Como vimos, a qualidade dos serviços que se utilizam das transmissões plesiócronas dependem da supervisão e monitoramento de alguns parâmetros. Nosso objetivo foi construir, dentro do modelo OSI de gerenciamento, utilizando os recursos da plataforma Q3ADE, um sistema que possa: periodicamente fazer a coleta das mensurações desses parâmetros e reuní-las em uma estrutura de armazenamento; promover as correlações necessárias para verificar se essas medidas estão dentro dos valores limítrofes especificados para cada um deles; e, quando constatar alguma distorção, fazer a sua notificação.

5.2 Especificações

As especificações originais elaboradas para o sistema estão relacionadas nesta seção.

Na construção do protótipo foi adotada a estrutura do modelo de gerenciamento OSI — *Open System Interconnection* (Agente, Gerente e Objeto Gerenciado).

Com relação ao Objeto Gerenciado (o Sistema PDH), foram supervisionados todos aqueles parâmetros já mencionados nas tabelas 1 e 2.

É mantida uma estrutura de armazenamento, MIB para guardar pelo menos os dez últimos valores apresentados por cada um dos parâmetros a serem supervisionados.

A MIB está associada ao Agente e é utilizada também para registrar outros eventos relacionados, como a criação das instâncias de seus objetos e as notificações a serem emitidas do Agentes para o Gerente, quando for atingido valores limítrofes de quaisquer dos parâmetros supervisionados.

As medições dos parâmetros a serem supervisionados são realizadas em intervalos de S segundo, que é informado ao Agente através de um atributo criado para essa finalidade. Esse atributo possui um valor inicial que poderá ser modificado posteriormente em tempo de execução.

Independentemente dessas notificações que o Agente eventualmente poderá encaminhar para o Gerente, este poderá solicitar do Agente, a qualquer instante, informações sobre os parâmetros supervisionados.

A interface do Objeto Gerenciado com o Objeto Real não entrou no escopo deste projeto, seu comportamento é simulado pela atribuição de valores randômicos aos parâmetros

supervisionados.

5.3 Projeto do Sistema

Para atender às especificações e tendo utilizando a plataforma Q3ADE, o protótipo tem as seguintes características:

5.3.1 Estrutura do Sistema de Gerência

A estrutura da solução, conforme a figura 4, é composta de três entidades básicas: Sistema Gerenciador de Teste no papel de Gerente, Sistema Agente e Processo Simulador do PDH, no papel de Objeto Gerenciado. A cada S segundos, o Processo Simulador do PDH informará os valores das medições sobre os atributos. Se tiver havido alterações nos valores, o Sistema Agente recebe e armazena as informações em uma base de informações de gerenciamento (MIB). Poderá notificá-las para o Sistema Gerenciador de Teste, se tiver sido programado para tal, ou poderá aguardar um pedido do mesmo para prestar essas informações. Toda notificação antes de ser emitida é registrada na MIB, onde suas informações são também armazenadas de acordo com o formato padrão preconizado no Modelo de Informação OSI/ISO para o tipo de evento que reporta. Da mesma forma, outros eventos de relevância que ocorrerem também são registrados na MIB (e.g. criação de objetos — *object creation*). A comunicação entre o Sistema Agente e o Sistema Gerenciador de Teste se dará via CMIP.

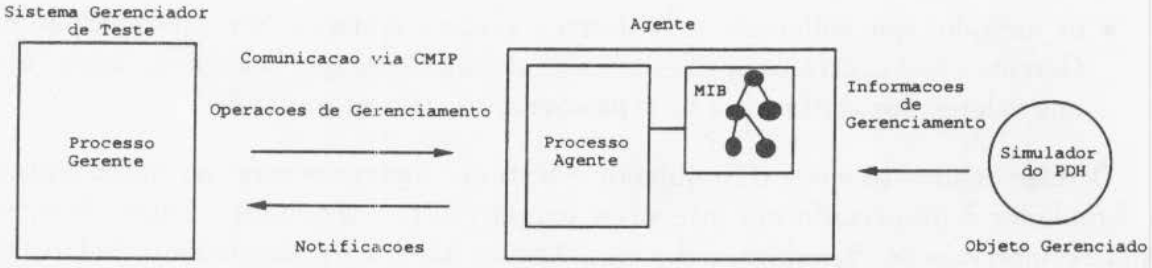

Figura 4: Estrutura da Solução.

5.3.2 Descrição do Sistema Gerenciador de Teste

O Sistema Gerenciador de Teste é a entidade responsável pelo recebimento das notificações do Sistema Agente e, independentemente disso, pode, a qualquer instante, solicitar informações sobre os parâmetros que estejam armazenados na MIB do Agente. Para o escopo do presente trabalho, o Sistema Gerenciador de Teste, no papel de gerente, é mais um elemento para viabilizar a realização dos testes, do que um processo gerente bem elaborado, dotado de recursos gráfico, de tomadas de decisão, etc. O objetivo maior foi mais a construção de agentes OSI do que a realização de uma aplicação de gerenciamento completa, que neste caso exigiria o desenvolvimento de diversas outras facilidades.

5.3.3 Descrição do Agente

O Agente recebe informações do Simulador do PDH e encaminha notificações ao Sistema Gerenciador de Teste, se estiver programado para tal. Caso contrário, só o faz sob demanda deste. No Agente também está localizada a MIB, onde ficam armazenadas todas as informações recebidas. Essas informações representam os dados sobre as medições dos parâmetros sob supervisão ou quaisquer outros eventos (e.g. uma Criação de Objeto) e são gravadas na MIB no formato previsto no modelo OSI para cada tipo de evento. Alterações dos valores dos atributos que representam os parâmetros sob supervisão são gravados sob o formato *attributeValueChangeRecord* e as ultrapassagens de seus valores limítrofes são gravados sob o formato de *processing error alarm*. Criações de Objetos são gravados sob o formato *objectCreationRecord*. Veremos na seção 5.4 que serão os dois formatos mais utilizados.

5.3.4 Descrição do Processo Simulador do PDH

O Processo Simulador do PDH, como dispositivo (Objeto Gerenciado) a ser monitorado, é implementado como uma nova classe de objeto, que em tempo de execução é uma das instâncias de objetos que compõe a estrutura do Sistema Agente. Essa nova classe possui todos os atributos para representar os parâmetros a serem supervisionados e contém ainda vários métodos. Dentre estes métodos, os mais importantes são:

- o método que programa o Sistema Agente para colocar em execução o Simulador em intervalos regulares com vistas à captação de dados;
- os métodos que fazem a captação dos dados do objeto real (neste caso a captação se reveste em processo de instanciação randômica dos atributos); e
- os métodos que solicitam ao Sistema Agente a emissão de notificações ao Sistema Gerente. Isso ocorre basicamente nas duas situações que já mencionamos: mudança nos valores dos atributos e ultrapassagem de valores limítrofes.

O relacionamento entre o Simulador e Sistema Agente ocorre por varredura, ou seja o Simulador é despertado em intervalos regulares de S segundos. Então, toda vez que vence o intervalo de varredura, o Sistema Agente solicita ao Simulador os valores de cada parâmetro naquele instante. Esses valores são armazenados em sua MIB e notificações podem ser encaminhadas ao Sistema Gerente caso as condições estabelecidas para isso sejam atingidas.

5.3.5 Considerações Gerais

Sob o ponto de vista prático, na maioria das vezes os atributos a serem utilizados numa aplicação de gerenciamento serão derivados de alguns poucos tipos comuns definidos em [ITU-T, 1992] — *gauges*, *counters*, etc. Eles normalmente estão disponíveis sob a forma de objetos nas bibliotecas de classes de objetos das plataformas orientadas por objetos (e.g. Q3ADE).

No modelo de gerenciamento que adotamos, o supervisionamento de fatores com comportamentos temporais são normalmente descritos em termos de *counters* e *gauges*. Nesse trabalho os atributos que utilizamos para representar os parâmetros a serem supervisionados serão descritos como *gauges*. De uma forma bastante resumida, *gauge* é uma

abstração de gerenciamento do valor de uma variável dinâmica, podendo estar relacionado com variáveis, para registrar os valores máximos e mínimos ocorridos no período (*tide-mark*) e/ou para informar valores limítrofes máximos e mínimos permitidos (*gauge-threshold*), etc. Informações mais detalhadas poderão ser obtidas em [ITU-T, 1992, Stallings, 1993].

5.4 Implementação

Dividimos a implementação nas seguintes partes:

1. Definição e implementação dos atributos;
2. Implementação do Processo Simulador do PDH;
3. O programa principal e inicialização; e
4. Elaboração dos *scripts* para configuração do software.

5.4.1 Definição e implementação dos atributos

Na tabela 3 descrevemos cada atributo necessário sua classe, nome, sintaxe, classe derivada e tipo.

Para implementar todas as classes da tabela 3 utilizamos as macros: `DEFINEPEPSYATTRCLASS` e `IMPLEMENTATTRCLASSTAG`. Esta forma simples de implementar novas classes de atributos só é possível de ser utilizada nos casos em que as novas classes possam ser derivadas de classes pré-definidas. A partir de então, nos Objetos Gerenciados, poderão ser declarados atributos do tipo da nova classe. Como exemplo, para criar a classe `TimeIntErrATC` e o atributo `timeIntErr`, por exemplo executamos as macros:

```
DEFINEPEPSYATTRCLASS(TimeIntErrATC, double, GaugeReal)
```

e

```
IMPLEMENTATTRCLASS(TimeIntErrATC, timeIntErrS5, SMIX-real, GaugeReal)
```

5.4.2 Implementação do Processo Simulador do PDH

Para implementar o Processo Simulador do PDH, como uma nova classe de objeto gerenciado, utilizamos novamente das facilidades providas pelo Q3ADE, através da utilização de *templates* existentes para essa finalidade. Seus preenchimentos devem ser feitos de forma muito cuidadosa e ali já estão indicadas todas as funções padrões, pré-definidas e disponíveis em bibliotecas, que devem constar em toda nova classe. Traz uma enorme economia de tempo.

São funções que fazem as inicializações, cuidam da comunicação, tornam o objeto serializável (*streamable*), etc. Assim, a maior parte do trabalho ficou concentrada nas funções que implementam a inteligência do objeto e que fazem o papel da interface com o objeto real. Para fazer essa parte foram desenvolvidos vários métodos, sendo o mais importante deles o método `simula`. Este método desempenha um papel central na nova classe. É por seu intermédio que são feitos:

- as atribuições dos valores limítrofes;

Classe do AtributO	Nome do Atributo	Sintaxe	Classe Derivada	Tipo do Atributo
PdhObjIdATC	pdhObjId	SMIX-ia5string	IA5StringATC	type_UNIV_IA5String
TextoExtraATC	textoExtra	SMIX-GraphicString	GraphicStringATC	type_UNIV_GraphicString
TipoCodificPdhATC	tipoCodificPdh	SMIX-integer	IntegerATC	long int
SegundosSATC	segundosS	SMIX-integer	IntegerATC	long int
UnitIntervalATC	unitInterval	SMIX-Real	RealATC	double
CategDesempATC	categDesemp	SMIX-integer	IntegerATC	long int
TipoNodoATC	tipoNodo	SMIX-integer	IntegerATC	long int
TimeIntErrATC	timeIntErr	SMIX-real	GaugeReal	double
TimeIntErrThldATC	timeIntErrThld	GaugeThreshold	GaugeThresholdReal	GaugeThresholdVal
TimeIntErrTideMarkATC	timeIntErrTideMark	TideMark	TideMarkMax	TideMarkVal
FreqDepartATC	freqDepart	SMIX-real	GaugeReal	double
FreqDepartThldATC	freqDepartThld	GaugeThreshold	GaugeThresholdReal	GaugeThresholdVal
FreqDepartTideMarkATC	freqDepartTideMark	TideMark	TideMarkMax	TideMarkVal
DegradeMinATC	degradeMin	SMIX-real	GaugeReal	double
DegradeMinThldATC	degradeMinThld	GaugeThreshold	GaugeThresholdReal	GaugeThresholdVal
DegradeMinTideMarkATC	degradeMinTideMark	TideMark	TideMarkMax	TideMarkVal
SevErrSecATC	sevErrSec	SMIX-real	GaugeReal	double
SevErrSecThldATC	sevErrSecThld	GaugeThreshold	GaugeThresholdReal	GaugeThresholdVal
SevErrSecTideMarkATC	sevErrSecTideMark	TideMark	TideMarkMax	TideMarkVal
ErrSecATC	errSec	SMIX-real	GaugeReal	double
ErrSecThldATC	errSecThld	GaugeThreshold	GaugeThresholdReal	GaugeThresholdVal
ErrSecTideMarkATC	errSecTideMark	TideMark	TideMarkMax	TideMarkVal
IntervalAtualizATC	intervalAtualiz	SMIX-integer	IntegerATC	type_UNIV_UTCTime

Tabela 3: Atributos para o PDH

- a captação dos dados simulados do objeto real; e
- os pedidos de notificação ao Sistema Agente.

Essa classe foi derivada por herança múltipla da classe Top e da classe KS. A classe Top representa a classe Top da Recomendação X.721 do ITU-T, incluindo informações sobre o OID da classe, alomorfismo, *packages* instanciados e relação de nomeação e toda nova classe de objeto gerenciado deve derivar da classe Top. A classe KS provê métodos para permitir a freqüência da captação dos valores dos parâmetros em intervalos regulares.

Os métodos que foram desenvolvidos imprimem uma lógica de processamento que permitirá o supervisionamento de apenas um parâmetro por instância da classe, ou seja, para supervisionar mais de um parâmetro deverá ser criada mais de uma instância da classe. Uma exceção é feita apenas nos casos do gerenciamento dos parâmetros *Degraded minutes*, *Severely errored seconds* e *Errored seconds*, que deve ser feito concorrentemente. A classe identifica o parâmetro a ser gerenciado a partir dos valores dos atributos `categDesemp` e `tipoNodo`, que são passados na criação da instância. Os valores que esses atributos podem assumir e o parâmetro que apontam para gerenciamento está enquadrado pelos *enumerates* `_categoriasDesempenho` e `_tipoDoNodo` definidos em `PdhMO1.h` que espelham toda

a gama de categorias tratadas nas tabelas 1 e 2.

A atribuição dos valores limítrofes é feita de forma dinâmica, em tempo de execução, através do método `calcThld` e de outros métodos auxiliares que foram implementados. Como vimos na tabela 1, para alguns parâmetros esses valores são calculados a partir de fórmulas, que dependem do valor de intervalo de S segundos de varredura bem como do *unit interval*. Esse intervalo é passado para a instância através do atributo `segundosS`. Já o tipo de codificação do PDH do qual depende o valor de *unit interval* é passado pelo atributo `tipoCodificPdh`.

O atributo `segundosS` é também utilizado para programar o intervalo de varredura, ou seja, a freqüência de captação dos valores de gerenciamento. Essa programação é feita através do método `wakeUp`, herdado da classe `KS`.

Os demais atributos da tabela 3 são utilizados para guardar as instâncias de valores que os parâmetros a serem gerenciados vão apresentar, seus valores limítrofes máximos e mínimos permitidos e os *tidemark* máximos.

Como se trata de uma simulação, os métodos elaborados, por exemplo `valTieANl`, para captar os valores junto ao objeto real na verdade executam uma simulação. É gerado um inteiro randômico e dependendo da faixa que se encontrar será escolhido uma valor dentre três (sendo dois deles os valores limítrofes permitidos para o tipo de parâmetro que estiver sendo supervisionado e um que está situado entre os dois limites). Isso permitirá a realização dos testes nos três casos mais importantes: na ocorrência de valores extremos e na ocorrência de valores na faixa permitida. Depois de captar o valor atual do parâmetro, o método `simula` invoca o método `emiteNotific` para atualizar seu valor na MIB e emitir as notificações quando couberem.

5.4.3 O programa principal e inicialização

Aqui também pôde ser utilizado outro *template*, que na nomenclatura do Q3ADE é denominado de *Sma* (*System Management Agent*), que contém funções pré-definidas para inicializar as pilhas do protocolo e suas tabelas de sintaxe, para inicializar o *CMISCoordinator* e a MIB. O *CMISCoordinator* gerencia a comunicação estabelecendo e desfazendo as associações do CMIS. Isso é feito por comandos que mostramos nos fragmentos abaixo. O programa principal *Sma* é o elemento gerenciador propriamente dito, cujo código é o mesmo tanto no papel de Agente como no papel de Gerente. Ou seja, tanto o Sistema Agente como o Sistema Gerenciador são disparados utilizando-se o mesmo código do *Sma*, modificando-se apenas o valor de *service* que é passado por parâmetro para o *Sma* e o atributo *systemId* da instância do objeto *system* que é criada para cada um dos sistemas que entram em operação, conforme está mostrado na seção 5.4.4. Ambos *system* e *systemId* são utilizados para denotar o serviço, ou seja, o nome do elemento gerenciador no papel de agente ou de gerente.

Vamos apresentar a seguir os principais pontos do *Sma* que foi utilizado na construção desse protótipo.

Captação dos valores passados via parâmetros

Dois dos parâmetros mais comumente utilizados são o `service` e o `initFile`. O `service` identifica o serviço, ou seja, é o nome dado ao Sistema Agente ou ao Sistema Gerenciador, que nessa implementação foram respectivamente denominados de **"smaagpdh"** **"smagrpdh"**. Já `initFile` identifica o arquivo de inicialização. Esse arquivo

é um *script*, conforme veremos logo abaixo na seção 5.4.4, contendo comandos do *Command Interpreter* para executar o restante das tarefas, como por exemplo: criar objetos, estabelecer valores de atributos, etc.

Tabela de Sintaxe

A Tabela de Sintaxe informa todas as sintaxes existentes nas bibliotecas de sintaxes. Através dessas sintaxes é que é possível a transformação entre formatos de permuta de dados padronizados em consonância com o código BER — *Basic Encoding Rules* do padrão X.209 do ITU-T e as estruturas internas em linguagem C. Agregam também outros métodos que permitem fazer liberação de memória utilizadas pelas estruturas em C, exibir, comparar, copiar e examinar o conteúdo dessas estruturas.

```
//Syntaxes Tables
VoidFunc        syntaxes[] = {
        smi_syntaxes,
        smix_syntaxes,
        isode_attr_syntaxes,
        IFX_syntaxes,
        CMIP_syntaxes,
        X721Attr_syntaxes,
        X721Not_syntaxes,
        ux_syntaxes,
        NULL };
```

Inicialização das pilhas de protocolo e das tabelas de sintaxes

```
  //First initialise protocol stacks and tables
    if ((res = initialiseMSAP(getenv(\"{U}SERPATH"),tailorFile, syntaxes)) !=OK) {
       cerr << "Failed to initialise syntaxes, cause " << res<<endl<<flush;
       return res;
    }
}
```

Inicialização do CMIS coord

```
    // initialise CMIS coord
    TSAPdisconnect tds;

    D_TRACE0("This agent includes the "<<TTYIf::classInfo.cppName);

  //Initialise the CMISCoordinator
    if (agent.initialise(service, &tds) == NOTOK) {
        cerr << "isode error: " << TErrString(tds.td_reason)<<endl<<flush;
        return NOTOK;
    }
```

Inicialização da MIB

```
  // Initialise the MIB.
    {
        CmisClient client ("cmdParser",destService,destHost);
        if (initFile) {
            if ((res = client.readFile(initFile)) != OK) {
                cerr << "Stopped Processing Initialisation File"<<endl<<flush;
                return res;
            }
        }
        else
            cout << "No Local Mib Service "<<endl<<flush;
    }
```

Os comandos de inicialização da MIB processam o arquivo `initFile`. Tanto o `initFile` quanto `service` na inicialização do `CMISCoordinator` são, como vimos, passados por parâmetros ao programa principal.

5.4.4 Elaboração dos *scripts*

Esta é uma parte importante do projeto porque é através desses comandos que o Sistema Agente e seus Objetos Gerenciados e o Sistema Gerenciador são postos plenamente em execução. Os *scripts* são arquivos ASCII contendo comandos da interface *Command Interpreter* para disparar outros objetos que também devem fazer parte do Sistema Agente e alguns do Sistema Gerenciador. Como exemplo, os objetos Simulador, *log*, *eventForwardingDiscriminator* e *system*. Os *scripts* servem também concomitantemente para estabelecer valores de atributos inerentes a esses objetos para, por exemplo, escolher os tipos de *event report* com permissões para serem levados adiante pelo Sistema Agente e qual o destino que devem ter, o estado do objeto, se este será persistente ou não, etc. Tudo isso podendo posteriormente ser modificado, em tempo de execução inclusive, caso surja a necessidade de ajustes na configuração do elemento gerenciador quanto aos recursos proporcionados pelas instâncias desses mesmos objetos.

Foram construídos diversos *scripts* para que o Sistema Agente executasse dentro das especificações de nossa proposta. Os comandos constantes em cada *script* são processados através de uma interface **TTY**, existente internamente no programa principal. Além disso através do comando `start TTYIf`, normalmente incluído no `initFile`, o programa principal fica habilitado para prover a interface **TTY** para uso externo por parte do usuário, por onde poderão ser emitidos qualquer comando da interface *Command Interpreter*, para realizar tarefas como obter o valor de atributo, obter o *dump* da MIB de um agente, alterar o valor de um atributo, etc.

A instância da classe *system* que é criada denota o sistema propriamente dito. O conjunto de instâncias resultante de todas essas inicializações, que representam o sistema em plena execução, formam uma estrutura em árvore, tendo a instância de *system* no nodo raiz e as instâncias dos demais objetos nos outros nodos interagindo uns com os outros de forma bastante complexa. Na nomenclatura do modelo de gerenciamento ISO/OSI essa estrutura recebe o nome de árvore de contenção ou árvore de nomeação.

Os *scripts* que foram elaborados para o Sistema Gerenciador são de menor complexidade do que aqueles do Sistema Agente. O gerente não dispõe de MIB e portanto não necessita de instância da classe *log*, etc.

6 Ambiente de Desenvolvimento, Integração e Testes

O ambiente de desenvolvimento constou de:

- Estações de trabalho Sun com Sistema Operacional Solaris 2.x;
- Linguagem de programação C e C++;
- Versão beta da Plataforma Q3ADE; e
- Porlatex, Xfig, Idraw, editor vi e X11.

A integração e os testes foram processos muito simples. Poderam ser feitos em uma mesma máquina onde foram abertos dois terminais separados. Em seguida, foram disparados os *scripts* do Sistema Agente e do Processo Simulador do PDH em um terminal e os

scripts do Sistema Gerenciador no outro. Após eliminados todos os problemas surgidos nos primeiros testes chegou-se a uma versão que foi capaz de operar dentro das especificações. Após vencido o intervalo de atualização (varredura) o Processo Simulador do PDH foi executado pelo Sistema Agente e começou gerar eventos. O Sistema Agente recebia as notificações decorrentes desses eventos e, ápos os registros na MIB, redirecionava essas notificações para o Sistema Gerenciador de Teste. O Sistema Gerenciador de Teste recebia as notificações e mostrava em seu terminal. A morte de um processo não implica na morte do outro, de modo que o comando *shutdown* tem de ser aplicado separadamente em ambos. Tanto o Sistema Agente quanto o Sistema Gerenciador poderam executar novos comandos através de suas interfaces `TTYIf`.

7 Conclusão

O desenvolvimento de agentes tal como aqui colocado não é um processo trivial. O modelo de padronização é complexo, principalmente no caso da modelagem da informação. Requer a utilização da técnica de orientação por objeto. Percebemos, no entanto, que essas plataformas que começam a surgir estão vindo acompanhadas de uma série de recursos, como biblioteca de classes de objetos (com seus métodos, e macros pré-definidos) e APIs[10]. São facilidades que diminuem enormemente o trabalho de construção dos códigos.

Houve dificuldades na pesquisa que realizamos para encontrar informações sobre o Sistema Digital PDH, de forma que a abrangência da supervisão que imprimimos ao Sistema inclui apenas aqueles parâmetros que estavam qualificados na bibliografia que nos foi possível colecionar em tempo hábil.

Todavia, a MIB desse Sistema, construída neste trabalho, poderá ser expandida para ampliar a capacidade do Sistema, quando se conseguir mais informações sobre o Sistema Digital PDH.

A ligação com o objeto real, que aqui substituímos pelo processo de simulação randômica dos valores a serem captados, é outro detalhe de implementação que depende essencialmente de informações técnicas sobre cada tipo de dispositivo. De maneira que no momento que essas informações forem detalhadas, os métodos de captação poderão ser reformulados para captar dados reais ao invés de dados simulados.

O paradigma de Agente, Gerente, MIB e Objetos Gerenciados que foi empregado na parte prática desse projeto é exatamente aquela que é atualmente a mais preconizada para o gerenciamento de múltiplas redes interligadas e heterogêneas (soluções proprietárias tornam-se ineficazes) em vários dos modelos de padronizações existentes. Os modelos e padrões que mais se destacam são o SNMP — *Simple Network Management Protocol* da Internet, o mais amplamente utilizado e o CMIP — *Commom Management Information Protocol* do OSI/ISO, cujo uso vem crescendo muito nos últimos anos principalmente em TMN — *Telecommunications Management Network*.

Os padrões estabelecidos para o modelo OSI/ISO que utilizamos nesse trabalho são, como já dissemos, complexos. O modelo SNMP da Internet, por ser mais simples, está sendo mais utilizado e tornou-se o padrão *de facto*. Entretanto o uso do modelo OSI/ISO vem crescendo muito nos últimos anos, principalmente em TMN, devido à grande demanda de empresas operadoras de telecomunicações.

[10] Application Programming Interface representa uma biblioteca de funções pré-definidas que podem ser utilizadas no desenvolvimento das aplicações, reduzindo consideravelmente o trabalho dos programadores

Referências

[Bapat, 1994] Subodh Bapat. *Object-Oriented Networks: Models for Architecture, Operations and Management.* Englewood Cliffs, New Jersey: Prentice-Hall, 1994.

[Black, 1992] Uyless Black. *Network Management Standards: The OSI, SNMP and CMOL Protocols.* New York: McGraw-Hill, 1992.

[Black, 1995] Uyless Black. *ATM: Foundation for Broadband Networks.* Englewood Cliffs, New Jersey: Prentice Hall PTR, 1995.

[BRISA, 1993] Sociedade Brasileira para Interconexão de Sistemas Abertos BRISA. *Gerenciamento de Redes: Uma abordagem de sistemas abertos.* São Paulo: Makron *Books*, 1993.

[BRISA, 1994] Sociedade Brasileira para Interconexão de Sistemas Abertos BRISA. *Arquiteturas de Redes de Computadores OSI e TCP/IP.* São Paulo: Makron *Books*, 1994.

[Freeman, 1989] Roger L. Freeman. *Telecommunication System Engineering.* New York: John Wiley & Sons, páginas 452-464, 2ª edição, 1989.

[Hayes, 1993] Stephen Hayes. Analyzing Network Performance Management. *IEEE Communications Magazine*, 31(5):52–58, May 1993.

[ITU-T, 1980] ITU-T. *Recommendation G.821 (Error Performance of an International Digital Connection Forming Part of an Integrated Services Digital Network).* ITU-T, 1980.

[ITU-T, 1984] ITU-T. *Recommendation G.826 (Digital Transmission Models).* ITU-T, 1984.

[ITU-T, 1992] ITU-T. *Recommendation X.721 (Structure of Management Information: Definition of Management Information).* ITU-T, january 1992.

[Klerer, 1993] S. Mark Klerer. System Management Information Modeling. *IEEE Communications Magazine*, 31(5):38–44, May 1993.

[Leinwand e Fang, 1993] Allan Leinwand e Karen Fang. *Network Management: A Practical Perspective.* Addison-Wesley UNIX and Open Systems Series. Addison-Wesley, 1993.

[McCarthy *et al.*, 1995] Kevin McCarthy, George Pavlou, Bhatti Saleem, e José Neuman Souza. Exploiting the power of OSI Management for the control of SNMP-capable resources using generic application level gateways. *Proceedings of the fourth international symposium on integrated network management*, páginas 440–465, 1995.

[Pavlou *et al.*, 1995] George Pavlou, Kevin McCarthy, e Graham Knight. The OSIMIS Platform: Making OSI Management Simple. *Proceedings of the fourth international symposium on integrated network management.*, páginas 480–493, 1995.

[Perrow *et al.*, 1995] Graeme S. Perrow, James W. Hong, e Hanan L. Lutfiyya. The Abstraction and Modelling of Management Agents. *Proceedings of the fourth international symposium on integrated network management*, páginas 466–478, 1995.

[Raman, 1993] Lakshmi Raman. CMISE Functions and Services. *IEEE Communications Magazine*, 31(5):46–51, May 1993.

[Rose *et al.*, 1991a] Marshall T. Rose, Julian P. Onions, e Colin J. Robbins. *The ISO Development Environment: User's Manual. Volume 1: Application Services*, 1991.

[Rose *et al.*, 1991b] Marshall T. Rose, Julian P. Onions, e Colin J. Robbins. *The ISO Development Environment: User's Manual. Volume 2: Underlying Services*, 1991.

[Rose *et al.*, 1991c] Marshall T. Rose, Julian P. Onions, e Colin J. Robbins. *The ISO Development Environment: User's Manual. Volume 3: Applications*, 1991.

[Rose *et al.*, 1991d] Marshall T. Rose, Julian P. Onions, e Colin J. Robbins. *The ISO Development Environment: User's Manual. Volume 4: The Applications Cookbook*, 1991.

[Rose, 1990] Marshall T. Rose. *The Open Book: A Practical Perspective on OSI.* Englewood Cliffs, New Jersey: Prentice-Hall, 1990.

[Souza *et al.*, 1995] José Newman Souza, Antonio Mauro B. Oliveira, e Suzana de Queiroz Ramos. Tópicos em Gerenciamento de Redes. In *MINICURSO IV, 13º Simpósio Brasileiro de Redes de computadores. Belo Horizonte: UFMG*, 1995.

[Stallings, 1993] William Stallings. *SNMP, SNMPv2, and CMIP: The Practical Guide to Network-Management Standards.* Reading, Massachusetts: Addison-Wesley, 1993.

[Stroustrup, 1991] Bjarne Stroustrup. *The C++ Programming Language.* AT&T Bell Laboratories, Murray Hill, New Jersey: Addison-Wesley, 2ª edição, 1991.

[UH Consulting, 1994a] UH Consulting. *The Q3 Agent Development Environment: Documento de Apresentação.* Rydtoften 9, DK-2750 Ballerup, Denmark. e-mail: uh@pip.dknet.dk, 1994.

[UH Consulting, 1994b] UH Consulting. *The Q3 Agent Development Environment: The Command Interpreter.* Rydtoften 9, DK-2750 Ballerup, Denmark. e-mail: uh@pip.dknet.dk, 1994.

[UH Consulting, 1994c] UH Consulting. *The Q3 Agent Development Environment: The Generic C++ Class Library (UHCpCI).* Rydtoften 9, DK-2750 Ballerup, Denmark. e-mail: uh@pip.dknet.dk, 1994.

[UH Consulting, 1994d] UH Consulting. *The Q3 Agent Development Environment: The Q3ADE Implementers Guide.* Rydtoften 9, DK-2750 Ballerup, Denmark. e-mail: uh@pip.dknet.dk, 1994.

[UH Consulting, 1994e] UH Consulting. *The Q3 Agent Development Environment: The Reference Manual.* Rydtoften 9, DK-2750 Ballerup, Denmark. e-mail: uh@pip.dknet.dk, 1994.

[Yemini, 1993] Yechiam Yemini. The OSI Network Management Model. *IEEE Communications Magazine*, 31(5):20–29, May 1993.