UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Um Modelo para a Descoberta de QoS em Redes de Computadores

por

RODRIGO UZUN FLEISCHMANN

Dissertação submetida à avaliação, como requisito
parcial para obtenção do grau de Mestre
em Ciência da Computação

Profa. Dr. Maria Janilce Bosquiroli Almeida

Orientadora

Prof. Dr. Lisandro Zambenedetti Granville

Co-orientador

Porto Alegre, dezembro de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Fleischmann, Rodrigo Uzun

Um Modelo para a Descoberta de QoS em Redes de Computadores / por Rodrigo Uzun Fleischmann. – Porto Alegre: PPGC da UFRGS, 2002.

87p.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, BR - RS, 2002. Orientador: Almeida, Maria Janilce B.; Co-orientador: Granville, Lisandro Z.

1. Redes de Computadores. 2. Gerenciamento de QoS. 3. Descoberta de QoS. 4. Script-MIB. I. Almeida, Maria Janilce Bosquiroli. II. Granville, Lisandro Z. III. Título.

Agradecimentos

Ao longo dessa jornada muitas pessoas contribuíram para a realização desse trabalho. Em primeiro lugar, agradeço minha orientadora e meu co-orientador, Janilce Almeida e Lisandro Granville, pelo incentivo e orientação. O auxílio prestado nos momentos de dúvida foram cruciais para a conclusão desse trabalho.

À minha esposa, Natalie, que acompanhou todos os detalhes dessa jornada, pela paciência dispensada, dedico essa dissertação. Aos meus pais e irmãos e demais familiares, que acompanharam os bastidores desse trabalho.

Agradeço ao meu amigo/sócio Marcelo pelas longas discussões que propiciaram uma visão mais “aberta” do modelo de desenvolvimento apresentado, ao qual atribuo o principal incentivo para uma abordagem de código aberto nesse trabalho. Finalmente, ao movimento de software livre, representados nesse trabalho pelos projetos TCL, NET-SNMP, Jasmin, PHP, Apache e, claro, GNU/Linux. A implementação desse trabalho, desenvolvida e testada na plataforma GNU/Linux, é a contribuição que ofereço para a comunidade.

Sumário

Lista de Abreviaturas

ACL	Access Control List
API	Application Programming Interface
AS	Agent Selector
AS	Autonomous System
ATM	Asynchronous Transfer Mode
CAIDA	Cooperative Association for Internet Data Anasilys
CLI	Command Line Interface
COPS	Common Open Policy Service
CORBA	Common Object Request Broker Architecture
DiffServ	Differentiated Services
DISMAN	Distributed Network Management
DNS	Domain Name Server
DS	DiffServ
DWDM	Dense DWM
FTP	File Transfer Protocol
GNU	GNU is not Unix
GUI	Graphical User Interface
HLM	High Level Manager
HTTP	HyperText Transfer Protocol
HTTPS	Secure HTTP
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IP	Internet Protocol
Jasmin	A Java Script-MIB
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MG-I	Manager Graphical Interface
MIB	Management Information Base
MIB-II	Management Information Base version II
MPLS	Multi-protocol Label Switching
NNM HP-OV	Network Node HP-Open View
PHB	Per-hop Behavior
PHP	Personal Home Pages
PIB	Policy Information Base
QAME	QoS-Aware Management Environment
QoS	Quality of Service
QoS-DM	QoS-Distributed Manager
RTP	Real-Time Transport Protocol
RPCP	Real-Time Control Protocol
RSVP	Resource Reservation Protocol
SBM	Sub-network Bandwidth Manager
SDK	Software Development Kit
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol

TCL	Tool Command Language
TCP	Transmission Control Protocol
TINA	Telecommunications Information Networking Architecture
TNM	Telecommunication Network Management
TP	Task Processor
WDM	Wavelength-Division
XML	Extensible Markup Language

Lista de Figuras

Lista de Tabelas

Resumo

A evolução das redes de computadores criou a necessidade de novos serviços de rede que fornecessem qualidade de serviços nos fluxos de dados dos usuários. Nesse contexto, uma nova área de pesquisa surgiu, o gerenciamento de QoS, onde foram apresentadas novas tarefas para o gerenciamento de recursos de rede que fornecem QoS. Uma delas, a descoberta de QoS, é responsável por identificar alvos numa rede de computadores. Um alvo é uma entidade capaz de implementar funcionalidades para o fornecimento de QoS. Essa dissertação apresenta uma proposta de modelo para a descoberta de QoS. O modelo é composto por duas partes: metodologia e arquitetura. A metodologia define os procedimentos para a realização da descoberta de QoS, enquanto a arquitetura define entidades que implementam tais procedimentos bem como o relacionamento entre essas entidades. O modelo proposto também tem por objetivo ser extensível, escalável e distribuído. Além disso, um protótipo baseado no modelo é apresentado.

Palavras-chave: redes de computadores, gerenciamento de QoS, descoberta de QoS, Script-MIB.

Title: “A MODEL FOR QOS DISCOVERY IN COMPUTER NETWORKS”

Abstract

The evolution o computer networks generated new network services that provide quality of services in users data flows. Based on this context, a new research area emerged, QoS management, where new tasks to manage network resources that provide QoS were defined. One of them, QoS discovery, is responsible for identify network targets in a computer network. A network target implements capabilities that provide QoS. This work presents a proposal of a model for QoS discovery. This model has two major parts: methodology and architecture. The methodology defines the procedures of QoS discovery, while the architecture defines the entities that implement such procedures. The proposed model also intends to be extensible, scalable and distributed. Moreover, a prototype implementation based on the model is presented.

Keywords: computer networks, QoS management, QoS Discovery, Script-MIB.

1 Introdução

The Internet Protocol's "best-effort" service has worked well so far, so why do we need to change it?

The Need for QoS (QoS Forum White Paper)

Atualmente, as redes de computadores estão presentes em praticamente todos os segmentos da sociedade e possuem complexidades e dimensões variadas, desde pequenas redes locais até redes mundiais. Acompanhando a diversificação dos usuários das redes de computadores, o surgimento de novas aplicações como telemedicina, educação à distância, voz sobre IP (*Internet Protocol*), entre outras, gerou novos perfis de tráfego que o paradigma do melhor esforço (*best-effort*) das redes IP não atende de forma adequada [XIA 99]. Neste paradigma, o tráfego de rede é processado tão rápido quanto possível, mas não há garantias temporais nem de entrega dos dados.

Com a rápida transformação da Internet numa infraestrutura comercial, a demanda por qualidade de serviço (*Quality of Service* – QoS) vem rapidamente se desenvolvendo. Neste contexto, o fornecimento de serviços de rede mais adequados para o transporte, tipicamente, de dados multimídia e de tempo real é necessário [HUT 94] para que novas aplicações operem eficaz e eficientemente. Várias pesquisas buscam soluções para o fornecimento de QoS em redes IP. Só o IETF (*Internet Engineering Task Force*) [INT 2001] coordena pesquisas em pelo menos quatro soluções: IntServ (*Integrated Services*) [SHE 97] e RSVP (*Resource reSerVation Protocol*) [BRA 97], DiffServ (*Differentiated Services*) [BLA 98], MPLS (*Multi Protocol Label Switching*) [ROS 2001] e SBM (*Subnet Bandwidth Management*) [YAV 2000]. Além destas, várias outras soluções podem ser encontradas em desenvolvimento por grupos de pesquisa [AUR 97].

O fornecimento de QoS envolve o uso de diversos mecanismos ausentes no paradigma de melhor esforço das redes IP. Técnicas de marcação de pacotes [NIC 99], disciplinas de filas [AIE 2000], conformação de tráfego [MCD 99], policiamento [AUR 97] e prevenção de congestionamento [DRA 2000] são alguns exemplos. Ainda que não exista uma arquitetura de fornecimento de QoS definitiva ou padrão, os mecanismos que auxiliam no fornecimento de QoS já podem ser encontrados em equipamentos de mercado e, como tal, podem ajudar na melhora dos serviços de rede fornecidos aos usuários, mesmo que alguns fabricantes implementem soluções proprietárias.

Apesar da melhora na qualidade dos serviços oferecidos, do ponto de vista do gerenciamento, uma rede com QoS inevitavelmente é uma rede mais difícil de ser gerenciada, pois um número maior de protocolos, serviços e informações de gerência estarão disponíveis aos administradores. As arquiteturas de fornecimento de QoS podem ser implantadas e fornecer garantias de serviços somente se os elementos de tais arquiteturas forem adequadamente configurados e monitorados. Logo, os administradores de redes devem estar cientes sobre as características das facilidades de fornecimento de QoS existentes na rede. Assim, além do gerenciamento de aspectos tradicionais (falhas, configuração, desempenho, etc.), os administradores devem gerenciar também aspectos relacionados às facilidades de QoS, ou seja, cria-se a

necessidade do *gerenciamento de* QoS [EDE 2001].

Freqüentemente, os administradores de rede utilizam os equipamentos de forma rudimentar, apenas para o encaminhamento de pacotes, ainda que muitas vezes tais equipamentos disponham de vários serviços extras. Embora vários mecanismos pudessem estar ativos nos equipamentos, tais mecanismos ficam sub-utilizados quando o administrador não consegue gerenciar tantas características em dispositivos de redes grandes e complexas. Em redes pequenas, o administrador contorna esse problema acessando cada dispositivo em particular, e ativando/configurando os mecanismos para fornecimento de QoS existentes. Essa configuração manual, orientada a dispositivos, entretanto, não pode ser realizada em redes grandes e complexas, pois o tempo gasto na tarefa é proibitivo. A automatização desse processo torna-se necessária para que o administrador passe a se ocupar com outras tarefas, como o planejamento da rede, no lugar de ficar acessando características de dispositivos manualmente.

A diversidade de tecnologias (DiffServ, IntServ, ATM, MPLS, etc.) e de equipamentos de rede (dispositivos baseados no sistema operacional GNU/Linux [FSF 99][TOU 99], CISCO IOS, 3COM, IBM, Cabletron, por exemplo) que implementam serviços de transmissão de dados com QoS é tanta que a necessidade de automatização do processo de identificação de mecanismos para fornecimento de QoS pelos dispositivos deixa de ser apenas uma facilidade para tornar-se uma necessidade fundamental a ser suportada pelos sistemas de gerenciamento. Logo, sistemas de gerenciamento de redes com QoS devem prover facilidades para a descoberta de QoS [HUS 2000].

O termo *descoberta de QoS* possui escopo amplo e interpretações variadas. Em geral, as pesquisas que estudam a descoberta de QoS investigam formas de permitir que aplicações conheçam os serviços de rede e os QoSs associados. Entretanto, pouco tem sido verificado em relação à descoberta de QoS do ponto de vista do gerenciamento de redes. Este trabalho justamente investiga a descoberta de QoS através deste último ponto de vista. Neste contexto, facilidades para descoberta de QoS disponibilizadas em um ambiente de gerenciamento podem auxiliar o administrador em dois principais momentos: durante a implantação de QoS e durante a manutenção de QoS. Entende-se por implantação de QoS as tarefas realizadas para se implantar uma arquitetura de fornecimento de QoS em uma nova rede ou em uma rede existente. Neste último caso, a rede existente deve ser não apenas mapeada, mas as facilidades já encontradas em cada dispositivo devem ser identificadas (ainda que tais facilidades estejam desativadas). Depois que uma arquitetura de QoS é implantada, a manutenção de QoS é responsável por determinar como a arquitetura existente deve se comportar. A adição, por exemplo, de novos roteadores em uma estrutura já implantada pode melhorar serviços porque novas rotas serão formadas. A descoberta de QoS, neste caso, permite que os serviços existentes nos novos roteadores possam ser identificados.

A descoberta de QoS está relacionada também com a descoberta de topologia em redes. Apesar de algumas soluções de descoberta de topologia já estarem presentes nas plataformas de gerenciamento mais importantes, o mesmo não acontece com a descoberta de QoS. Entretanto, nas novas redes de computadores uma solução para descoberta de QoS deveria possuir as seguintes características:

- Ser integrada a um ambiente de gerenciamento de QoS;
- Ser escalável, para suportar a descoberta de facilidades de QoS em redes de

tamanhos diferentes em escalas temporais adequadas ao gerenciamento;

- Ser extensível, possibilitando que o conjunto de facilidades de descoberta de QoS possa ser aumentado quando um novo tipo de mecanismo de QoS tornar-se disponível;
- Ser distribuída, para que a descoberta possa ser realizada por processos ativos em pontos diversos da rede gerenciada, sem depender de um ponto de processamento único.

As soluções existentes atualmente não possuem todas as características da lista acima, o que acaba por motivar o estudo, definição e implementação de uma solução para descoberta de QoS. Essa dissertação de mestrado apresenta então uma solução para descoberta de QoS que apresenta as características listadas anteriormente. Tal solução é também integrada ao ambiente de gerenciamento QAME (*QoS-Aware Management Environment*) [GRA 2001] [GRA 2001a]. Um protótipo foi desenvolvido seguindo a arquitetura para descoberta de QoS proposta neste trabalho. Assim, são objetivos desta dissertação:

- Investigar o estado da arte e os trabalhos relacionados à descoberta de QoS em geral, e em relação à descoberta de QoS específica em sistemas de gerenciamento de redes;
- Propor uma solução para descoberta de QoS em redes IP, integrada ao ambiente QAME, e que seja escalável, extensível e distribuída;
- Apresentar a implementação e análise de um protótipo que segue a solução proposta.

O presente trabalho encontra-se organizado da seguinte forma: no capítulo 2 é revisado o gerenciamento de QoS e os trabalhos relacionados à descoberta de QoS especificamente. Também são apresentados a contextualização deste trabalho e o estado da arte, juntamente com a definição do problema investigado. O capítulo 3 apresenta a solução para descoberta de QoS proposta, onde os diversos módulos da arquitetura são descritos. No capítulo 4 é apresentado o protótipo desenvolvido de acordo com a solução proposta no capítulo 3. O protótipo é descrito de acordo com a lista principal de características anteriormente apresentada. Por fim, o capítulo 5 apresenta as conclusões e os possíveis trabalhos futuros.

2 Gerência de QoS e definição do problema

Uma necessidade atual em redes de computadores é a capacidade de transmissão de dados a altas taxas e com restrições temporais estritas. Infelizmente, a maioria das redes implantadas, incluindo a Internet, não possibilita essa adequada transmissão de informações porque os mecanismos utilizados não fazem diferenciação de tráfego. Por exemplo, uma aplicação de correio eletrônico, que possui requisitos de transmissão diferentes de uma aplicação de videoconferência, concorre igualmente pelos recursos de rede (largura de banda, principalmente).

Neste capítulo será verificado o estado da arte em relação ao fornecimento de QoS em redes de computadores e, principalmente, sua respectiva necessidade de gerenciamento. Em particular, a descoberta de QoS é de especial interesse no contexto desta dissertação. Assim, são também apresentados os trabalhos relacionados com tal descoberta. A proposta apresentada nesta dissertação não é isolada, mas sim incluída em um contexto maior. A seção 2.3 mostra então a contextualização deste trabalho. Por fim, o problema investigado é mais precisamente apresentado em relação ao estado da arte e as soluções para descoberta de QoS existentes na atualidade. A definição do problema investigado é apresentada na seção 2.4, ao final deste capítulo.

2.1 Gerenciamento de QoS

A Internet está sendo utilizada para negócios e por comunidades de usuários com amplas expectativas de fornecimento de diversos serviços da infraestrutura da rede, como o gerenciamento diário de negócios globais por parte de muitas companhias. Essas companhias concordariam em pagar um custo substancialmente alto para obter melhores níveis de serviços compatíveis com suas necessidades [DOV 99]. Da mesma maneira, há muitos usuários que pagariam por uma Internet de rápido acesso de maneira que fosse possível o uso de aplicações emergentes, como telefonia sobre IP [ROS 2000] e videoconferência. Ao mesmo tempo, há milhões de usuários que querem pagar uma mínima quantia para utilizar serviços mais elementares, como correio eletrônico ou simples navegação na Web.

Tais serviços elementares, que compõem o conjunto de aplicações disponíveis no dia a dia do usuário da Internet, tiveram uma rápida evolução. Há poucos anos as aplicações da Internet se resumiam a praticamente correio eletrônico, transferência de arquivos via FTP [POS 85] ou serviços de notícias (*newsgroup*) [KAN 86]. Em contraste a esse cenário, atualmente as aplicações da Internet têm necessidades de diversos serviços devido à ampla escala de tipos de informações, incluindo voz, música, vídeo e gráfico. Com essa transformação nas necessidades dos usuários e das aplicações da Internet, há uma crescente demanda para a substituição do paradigma de melhor esforço do IP por um modelo que atenda os requisitos de tráfegos diferenciados, ou seja, um modelo que ofereça QoS para o usuário [DOV 99].

Embora as aplicações possuam diferentes exigências de QoS, uma tendência é o fornecimento de classes de serviços de rede ao invés de soluções que atendam solicitações de fluxos individualmente. Pode-se comparar classes de serviço de rede

com as classes do transporte aéreo, nas quais a primeira classe de um vôo possui equivalência com a classe de serviço de rede que oferece o melhor QoS. Assim como no transporte aéreo, onde cada classe oferece um conjunto de serviços que é comum para um grupo específico de passageiros, uma classe de serviço de rede fornece o mesmo perfil de tráfego para todos os usuários da classe. O fornecimento de classes de serviços de rede não pretende atender todos os fluxos solicitados na Internet, mas oferecer um ambiente de serviços mais eficiente e viável, na medida que os diversos perfis de tráfego solicitados são enquadrados em perfis pré-definidos [NIC 99]. Assim, uma classe de serviço poderá fornecer serviços de Internet previsíveis para companhias que têm seus negócios baseados na Web. Tendo garantia de um serviço estável e confiável, as transações na Web se tornariam ágeis e os clientes criariam uma boa impressão sobre os *sites* Web das empresas, o que poderia aumentar o volume de negócios. Outra classe irá fornecer baixa latência e baixa variação do atraso para aplicações como voz sobre IP e videoconferência. As companhias pagariam por videoconferência de alta qualidade para economizar tempo e custos de viagens. Por último, serviços do tipo *best-effort* irão permanecer para aqueles usuários que precisam de conectividade básica.

Mecanismos de fornecimento de QoS

A necessidade de mecanismos para fornecimento de QoS tem sido fortemente discutida. Uma opinião é que as fibras óticas e as tecnologias baseadas em multiplexação por divisão de comprimentos de ondas, tais como WDM (*wavelength-division multiplexing*) e DWDM (*dense wavelength-division multiplexing*) tornarão a largura de banda tão abundante e barata que o QoS será automaticamente fornecido. Uma outra visão é a de que não importa quanta largura de banda seja oferecida pelas redes, pois sempre haverá aplicações que irão consumi-las. Nesta situação, mecanismos ainda serão necessários para fornecer QoS [JAI 92]. Nesse trabalho é assumido que mesmo que a oferta de largura de banda seja abundante e barata, ainda assim facilidades de fornecimento de QoS estarão presentes na rede de interesse. Então, é assumido que mecanismos são definitivamente necessários para fornecer QoS.

Além disso, o simples oferecimento de largura de banda sem um controle adequado poderia levar a uma situação onde houvesse desperdício dos recursos de rede, indesejável para a implementação de um serviço eficiente. Por exemplo, se um roteador for periodicamente sobrecarregado por curtas e intensas rajadas de tráfego, ele poderá, eventualmente, atrasar ou descartar pacotes nesse período. Embora a sobrecarga seja temporária e na maioria do tempo o roteador atenda às transmissões de pacotes rapidamente, nesses períodos de rajadas haveria uma degradação do serviço, o que prejudicaria outros fluxos, atrasando a transmissão de pacotes, o que seria prejudicial para a aplicação. O uso de um esquema de descarte de pacotes para aqueles fluxos que ultrapassem um determinado limiar por um período de tempo ou uma simples priorização de tráfego, são algumas das alternativas que poderiam evitar que as rajadas ocupassem quase que por completo os recursos do roteador. Essa visão é reforçada pelo simples fato da maioria dos fabricantes apresentarem mecanismos de fornecimento de QoS nos seus produtos [CIS 2001] [LUC 2001] [JUN 2001] [3CO 2001]. Tais mecanismos, inexistentes nas redes IP tradicionais, incluem marcação de pacotes, disciplinas de filas, conformação de tráfego, policiamento, prevenção de congestionamento e roteamento baseado em restrições (*constraint-based routing*) [CRA 98].

A marcação de pacotes é caracterizada por atribuir valores específicos a um campo dos pacotes. Esse valor é posteriormente analisado e uma ação é tomada baseada nesse valor, como a priorização do pacote numa fila de um roteador. A conformação de tráfego propõe adequar fluxos ao perfil de tráfego contratado. Assim, um conformador de tráfego é um processo que "atrasa" o repasse de pacotes que excedem a largura de banda acordada, por exemplo. O policiamento (*policing*), mais punitivo que a conformação de tráfego, consiste em descartar pacotes dos fluxos que não obedeçam ao perfil de tráfego acordado. Já o roteamento baseado em restrições procura por rotas que atendam alguns requisitos, tais como largura de banda mínima ou atraso. Cada mecanismo isoladamente já oferece grandes vantagens e melhores tratamentos dos fluxos da rede. Porém, o uso coordenado de tais mecanismos, organizados em arquiteturas, permite o fornecimento de serviços mais controlados, o que aumenta as garantias para o usuário, isto é, fornecem QoS. O uso combinado de policiamento no tráfego ingresso e conformação de tráfego no egresso, por exemplo, auxiliam em aplicações como telefonia ou videoconferência sobre IP, onde o tráfego é tipicamente constante e os atrasos são indesejados. O policiamento descarta os pacotes dos fluxos que estejam excedendo a largura de banda acordada de maneira que os fluxos que estejam respeitando o perfil não sejam prejudicados e a conformação de tráfego não entrega os pacotes antes do tempo de entrega.

Em relação às arquiteturas de fornecimento de QoS, destacam-se duas propostas do IETF [INT 2001]: a arquitetura de serviços integrados (IntServ) [SHE 97] e a arquitetura de serviços diferenciados (DiffServ) [BLA 98].

IntServ

O modelo IntServ propõe duas classes de serviços adicionais ao serviço de melhor esforço. São eles:

- Serviço garantido (*guaranteed service*) [SHE 97a]: para aplicações com limiares de atrasos pré-determinados.
- Serviço de carga controlada (*controlled-load service*) [WRO 97]: destinado a aplicações que solicitam serviços confiáveis e melhores que o *best-effort*.

A filosofia desse modelo é que há uma demanda por roteadores capazes de reservar recursos de maneira a fornecer QoS para *streams* de pacotes ou fluxos específicos. Assim, é preciso que os roteadores armazenem informações de estado dos fluxos [BRA 94]. Para implementar esse modelo, a arquitetura foi projetada com três componentes: protocolo de sinalização (*signaling protocol*), rotina de controle de admissão (*admission control routine*) e escalonadores de pacotes (*packet scheduler*). O protocolo de sinalização utilizado é o RSVP [BRA 97], responsável por alocar rotas e reservar recursos.

O IntServ representou uma mudança fundamental na arquitetura da Internet, que era baseada apenas em um modelo onde as informações de estado dos fluxos deveriam estar nos sistemas finais [CLA 88]. Embora inovadora, a arquitetura IntServ possui alguns problemas:

- A quantidade de informações armazenadas nos roteadores aumenta proporcionalmente com o número de fluxos. Isso implica em um aumento significativo de armazenamento e processamento nos roteadores, o que não

permite uma escalabilidade, considerando o cenário típico da Internet.

- Os requisitos para a implementação de IntServ nos roteadores são elevados, pois todos o roteadores devem implementar o RSVP (ou outro mecanismo para reserva de recursos), controle de admissão, classificação MF e escalonamento de pacotes.

A ubiqüidade dos roteadores é necessária para a implementação de serviços garantidos. Além disso, a implementação de serviços de carga controlada é possibilitada pelo uso de RSVP nos nodos que representam um gargalo no domínio e a criação de túneis de mensagens RSVP nos demais nodos do domínio.

DiffServ

Devido à dificuldade de implementar e implantar IntServ e RSVP, DiffServ é apresentado. Essencialmente baseado num esquema de priorização de tráfego agregado, a arquitetura DiffServ é composta de elementos funcionais nos nodos da rede, incluindo um pequeno conjunto de comportamento de encaminhamento por salto (*per-hop behavior* - PHB), classificação de pacotes e funções de condicionamento de tráfego (medição, marcação, conformação e policiamento). Para que um usuário utilize um serviço DiffServ de seu provedor de serviços de Internet (*Internet service provider* – ISP), ele deve antes estabelecer um acordo de nível de serviço (*service level agreement* – SLA). A priorização dos agregados DiffServ é baseada na marcação do campo DSCP (*DiffServ CodePoint*) [NIC 98] de pacotes IP executada previamente. Essa marcação pode ser feita pelo cliente [GRA 2000] ou tipicamente pelos roteadores folhas. No ingresso dos domínios DS (DiffServ), os pacotes são classificados, marcados, policiados e possivelmente conformados, baseados no SLA acordado. Alguns exemplos de serviços oferecidos pela arquitetura DiffServ são *premium service*, *assured service* e *olympic service* [NIC 99].

Considerações Finais

Como visto, em ambas as arquiteturas de fornecimento de QoS, IntServ e DiffServ, os mecanismos e protocolos envolvidos permitem uma variedade de serviços bastante ampla. Porém, nenhum mecanismo ou protocolo é definido de maneira que o usuário final possa solicitar o QoS de maneira automática. O RSVP aloca recursos fim a fim, mas não é explicitamente dito quem informa ao protocolo quais recursos serão alocados e quais fluxos permitir. Num contexto distribuído, onde outros fatores devem ser considerados para implementar o QoS, como a restrição a alguns fluxos através de filtros implementados em *firewalls*, a carência de integração entre as tarefas de gerenciamento da rede pode causar problemas durante o fornecimento de QoS. Um usuário poderia solicitar um serviço para executar uma aplicação do tipo *audio-streaming* e, durante a alocação dos recursos, não é permitida a utilização de uma porta específica pelo *firewall* do ISP. Assim, ao tentar executar a aplicação, o usuário não terá êxito, devido a um problema que poderia ser evitado se as informações estivessem disponíveis. Além disso, as arquiteturas não definem explicitamente como o QoS fornecido será monitorado e, por conseqüência, nem como essas informações poderiam ser utilizadas para o planejamento da rede, por exemplo.

Com a introdução de novos protocolos e mecanismos para a implementação de

arquiteturas de fornecimento de QoS, além do gerenciamento dos aspectos de gerenciamento tradicional, é necessário o gerenciamento desses novos elementos apresentados pelas arquiteturas de fornecimento de QoS, o que gera a necessidade de arquiteturas de gerenciamento de QoS [EDE 2001]. Neste contexto, um dos argumentos importantes desta dissertação é o de que o correto fornecimento de QoS requer não apenas uma arquitetura de QoS, mas também um sistema de gerenciamento desta arquitetura, isto é, existe a necessidade do gerenciamento de QoS.

2.2 Descoberta de QoS e trabalhos relacionados

Para gerenciar as funcionalidades de redes envolvidas no fornecimento de QoS, os administradores devem realizar várias tarefas, além das tarefas de gerenciamento tradicional (gerenciamento de falhas, contabilização, configuração, desempenho e segurança) [STA 98]. As tarefas relacionadas à QoS devem ser realizadas utilizando facilidades fornecidas pelo ambiente de gerenciamento de rede. Tais tarefas incluem descoberta de QoS, implantação de QoS, manutenção da operação dos elementos envolvidos no fornecimento de QoS, monitoração de QoS, análise de QoS e visualização de QoS [GRA 2001]. A implantação de QoS consiste na tarefa de escolher e implantar as soluções de QoS para uma rede. Depois que a arquitetura de QoS é definida e implantada, os serviços de QoS estão prontos para serem oferecidos para os usuários. A manutenção da operação de QoS é classificada como qualquer procedimento que mantenha os serviços de QoS operando. Descoberta de Qos é definida como a tarefa de procurar na rede por características capazes de auxiliar ou melhorar o fornecimento de QoS. A monitoração de QoS informará as diferenças entre o QoS desejado e o obtido para determinar os pontos de degradação, por exemplo. A análise de QoS permite ao administrador antecipar os problemas e resolvê-los tão logo seja possível, bem como traçar as tendências da rede, o que pode auxiliar num planejamento mais preciso da mesma. A visualização de QoS auxilia o gerente de rede compreender melhor, através de recursos gráficos, o QoS implantado, ao invés de olhar extensas tabelas de descrição dos elementos de QoS, por exemplo.

Descoberta de QoS e adaptação de aplicações ao QoS fornecido

O termo descoberta de QoS possui escopo amplo e diversas interpretações na literatura. Em ambientes de transmissão sem fio (*wireless communication*), onde variações na qualidade de transmissão em função da carga do canal e distância do ponto de acesso à rede são comuns, descobrir e fornecer informações a respeito da qualidade do canal de transmissão para a aplicação permitiria que a aplicação pudesse se adaptar a essas situações, de acordo com as condições de transmissão apresentadas [MIT 2000]. Pode-se estender para um modelo mais genérico, onde o meio de transmissão não garanta QoS. Comparando com aplicações multimídia que utilizam RTP [SCH 96] para transmitir as mídias sobre um canal que não garanta o QoS desejado, uma transmissão adaptativa à qualidade do canal de transmissão é necessária para minimizar a falta de QoS [GRA 2001b]. Nesse caso, o protocolo RTCP [SCH 96] fica continuamente verificando as condições da rede para adequar o tráfego RTP. Associado a isso, seria interessante que uma aplicação servidora descobrisse as características de um receptor [MIT 2000]. Assim, seria possível que a aplicação se adaptasse aos recursos do cliente.

Analogamente, seria interessante que um cliente pudesse saber as características relativas a QoS de um servidor de maneira que ele pudesse escolher o servidor mais adequado.

Descoberta de QoS em caminhos de rede

Na seção anterior verificou-se a necessidade de identificar as características de QoS da entidade par de comunicação, o que não teria êxito se o canal de transmissão fosse inadequado. Não há mecanismos robustos e expansíveis para todas as arquiteturas de computadores com suporte a QoS que forneça descoberta de caminhos de rede com atributos específicos de desempenho [HUS 2000]. A hipótese que as arquiteturas DiffServ e IntServ assumem é que a rota padrão utilizada no paradigma de melhor esforço é a escolhida, onde o caminho é ou não capaz de sustentar a carga de serviço solicitada. Tais arquiteturas não consideram a possibilidade de uma rota alternativa de maior custo (maior descarte, por exemplo) ser capaz de atender um fluxo solicitado, por exemplo. Baseado nisso, como uma aplicação de um determinado *host* poderia escolher o melhor caminho de sua conexão? Ela poderia solicitar a um serviço de rede um conjunto de caminhos possíveis (candidatos) para então escolher o que melhor atenda suas necessidades, baseado em alguma funcionalidade de roteamento com QoS.

Descoberta de QoS em arquiteturas computacionais distribuídas

Na plataforma TINA [STE 97], baseada em CORBA [OMG 2001], a descoberta de QoS fim-a-fim (isto é, descoberta em todos os nodos que participam de uma transmissão) é vista como a base do gerenciamento de QoS de serviços multimídia. Com esse objetivo, uma arquitetura em três camadas, que trata as perspectivas do usuário, dos sistemas finais e da rede, foi proposta [VAN 2000]. Nessa arquitetura, as três camadas devem interoperar de maneira que o equilíbrio de QoS seja atingido. Primeiramente, a "camada de especificação, apresentação e parametrização de QoS" é responsável por mapear o QoS desejado pelo usuário em um modelo baseado em moléculas biológicas. O modelo biológico, chamado QoS-M (QoS molecule), foi utilizado devido à analogia feita entre os elementos de QoS e átomos, bem como entre moléculas e serviços de rede. O elemento de QoS com suporte ao protocolo RTP seria um átomo de uma molécula, que representa um serviço de transmissão de vídeo no QoS-M, por exemplo. Assim, uma reação entre moléculas poderia representar o mapeamento de solicitações de usuários para serviços de rede oferecidos pela arquitetura. A camada de negociação e verificação de QoS utiliza o QoS-M (informado pela camada de especificação, apresentação e parametrização) e verifica a possibilidade de implementar o QoS na rede de maneira que o equilíbrio de QoS no fornecimento de serviços seja garantido, ou seja, nenhum usuário terá seu serviço degradado em função da adição de um novo serviço. Enfim, a camada de implementação de QoS, através de protocolos específicos, implementa o QoS desejado. A descoberta de QoS é implementada como resultado da interoperabilidade das três camadas da arquitetura possibilitando o fornecimento de QoS fim-a-fim.

O surgimento de *grids* computacionais globais mudou os paradigmas tradicionais de computação distribuída. Agora é possível que uma tarefa computacional seja executada por recursos de computação e comunicação autônomos, distribuídos e

heterogêneos [XU 2001]. Nesse contexto, um recurso de comunicação ou computação pode ser descoberto sob demanda nos *grids*, onde múltiplas instâncias de um recurso (espaço de armazenamento e ciclos de processador, por exemplo) são encontrados e uma deles será selecionado dinamicamente com base na sua ocupação corrente. Uma etapa seguinte a essa é o surgimento de *grids* de serviços globais, baseados em *grids* computacionais, como um mercado de serviços distribuídos em nível de aplicação que agregam serviços aos recursos de computação e comunicação. Exemplos de tais serviços incluem serviços de laboratórios científicos virtuais e comunidades virtuais.

Um desafio num *grid* de serviço disposto em uma ampla área é construir um ambiente (*framework*) para um serviço de descoberta escalável e eficiente. Os principais requisitos de sistemas tradicionais de descoberta incluem:

- A distribuição e o gerenciamento das informações do serviço não deveriam causar sobrecargas adicionais nem criar gargalos com um eventual aumento de serviços disponíveis ou solicitações de usuários;
- A resposta de uma solicitação de serviço deve ser razoavelmente rápida;
- O resultado da descoberta deveria escolher o melhor recurso, caso houvesse mais de uma opção.

Embora esses ambientes forneçam soluções escaláveis para gerenciamento e distribuição de informações em grandes áreas, foram identificados dois problemas [XU 2001]:

- Tempo de resposta da solicitação potencialmente longo;
- Nenhuma consideração explícita sobre o QoS fornecido pelo provedor de serviços (*Service Provider* – SP).

Assim, baseado num ambiente tradicional de descoberta de serviços em grandes áreas, foi proposto um ambiente (*framework*) que adota uma arquitetura escalável formada por uma hierarquia de Servidores de Descoberta (*Discovery Servers* - DS) [XU 2001]. Para solucionar os problemas mencionados acima foram introduzidas duas funcionalidades:

- Disposição das informações relativas à QoS para os usuários;
- Esquema de *caching* e propagação dos resultados de descoberta com informações de QoS na hierarquia dos Servidores de Descoberta.

Com esse novo ambiente e as novas funcionalidades, a descoberta dos serviços considerando informações de QoS passou a relatar, por exemplo, o grau de previsibilidade de QoS nos serviços, embora tenha adicionado latência e sobrecarga de mensagens no processo de descoberta.

Plataformas e Ferramentas de Gerenciamento

A evolução das plataformas e ferramentas de gerenciamento tem auxiliado bastante os gerentes de rede nas suas tarefas diárias, principalmente no gerenciamento de falhas e configuração. Algumas plataformas possuem navegação gráfica em duas ou três dimensões, como UNICENTER TNG [UNI 2001] e *Network Node Manager HP Openview* (NNM HP-OV) [NET 2001], que facilitam o processo de visualização da topologia de rede. Outras enfatizam a padronização na comunicação com os

dispositivos, como o Orchestream [ORC 2001] e o OpenNMS [OPE 2001]. Alguns fabricantes de dispositivos de rede fornecem ferramentas específicas para soluções específicas de seus equipamentos (como CISCO [CIS 2001a], 3COM [3CO 2001a] e IBM [IBM 2001]), o que dificulta a integração desses dispositivos com os de outros fabricantes e dessas ferramentas com outras. Essas soluções proprietárias comumente são implementadas devido a características não abordadas ou com definição vaga nas normas internacionais. Enfim, há ainda algumas ferramentas que fornecem facilidades para expansão das suas funcionalidades como o Scotty [SCO 2001]. Como novas características foram adicionadas aos dispositivos de rede para fornecerem QoS, tais ferramentas foram atualizadas. Entretanto, a maioria delas possui alguma deficiência em relação aos requisitos de descoberta de QoS.

A CAIDA (Cooperative Association for Internet Data Analisys) [CAI 2001] desenvolveu uma ferramenta chamada *skitter* [SKI 2001] para avaliação dos principais *backbones* da Internet, de maneira a auxiliar no planejamento e estruturação da mesma. Para tal, essa ferramenta mapeia a topologia dos *backbones*, mede RTT (*round trip time*), rastreia alterações em rotas persistentes, entre outras funcionalidades. Vários monitores *skitters* [SKI 2001a] espalhados por diversos continentes realizam consultas com baixa freqüência, para posteriormente gerar as informações a respeito dos *backbones*, que podem ser acessadas por HTTP. Embora essa ferramenta não implemente uma tarefa para identificar o QoS fornecido pelos dispositivos de rede, essa funcionalidade poderia fornecer informações sobre os dispositivos consultados, para auxiliar na elaboração das funções de avaliação dos *backbones*, bem como na análise das informações consultadas pelas tarefas existentes. Atualmente a tecnologia utilizada pelo *skitter* está licenciada para CAIMIS Inc [CAI 2001a].

O projeto OpenNMS, baseado na filosofia *open source* [OPE 2001a], possui uma plataforma de gerenciamento com o mesmo nome que implementa uma arquitetura centralizada com transações de informações baseadas em XML (*eXtensible Markup Language*) [EXT 2000], tendo grande parte de sua arquitetura implementada em JAVA [SUN 2001]. Um ponto em comum entre essa plataforma com a proposta desse trabalho é a identificação dos dispositivos. A ferramenta dessa plataforma que identifica os dispositivos se baseia na descrição do equipamento fornecida pelo objeto *sysDescr* da MIB-II [MCC 91]. Porém, a identificação não oferece, no seu atual estágio, informações sobre o QoS fornecido pelo dispositivo, nem utiliza outras abordagens para identificar os dispositivos.

Scotty é um pacote de software *open source* que permite a implementação de softwares específicos de gerenciamento de redes através de bibliotecas (*Tnm TCL*), que fornecem acesso a informações de gerenciamento de rede básicas (tais como informações oferecidas por SNMP). Acompanha este pacote um editor gráfico (TkIned), que fornece um ambiente para um sistema de gerenciamento de rede extensível. O grande benefício dessa ferramenta é poder adicionar funcionalidades utilizando a linguagem TCL (*Tool Command Language*), de alto nível e orientada a *string*, embora ela só ofereça descoberta de topologia.

A ferramenta Orchestream realiza descoberta automática de dispositivos com QoS baseados em DiffServ e MPLS, utilizando protocolos e MIBs padronizados. Embora a ferramenta exporte uma API (*Application Program Interface*) em CORBA, a mesma não permite a adição de novos módulos de *software* desenvolvidos por terceiros para a descoberta de outros serviços como SBM ou IntServ, além de possuir uma

arquitetura centralizada. O fabricante informa que está avaliando a possibilidade desse tipo de integração.

A plataforma NNM HP-OV não implementa na sua concepção original qualquer forma de descoberta de QoS. Porém, é possível ampliar as funcionalidades da plataforma adquirindo-se um SDK (*System Developtment Kit*) junto ao fabricante. Embora o NNM HP-OV ofereça uma solução proprietária, pelo menos oferece uma alternativa. De outro modo, as ferramentas de gerenciamento da CISCO, e em geral dos fabricantes de dispositivos de rede, não fornecem nenhuma possibilidade de ampliar funcionalidades, a menos que seja implementada pelo próprio fabricante. Apesar dessa restrição, a CISCO oferece um conjunto de ferramentas de gerenciamento (*QoS Policy Manager* – QPM [CIS 2001b], *Service Level Agreement Manager* - SLAM [CIS 2001c], *Internetwork Perfomance Monitor* - IPM [CIS 2001d]e *QoS Device Manager* – QDM [CIS 2001e]) que implementa descoberta de QoS em seus dispositivos e que não funciona com dispositivos de outros fabricantes, mesmo que eles utilizassem protocolos e MIBs padronizados.

2.3 Contextualização do trabalho

Esse trabalho está diretamente relacionado ao ambiente de gerenciamento QAME (*QoS-Aware Management Environment*) [GRA 2001] [GRA 2001a] que propõe uma arquitetura para gerenciamento integrado de QoS e define seis tarefas de gerenciamento: implantação, manutenção, descoberta, monitoração, análise e visualização de QoS. Um dos principais objetivos desse trabalho é estudar os aspectos relacionados à descoberta de QoS considerando o ambiente QAME. No QAME os alvos são elementos ativos que influenciam diretamente no QoS final presente na rede. Cada dispositivo de rede pode possuir diversos alvos que influenciam nos serviços de QoS existentes. Por exemplo, em um roteador, cada interface do dispositivo é um alvo. Assim, os alvos são os elementos finais que efetivamente implementam a arquitetura de fornecimento de QoS.

A descoberta de QoS, nesse contexto, consiste na identificação, dentro da rede gerenciada, dos alvos que implementam a arquitetura de fornecimento de QoS. Esta atividade, por si só, já se configuraria em um processo de descoberta de QoS, já que os elementos que podem melhorar o QoS fornecido (os alvos) são descobertos. Entretanto, por uma questão de integração do processo de descoberta de QoS (proposto nesta dissertação) com o ambiente QAME, o trabalho realizado apresenta funcionalidades extras necessárias. Tais funcionalidades extras, que complementam a descoberta, estão relacionadas com a definição de associações entre os alvos descobertos e outros elementos.

A primeira associação necessária é a correspondência entre um alvo descoberto e o dispositivo de rede que contém este alvo. Esta associação é necessária porque a programação de um alvo (por exemplo, na definição de prioridades dos fluxos em uma fila de uma interface) só é possível através do acesso ao dispositivo que contém o alvo. O segundo tipo de associação é a classificação de um alvo de acordo com sua funcionalidade. Alvos diferentes, mas que funcionalmente executam tarefas similares, são classificados como alvos de um mesmo tipo. Por exemplo, a priorização de fluxos na arquitetura de serviços diferenciados é diferente da priorização de fluxos da arquitetura de serviços integrados. Entretanto, os alvos diferentes que executam as

priorizações nas duas arquiteturas podem ser classificados como alvos de um tipo "priorização de tráfego". Outros possíveis tipos de alvos seriam "policiamento", "conformação de tráfego" e "filtros de segurança", entre outros. A última associação a ser executada é a determinação da solução de fornecimento de QoS da qual o alvo faz parte. Tipicamente um alvo pode ser classificado como sendo parte de uma solução de serviços diferenciados ou integrados, mas outras soluções (como MPLS, ATM, SBM, etc.) também podem ser verificadas. Um mesmo alvo pode fazer parte de soluções diferentes ao mesmo tempo: um alvo pode ser capaz de fornecer facilidades tanto nos serviços diferenciados quanto nos serviços integrados.

2.4 Problema investigado

A seção anterior apresentou as características de descoberta de QoS considerando o ambiente em que a solução apresentada nessa dissertação fará parte. Na seção 2.2 foram analisados os trabalhos relacionados e o estado da arte em relação à descoberta de QoS. Como pôde ser visto, o termo descoberta de QoS possui variadas interpretações e, na maioria das vezes, seu propósito visa a atender as necessidades de informações sobre o QoS disponível e oferecido pela rede do ponto de vista das aplicações finais. Para alguns, descoberta de QoS deveria informar a qualidade do canal de transmissão de maneira que a aplicação pudesse se adaptar ao estado de canal, fornecendo um serviço mais adequado para o usuário [GRA 2001b]. Como visto, as arquiteturas computacionais distribuídas compreendem descoberta de QoS de maneiras distintas: a arquitetura TINA [STE 97] verifica a possibilidade de implementar um serviço solicitado pelo usuário baseado em seu modelo QoS-M, enquanto a arquitetura DS [XU 2001] procura na rede por recursos de computação ou comunicação em *grids* de processamento para que possam ser utilizados na execução de tarefas complexas. Em todas as situações, a descoberta de QoS não é explorada sob o ponto de vista do gerenciamento de rede. Não é informado à gerência da rede como os dispositivos de rede se inserem nesse contexto, isto é, quais os protocolos e informações utilizados e quais os benefícios que tais dispositivos ofereceriam ou poderiam usufruir. Por exemplo, no caso onde a aplicação se adapta às condições de comunicação, um processo de descoberta de QoS poderia ter previamente mapeado, nas estações de gerenciamento da rede, os alvos dos dispositivos e, com essas informações, seria possível ao administrador da rede escolher um canal de comunicação mais adequado antes do início da transmissão, de maneira que a aplicação realizasse uma quantidade menor de tarefas para se adaptar.

Embora a necessidade de uma tarefa que implemente descoberta de QoS esteja identificada, as plataformas e ferramentas de gerenciamento de redes atuais não atendem essa demanda. A maioria delas possui processos de descoberta apenas de topologia de rede. Como regra geral, tais processos são centralizados, o que para redes grandes pode gerar um tráfego de descoberta razoavelmente elevado, algo indesejável, na medida que o tráfego de gerenciamento deve ser o menor possível. Nesta situação, processos de descoberta distribuídos poderiam ser utilizados. Estes se caracterizam principalmente por colocar módulos de software "perto" dos alvos de interesse, confinando assim o tráfego de descoberta a porções específicas da rede. Infelizmente, entretanto, processos de descoberta de topologia distribuídos não são freqüentemente encontrados e utilizados. O autor deste trabalho acredita que uma possível razão para tal fato esteja relacionada com a complexidade de implementação do problema, o que poderia

inviabilizar a tarefa.

Ainda sobre a implementação de descoberta de QoS nas ferramentas atuais existem situações ainda mais complexas. Ferramentas como o *skitter* [SKI 2001], por exemplo, ainda não consideram informações de QoS na realização das suas tarefas, o que pode impedir a geração de relatórios mais precisos. Se uma determinada rota R1, por exemplo, não garante largura de banda mínima para fluxos de dados do tipo *best-effort*, e num determinado período essa rota fica com sua capacidade máxima ocupada por fluxos DiffServ, o administrador da rede, conhecendo as características de seus dispositivos, poderia determinar uma rota R2 alternativa, onde os fluxos *best-effort* não fossem totalmente descartados. Já a Cisco oferece uma solução extremamente proprietária de descoberta, o que impossibilita a interoperabilidade com outras plataformas de gerenciamento de rede. Por fim, algumas ferramentas implementam parcialmente descoberta de QoS como o OpenNMS e o Orchestream.

Assim, o contexto atual aponta para a inexistência de uma solução de descoberta de QoS que seja, ao mesmo tempo:

- Distribuída, evitando sobrecargas na rede devido ao tráfego gerado pelo processo de descoberta;
- Escalável, de forma que o processo de descoberta se acomode adequadamente em redes de tamanhos diferentes; e
- Extensível, para que novas funcionalidades de fornecimento de QoS inseridas na rede possam ser também descobertas, mesmo que inicialmente não previstas na solução.

Logo, esta dissertação de mestrado busca a definição, implementação e análise de uma solução para descoberta de QoS que possua efetivamente as características ausentes nas soluções atuais, e que seja integrada ao ambiente de gerenciamento de QoS QAME, introduzido na seção anterior.

3 Modelo de descoberta de QoS proposto

No capítulo anterior analisou-se o estado da arte da descoberta de QoS e os objetivos desse trabalho foram apresentados. Esse capítulo propõe um modelo para descoberta de QoS como solução para o problema apresentado anteriormente.

A maioria das plataformas de gerenciamento atuais possui mecanismos de descoberta de topologia. Se a descoberta de QoS também implementasse a descoberta de topologia, haveria sobreposição de funções com as plataformas de gerência, o que geraria um desperdício de processamento, possivelmente dificultando a integração da descoberta de QoS nessas plataformas de gerência. A seção 3.1 apresenta uma discussão sobre a relação entre a descoberta de QoS e a descoberta de topologias de rede.

Na seção 3.2 serão analisadas as características da descoberta de QoS que devem ser suportadas pelo modelo proposto, verificando os motivos e decisões considerados na sua elaboração.

As duas seções seguintes apresentam o modelo propriamente dito, que é composto por uma metodologia e uma arquitetura. A metodologia, que define os procedimentos que realizam a descoberta de QoS, é apresentada na seção 3.3. A arquitetura, que define os elementos da descoberta de QoS, é apresentada na seção 3.4.

3.1 Descoberta Topologia X Descoberta de QoS

Como visto na seção 2.3, no QAME a descoberta de QoS irá procurar alvos e informar as características relacionadas a QoS, onde tais informações farão parte de um contexto. Um classificador de tráfego na arquitetura DiffServ, por exemplo, fará parte da visão topológica da rede DiffServ. Como a maioria das arquiteturas de fornecimento de QoS utiliza as informações de roteamento padrão para encaminhar os pacotes, a topologia lógica IP terá um papel fundamental na estrutura de topologias específicas como da rede Diffserv, por exemplo, o que demonstra uma inter-relação, de modo geral, entre as informações de topologia IP e de QoS. Desta maneira, pode-se prever a colaboração entre processos de descoberta de topologia e de QoS, onde um complementaria o outro. As interações entre esses processos podem ser de dois tipos: seqüencial e paralela. Numa abordagem paralela, com caráter mais colaborativo, enquanto a descoberta de topologia IP é executada, seria possível informar a um processo de descoberta de QoS, que fica permanentemente procurando por dispositivos que implementem RSVP, por exemplo, um novo nodo que foi adicionado à rede tão logo seja verificada a sua existência, o que pode diminuir o tempo para as informações de QoS estarem disponíveis. De outro modo, segundo uma abordagem seqüencial, o processo de descoberta de QoS esperaria a descoberta de topologia terminar suas atividades para receber a lista de dispositivos da rede e então começar a executar suas tarefas, o que claramente poderia causar um atraso considerável no tempo de resposta total, considerando o processo de descoberta de topologia e o de QoS. Como não é objetivo desse trabalho implementar um esquema que contemple a descoberta de topologia física (Ethernet) e lógica (IP), será assumido que essas informações estarão disponíveis para o processo de descoberta de QoS. Especificamente, as informações de

topologia física e lógica irão auxiliar nas seguintes tarefas:

- Fornecimento de listas de endereços IPs válidos na rede, ou seja, IPs que já foram previamente verificados, no sentido de evitar que seja investigado um dispositivo que não exista na rede, o que implicaria em desperdício de tempo de processamento e tráfego desnecessário;
- Visualização de topologias de redes que fornecem QoS, pois as interconexões da topologia lógica da camada IP serão a base de informações para apresentar as interconexões, visto que normalmente os dispositivos que fazem parte de arquiteturas que fornecem QoS consideram que o próximo salto que fornece QoS é o *gateway* padrão da camada IP e que a maioria das soluções consideradas nesse trabalho estão baseadas na arquitetura de rede Internet [TAN 96].
- Na distribuição dos processos de descoberta de QoS, pois a topologia física e lógica facilitará na escolha dos pontos ideais da rede para colocar processos de descoberta de QoS, de maneira a minimizar o tempo total de descoberta, bem como o número de segmentos de rede por onde o tráfego desses dados da descoberta passará.

3.2 Características de descoberta de QoS consideradas no modelo

A arquitetura de gerenciamento Internet [STA 98], baseada no protocolo SNMP, possui determinadas características importantes. Nesta seção tais características serão utilizadas como base comparativa para a determinação das características a serem consideradas no modelo proposto. A FIGURA 3.1 apresenta os quatros elementos que compõem a arquitetura de gerenciamento Internet: nodos gerenciados, estações de gerenciamento, informações de gerenciamento e protocolo de gerenciamento. Os nodos gerenciados são os equipamentos de rede (roteadores, *switches*, *hubs*, pontes, *hosts*, etc.) capazes de fornecer informações sobre o seu estado interno às estações de gerenciamento. Um nodo gerenciado possui internamente um agente SNMP que mantém uma base de dados de gerenciamento (MIB – *Management Information Base*) [MCC 91] com objetos que descrevem seu estado. Um agente SNMP permite que o estado interno de um dispositivo seja consultado e/ou modificado através da correta manipulação do conteúdo dos objetos da MIB. O gerenciamento é realizado pelas estações de gerenciamento, que implementam algoritmos específicos de gerência. As estações de gerenciamento são responsáveis pelas tomadas de decisão de gerência, deixando aos agentes SNMP pouco ou nenhuma decisão neste sentido. A comunicação entre as estações de gerenciamento e os agentes SNMP se dá através do protocolo SNMP. A troca de mensagens entre agentes e estações de gerenciamento consome recursos de rede (banda disponível, capacidade processamento dos roteadores, etc.), sendo que o total de recursos consumidos deve ser suficientemente baixo para que a operação normal da rede não seja impactada.

Normalmente, a arquitetura SNMP é utilizada para uma abordagem de gerenciamento centralizado, onde uma única estação de gerenciamento controla diversos dispositivos gerenciados de uma mesma rede. Em redes de dimensões pequenas (em torno de 400 dispositivos), o impacto do tráfego de gerenciamento normalmente não é percebido. Entretanto, em redes de proporções maiores (mais de 1000 dispositivos), o tráfego de gerência pode introduzir alguns problemas. O mais imediato é a sobrecarga

dos segmentos próximos à estação de gerenciamento. Como a estação é o ponto central (de onde partem todas as solicitações e para onde são enviadas todas as respostas de gerenciamento), quanto maior a dimensão das redes, maior será o número de mensagens necessárias, e maior também será o tráfego relacionado à estação de gerenciamento. Outro problema existente é que uma rede maior exige uma capacidade de processamento maior da estação de gerenciamento, de forma que a mesma consiga tomar decisões sobre os vários dispositivos em tempo hábil. Assim, quanto maior for a dimensão da rede gerenciada, maior deverá ser a capacidade de processamento da estação de gerenciamento para que o tempo de resolução de problemas permaneça aceitável.

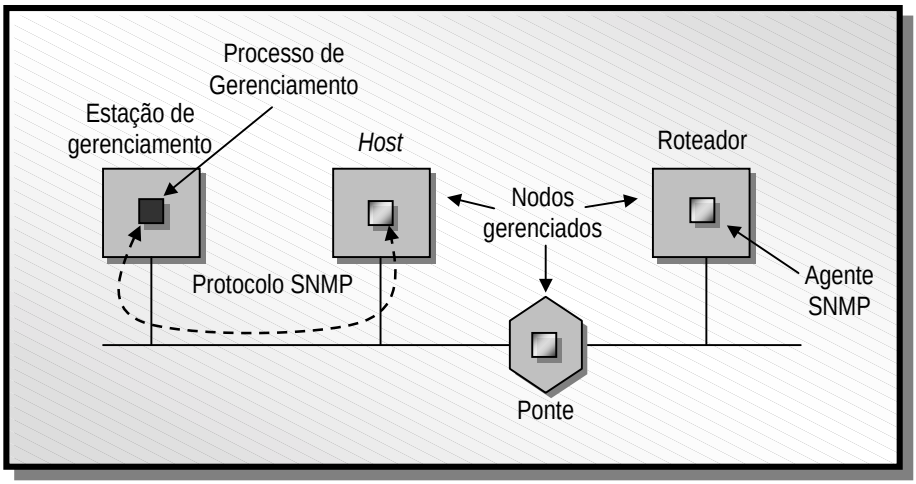

FIGURA 3.1 - Componentes da arquitetura de gerenciamento Internet

Para a resolução destes problemas, uma solução de gerenciamento deve ser escalável (para que o tempo de resolução dos problemas não seja linearmente dependente das dimensões da rede) e distribuída (para que o tráfego de gerenciamento permaneça confinado nos segmentos de interesse). Tais características são também desejadas na descoberta de QoS. A escalabilidade de um esquema de descoberta é importante para que uma rede de dimensões maiores possa ter os mecanismos de fornecimento de QoS descobertos em limites de tempo definidos. A distribuição da descoberta de QoS é necessária para que o tráfego gerado no processo não prejudique a rede gerenciada.

A FIGURA 3.2 apresenta dois cenários de descoberta de QoS: um cenário de descoberta centralizada e um cenário de descoberta distribuída. Os retângulos de R1 a R6 representam os roteadores da rede, enquanto os *hosts* H1, H2 e H3 representam estações de gerenciamento da rede. No exemplo deseja-se descobrir quais os *hosts* de um determinado segmento da rede gerenciada possuem suporte ao protocolo RSVP. Primeiramente, no exemplo de uma abordagem centralizada, a estação de gerenciamento H3 está interessada em descobrir os *hosts* com QoS no segmento S3. Nesta situação, H3 irá consultar cada dispositivo do segmento remoto, consumindo recursos de rede de forma extensiva, já que existirão diversas trocas entre H3 e cada dispositivo de interesse. Além disso, essa troca de informação será “percebida” por outros segmentos que não possuem dispositivos a serem investigados (sub-redes entre R3 e R1, entre R1 e R2 e entre R2 e R6). Esse problema poderia ser minimizado se o processo de descoberta estivesse localizado em uma estação de gerenciamento mais próxima dos elementos de

interesse, desta forma afetando um número menor de segmentos. Assim, no segundo exemplo comparativo, tem-se o uso de uma abordagem distribuída. Neste exemplo, a estação de gerenciamento H1 está interessada em descobrir os *hosts* com QoS do segmento S5. Nesta abordagem distribuída, o gerente H1, de mais alto nível, delega a tarefa de descoberta a um gerente de mais baixo nível. O tráfego entre H2 e H1 é composto basicamente das mensagens necessárias para o envio do processo de descoberta e das respectivas respostas com as informações de QoS coletadas. Neste cenário, o gerente H2 consulta em seu segmento local os dispositivos de interesse, não consumindo recursos de outros segmentos para a realização de consultas aos dispositivos. Além disso, processos de descoberta em vários segmentos diferentes poderiam ser executados em paralelo (diminuindo o tempo de descoberta), desde que em cada segmento de interesse esteja presente um gerente de nível inferior similar ao gerente H2.

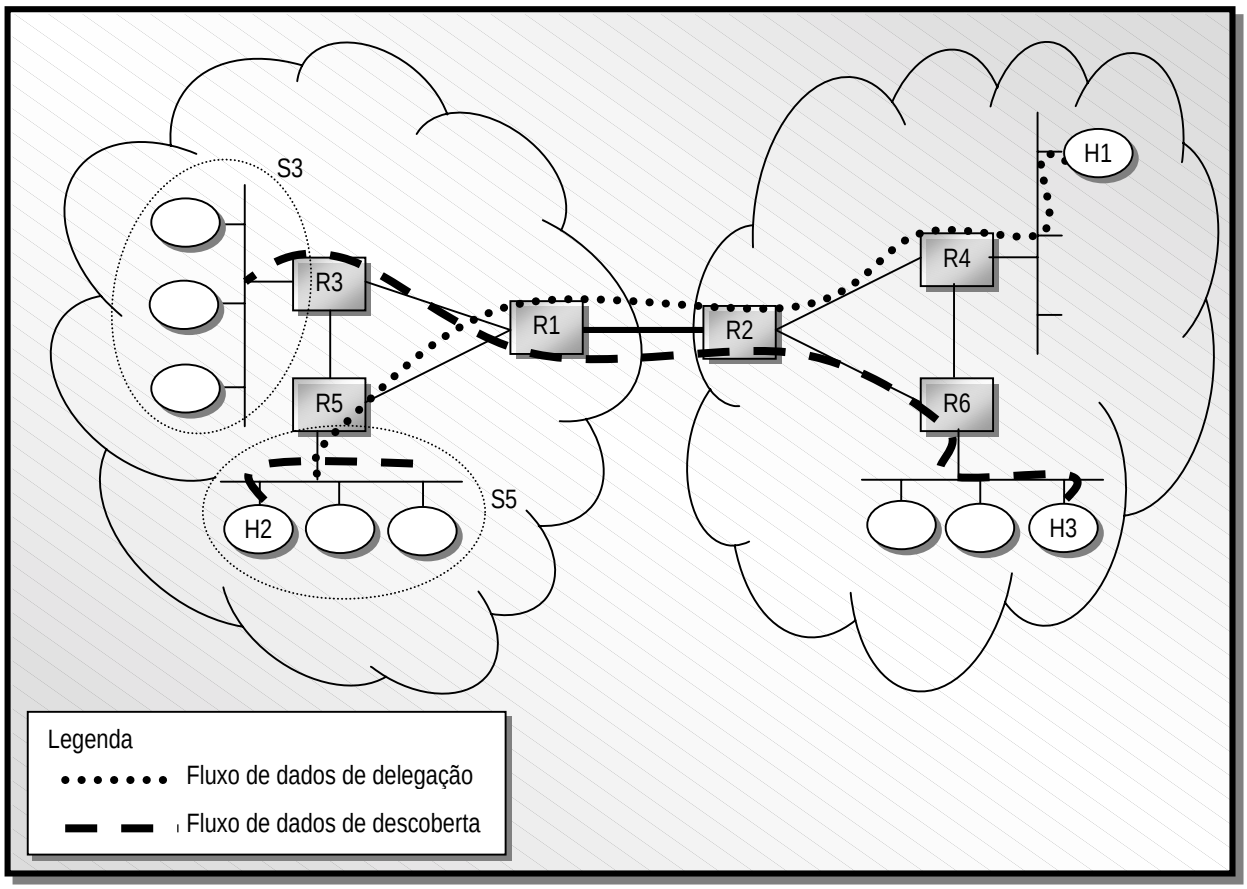

FIGURA 3.2 - Comparação entre processos de descoberta de QoS distribuídos e centralizados

Como mencionado no capítulo 1, o conjunto de mecanismos e arquiteturas de fornecimento de QoS tem aumentado. Nesse conjunto, algumas soluções estão tendo suas normas concluídas [ROS 2001], outras estão evoluindo [BLA 98], além da tendência de novas abordagens serem propostas. Além disso, nem todos os dispositivos que possuem mecanismos para fornecimento de QoS são necessariamente gerenciados através de SNMP. Logo, em relação ao fornecimento de QoS nos dispositivos, duas situações podem ser observadas:

- O conjunto de informações de gerenciamento dos mecanismos de fornecimento

de QoS é dinâmico;

- Os métodos de acesso a tais informações são diversos e não restritos ao SNMP.

Para que uma solução de descoberta de QoS aborde adequadamente estes dois importantes itens, um mecanismo de extensão da solução deve existir. Tal mecanismo é necessário para que o conjunto de informações a serem descobertas possa ser definido de maneira flexível, e para que o acesso aos dispositivos possa ser realizado através de diversos métodos diferentes, não restritos apenas ao SNMP. Logo, além de ser distribuído e escalável, como visto anteriormente, o modelo de descoberta de QoS proposto neste trabalho deve também ser extensível. A TABELA 3.1 resume as características consideradas no modelo de descoberta de QoS em questão.

TABELA 3.1 - Características da descoberta de QoS consideradas no modelo

Característica	**Função**
Modelo distribuído	Para confinar o tráfego de descoberta nos segmentos de interesse.
Modelo escalável	Para permitir uma descoberta em tempo hábil mesmo em redes com proporções maiores
Modelo extensível	Para permitir o acesso a informações de arquiteturas de fornecimento de QoS diversas, utilizando métodos variados.

Nesta seção foram apresentadas as características do modelo que será definido nas próximas seções. O modelo é composto de duas partes: metodologia e arquitetura. A metodologia define os procedimento que realizam a descoberta de QoS, e é apresentada na seção 3.3. A arquitetura define os elementos da descoberta de QoS, onde cada elemento implementa um ou mais procedimentos da metodologia, e é apresentada na seção 3.4.

3.3 Metodologia

A metodologia de descoberta de QoS do modelo proposto é composta por procedimentos. Os primeiros procedimentos são responsáveis por um levantamento de informações mais gerais dos dispositivos de interesse. Os procedimentos seguintes são então responsáveis pelo refinamento de análise, de forma a identificar as características de QoS mais específicas dos dispositivos. A FIGURA 3.3 ilustra a metodologia de descoberta e apresenta os quatro procedimentos existentes.

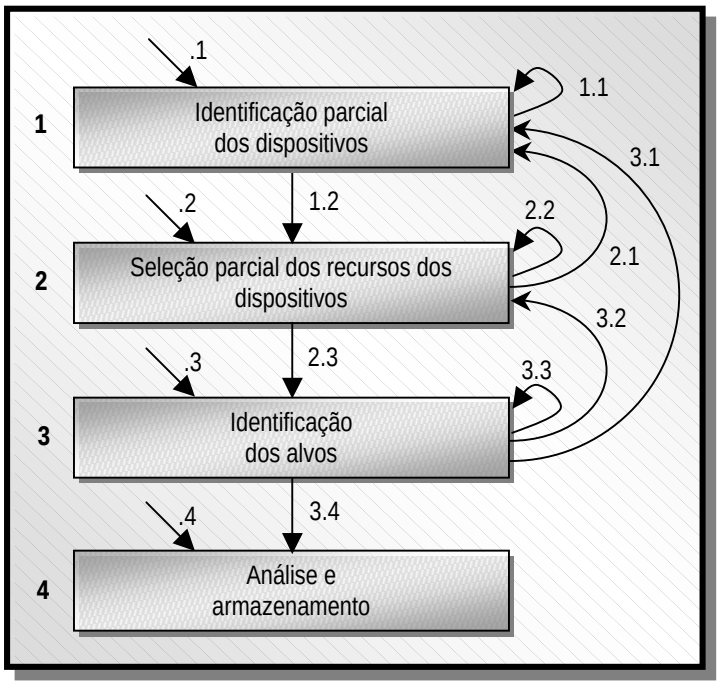

FIGURA 3.3 – Procedimentos da metodologia de identificação dos alvos

Primeiro procedimento: identificação parcial dos dispositivos

O procedimento **identificação parcial dos dispositivos** é responsável por gerar uma lista de identificadores parciais para cada dispositivo. Um identificador parcial é definido como qualquer informação que permita fazer concepções a respeito dos recursos de fornecimento de QoS que o dispositivo implementa, e é obtido tipicamente através de consultas ao dispositivo. Como será visto adiante, os próximos procedimentos dependem bastante do correto fornecimento da lista de identificadores parciais associada aos dispositivos. Alguns exemplos de identificadores parciais são:

- Objeto *system.Description* da MIB-II;
- Cabeçalhos de páginas HTML (*HyperText Markup Language*);
- Respostas a mensagens SYN do TCP;
- Respostas a mensagens ICMP;
- Fabricante, modelo e versão de *firmware* de um dispositivo.

A FIGURA 3.4 apresenta os dados de entrada e saída relativos ao primeiro procedimento da metodologia. Uma lista de dispositivos D é passada ao procedimento para que os identificadores parciais de tais dispositivos possam ser determinados. Tipicamente, os elementos da lista D são endereços de rede dos dispositivos de interesse fornecidos pelo administrador da rede. Ao final do procedimento, uma lista IPD de tuplas *id* é devolvida. Cada tupla *id* é constituída pela identificação de um dispositivo *di*, e de uma lista de identificadores parciais *idp* daquele dispositivo. Assim, o elemento id_1 é uma tupla formada pelo identificador do dispositivo d_1 e pela lista de identificadores parciais idp_1 do dispositivo d_1. Generalizando, o elemento id_n é uma tupla formada pelo

identificador do dispositivo d_n e pela lista dos identificadores parciais idp_n do elemento d_n. Cada dispositivo de interesse pode não possuir um identificador parcial, ou pode mesmo possuir diversos identificadores ao mesmo tempo. Logo, o número de elementos de uma lista idp_n é de [0,X]. Além disso, o número de elementos de idp_n pode ser diferente do número de elementos de idp_{n+1}.

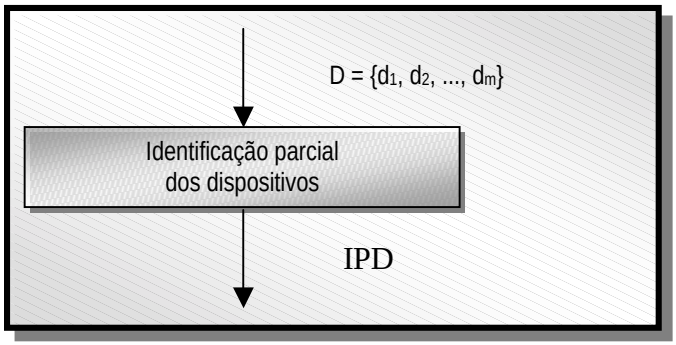

FIGURA 3.4 - Entrada e saída de dados do procedimento
de identificação parcial dos dispositivos

A FIGURA 3.5 mostra uma rede IP exemplo, cujos dispositivos possuem endereços de rede entre 200.132.73.3 a 200.132.73.39. I_A até I_F representam identificadores parciais de alguns dispositivos.

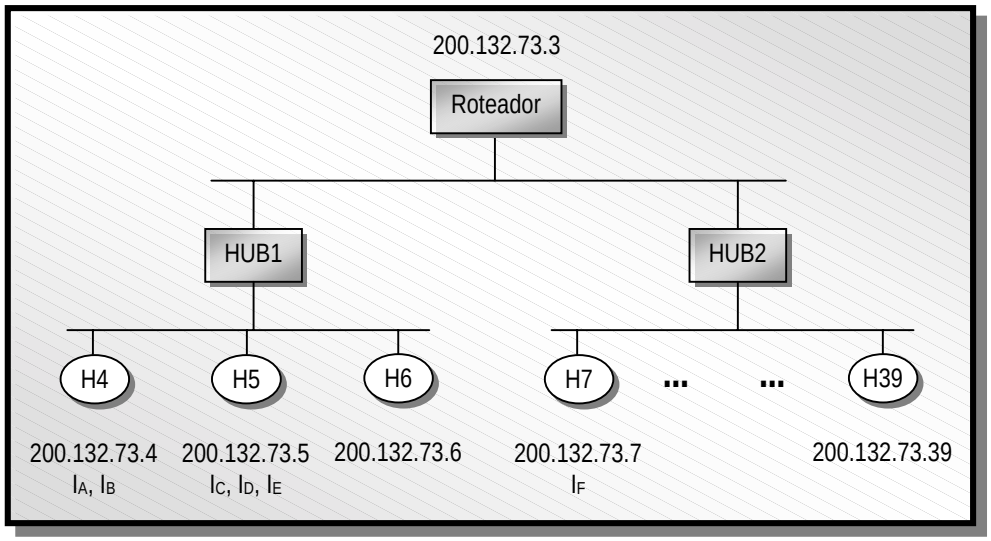

FIGURA 3.5 - Rede IP exemplo para demonstrar uso
dos dados de entrada e de saída do primeiro procedimento

É suposto que apenas alguns dispositivos serão investigados (de 200.132.73.4 a 200.132.73.7). Logo, para esta rede tem-se:

D = {200.132.73.4, 200.132.73.5, 200.132.73.6, 200.132.73.7}

Após a execução desse procedimento, os seguintes resultados serão obtidos:

- IPD = {id_1, id_2, id_3, id_4}
- id_1 = {{200.132.73.4, {I_A, I_B}}
- id_2 = {200.132.73.5, {I_C, I_D. I_E}}

- id_3 = {200.132.73.6, {}}
- id_4 = {200.132.73.7, {I_F}}

A FIGURA 3.6 apresenta o algoritmo base para identificação parcial de alvos em detalhes. A função Consulta do passo 2.b representa todas as consultas que são feitas ao dispositivo para se obter a lista dos identificadores parciais. No passo 2.d.i é testado se o elemento $idptmp_j$ (pertencente à lista temporária de identificadores parciais idptmp) não pertence à lista de identificadores parciais idp_i do dispositivo di_i, para então adicioná-lo a idp_i no passo 2.d.i.1. Esse teste é feito pois diferentes consultas ao dispositivo podem retornar o mesmo identificador parcial.

```
Procedimento ObterIdentificadoresParciais(D)
/* Entrada: D = {d1, d2, ..., dm} : lista de */
/* dispositivos a serem identificados parcialmente. */
/* Saída: IPD = {id1, id2, ..., idm}: lista de */
/* associações (tuplas) entre dispositivos e a lista
*/
/* identificadores parciais. */
/* idi = {dii, idpi} */
/* dii : i-ésimo dispositivo. */
/* idpi = {idpi1, idpi2, ... idpin} : idpi é a lista */
/* de identificadores parciais associados */
/* ao i-ésimo dispositivo. */
/* idptmp = {idptmp1, idptmp2, ..., idptmpj} : */
/* lista temporária de identificadores parciais */
começar
      1.    IPD = ϕ
      2.    Para cada dispositivo di em D faça
            a.    idptmp = ϕ
            b.    idptmp = Consulta(di)
            c.    idpi = ϕ
            d.    Para cada idptmpj em idptmp faça
                  i.Se idptmpj ∉ idpi então
                        1.    idpi = idpi ∪ idptmpj
            e.    IPD = IPD ∪ (dii, idpi)
      3.    Retornar IPD
terminar
```

FIGURA 3.6 - Algoritmo base para identificação parcial dos dispositivos

Segundo procedimento: seleção parcial dos recursos dos dispositivos

O procedimento **seleção parcial dos recursos dos dispositivos** é responsável por enumerar os recursos que o dispositivo implementa. Para tal, esse procedimento receberá a lista IPD de tuplas (dispositivo e lista de identificadores parciais) gerada no procedimento anterior. Para listar os recursos que um dispositivo implementa esse procedimento irá utilizar, além da lista de entrada IPD, uma base de conhecimento. Tal base de conhecimento possui relacionamentos entre identificadores parciais e uma lista de recursos associados que possui [1, N] elementos. A TABELA 3.2 apresenta exemplos de associações entre identificadores parciais e recursos de fornecimento de QoS. Por exemplo, se numa consulta realizada a um dispositivo for retornado o

identificador parcial “Cisco system WS-c6509 Catalyst Operating System Software, Version 5.4(4)”, os recursos associados seriam “SNMP, CISCO-COPS-CLIENT-MIB, CISCO-QoS-PIB-MIB”, e assim por diante. Os exemplos 1 e 2 dessa tabela possuem recursos em comum (SNMP, CISCO-COPS-CLIENT-MIB, CISCO-QoS-PIB-MIB), onde o exemplo 2 possui recursos adicionais (CISCO-CATOS-ACL-MIB) por ser um identificador parcial que representa uma atualização em relação ao identificador parcial do exemplo 1. Já os exemplos 3 e 4 não possuem recursos em comum.

TABELA 3.2 - Exemplo de identificadores parciais e recursos de fornecimento de QoS

Exemplo	Identificador Parcial	Recursos associados ao fornecimento de QoS
1	Cisco Systems WS-C6509 Cisco Catalyst Operating System Software, Version 5.4(4)	SNMP, CISCO-COPS-CLIENT-MIB, CISCO-QOS-PIB-MIB
2	Cisco Systems WS-C6509 Cisco Catalyst Operating System Software, Version 7.1	SNMP, CISCO-CATOS-ACL-QOS-MIB, CISCO-COPS-CLIENT-MIB, CISCO-QOS-PIB-MIB, CISCO-QOS-POLICY-CONFIG-MIB
3	IntServ	INTEGRATED-SERVICES-MIB
4	AF PHB	DIFFSERV-MIB

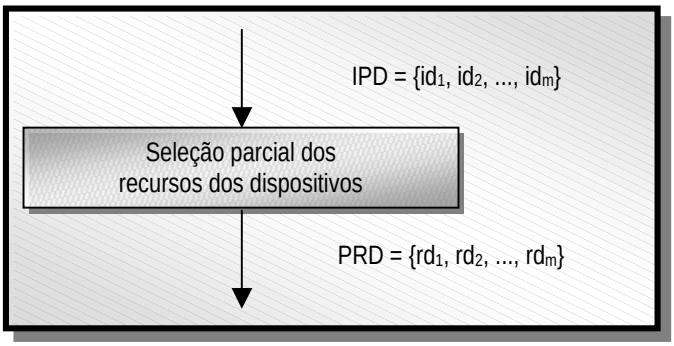

FIGURA 3.7 - Entrada e saída de dados do procedimento de seleção parcial dos recursos dos dispositivos

A FIGURA 3.7 apresenta os dados de entrada e saída relativos ao segundo procedimento da metodologia. Uma lista IPD de tuplas, onde cada tupla é formada por um identificador de dispositivo e uma lista de identificadores parciais desse dispositivo, é passada ao procedimento para que os recursos de tais dispositivos possam ser determinados, tendo como base os identificadores parciais. Ao final do procedimento uma lista PRD de tuplas *rd* é fornecida. Cada tupla *rd* é constituída pela identificação de um dispositivo *di,* e de uma lista *ri* de recursos de fornecimento de QoS daquele dispositivo. Assim, o elemento rd_1 é uma tupla formada pelo identificador do dispositivo di_1 e pela lista de recursos de fornecimento de QoS ri_1 do dispositivo di_1. Generalizando, o elemento rd_n é uma tupla formada pelo identificador do dispositivo di_n e pela lista dos recursos de fornecimento de QoS ri_n do elemento di_n. Cada dispositivo

de interesse pode não possuir um recurso de QoS, ou pode mesmo possuir diversos recursos de QoS ao mesmo tempo. Logo, o número de elementos de uma lista ri_n é de [0,Y]. Além disso, o número de elementos de ri_n pode ser diferente do número de elementos de ri_{n+1}.

A FIGURA 3.8 apresenta o algoritmo base para a seleção parcial dos recursos de fornecimento de QoS dos dispositivos.

```
Procedimento ObterRecursosParciaisDispositivo(IPD)
/* Entrada: IPD = lista de associações (tuplas) */
/* entre dispositivos e identificadores parciais*/
/* Saída: PRD = {rd1, rd2, ..., rdm} : lista de */
/* associações (tuplas) entre dispositivo e lista de */
/* recursos de fornecimento de QoS que o */
/* dispositivo implementa */
/* rdi = (dii, rii) */
/* dii : i-ésimo dispositivo */
/* rii = {ri1, ri2, ..., rim} : lista de */
/* recursos de fornecimento de QoS associada ao */
/* i-ésimo dispositivo */
/* rtmp : lista temporária de recursos */
/* de fornecimento de QoS */
começar
      2.    PRD = ϕ
      3.    Para cada idi em IPD faça
            a.    rtmp = ϕ
            b.    Para cada idpij em idpi faça
                i.rtmp = ListarRecursos(idpij)
               ii.Para cada rtmpk em rtmp faça
                        1.    Se rtmpk ∉ rii então
                              a.    rii = rii ∪ rtmpk
              iii.rtmp = ϕ
               iv.PRD = PRD ∪ (dii, rii)
      4.                    Retornar PRD
terminar
```

FIGURA 3.8 – Algoritmo base para seleção parcial
dos recursos dos dispositivos

O algoritmo é bastante simples e os principais passos são:

1. Para cada identificador parcial, a função ListarRecursos (passo 2.b.i) consulta a base de conhecimentos para gerar uma lista de saída PRD.
2. No passo 2.b.ii.1 é testado se elemento $rtmp_j$ (da lista temporária de identificadores parciais *rtmp*) pertence à lista de recursos de fornecimento de QoS ri_i do dispositivo di_i.
3. No passo 2.b.ii.1.a. o elemento $rtmp_j$ é adicionado a ri_i se já não estiver presente. Esse teste é necessário, pois diferentes identificadores parciais podem possuir associações com o mesmo recurso de fornecimento de QoS.

Terceiro procedimento: identificação dos alvos

O procedimento **identificação dos alvos** é responsável por fazer consultas aos dispositivos para obter uma lista de recursos disponíveis no mesmo, que será utilizada para identificar os alvos propriamente ditos. Os alvos não são identificados diretamente neste procedimento porque nem todos os recursos de fornecimento de QoS são alvos em dispositivos. Por exemplo, um dispositivo pode possuir recursos que auxiliem na implementação de DiffServ, mas não atuem diretamente no fornecimento de QoS: um medidor de tráfego baseado em Token Bucket não é um alvo, mas o elemento, tal como um classificador, que realiza uma ação baseada no resultado do medidor é um alvo. Assim, a classificação dos recursos em alvos será realizada no próximo procedimento.

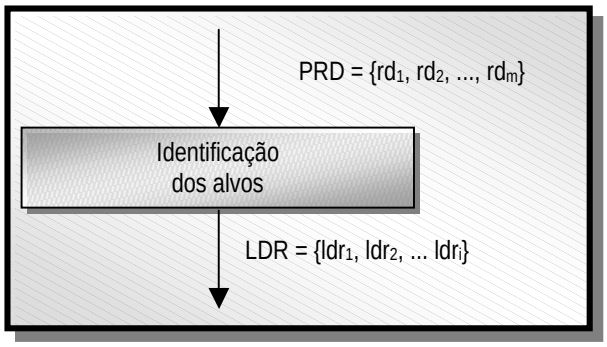

FIGURA 3.9 – Entrada e saída de dados do procedimento de identificação dos alvos

A FIGURA 3.9 apresenta os dados de entrada e saída relativos ao terceiro procedimento da metodologia. Uma lista PRD de tuplas, onde cada tupla é formada pelo identificador de dispositivo e uma lista de recursos desse dispositivo a serem identificadas, é passada ao procedimento. Ao final do procedimento, uma lista LDR de tuplas *ldr* é fornecida. Cada tupla *ldr* é constituída pela identificação de um dispositivo *d*, e de uma lista *lr* de recursos de fornecimento de QoS identificadas no dispositivo. Assim, o elemento ldr_1 é uma tupla formada pelo identificador do dispositivo d_1 e pela lista de recursos de fornecimento de QoS identificados lr_1 do dispositivo d_1. Generalizando, o elemento ldr_n é uma tupla formada pelo identificador do dispositivo d_n e pela lista dos recursos de fornecimento de QoS identificados lr_n do elemento d_n. Cada dispositivo de interesse pode não ter todos os recursos solicitados em *ri* de *rd* em PRD (lista de entrada) identificados, pois tais recursos podem não estar implementados no dispositivo. Esta situação é diferente do problema de não identificar alvos, mesmo que o recurso esteja implementado. Neste caso, o recurso não é identificado porque não está implementado (presente) no dispositivo, enquanto no caso do alvo não identificado, o recurso está presente no dispositivo, mas não disponível para ser utilizado. Por fim, podem ter sido identificados diversos recursos de QoS para um único dispositivo. Logo, o número de elementos de uma lista ldr_n é de [0,Z]. Além disso, o número de elementos de ldr_n pode ser diferente do número de elementos de ldr_{n+1}.

O algoritmo desse procedimento é definido na FIGURA 3.10. Como já mencionado, ele recebe como entrada de dados uma lista (PRD) de tuplas (rd_i) entre dispositivo (di_i) e lista (ri_i) de recursos de QoS que se deseja identificar no dispositivo di_i. Para cada possível recurso (ri_i) a ser identificado, o dispositivo é consultado (passo 2.a.i.2.a). Nessa consulta pode se obter configurações e instâncias do recurso. Se o dispositivo não possui recurso a ser identificado (a lista é vazia), uma consulta genérica é realizada (passo 2.a.ii). Uma consulta genérica irá obter todos os recursos presentes no dispositivo. A lista vazia ri_i pode ocorrer em diversas situações. Uma delas é não ter sido encontrado nenhum possível recurso para o dispositivo di_i, no segundo procedimento. Outra possibilidade é a intenção de identificar todos os recursos de fornecimento de QoS e alvos que um dispositivo possui.

```
Procedimento IdentificarAlvos(PRD)
/* Entrada: PRD = lista de associações (tuplas) entre */
/* dispositivo e lista de recursos de */
/* fornecimento de QoS a serem identificada */
/* Saída: LDR = {ldr1, ldr2, ... ldri}: lista de */
/* associações (tuplas) entre dispositivos e */
/* lista de recursos de QoS identificada */
/* ldri = (di, lri) */
/* di : i-ésimo dispotivo */
/* lri : lista de recursos identificados no i-ésimo dispositivo */
/* ltmp : lista temporária de recursos de QoS */
começar
        1. LDR = ϕ
        2. Para cada rdi em PRD faça
            a.    Para cada dii em rdi faça
                i.Se ri ≠ ϕ então
                        1.    ltmp = ϕ
                        2.    Para cada rij em rii faça
                              a.    ltmp = lmtp ∪
                              Consultadispositivo(dii, mij)
               ii.Senão ltmp = ConsultaGenericaDispositivo(dii)
              iii.lri = ltmp
               iv.LDR = LDR ∪ (di, lri)
        3. Retornar LDR
terminar
```

FIGURA 3.10 - Algoritmo base para identificação dos alvos

Quarto procedimento: análise e armazenamento

O procedimento **análise e armazenamento** é responsável por analisar e armazenar as informações identificadas nesse terceiro procedimento, disponibilizando as informações para as outras tarefas de gerenciamento de QoS. A análise irá realizar verificações de consistência e validação das informações coletadas, enquanto o armazenamento irá formatar os dados e armazená-los segundo o modelo de dados da base de informações de gerenciamento. Como dados de entrada esse procedimento recebe pelo menos as listas IPD (gerada no primeiro procedimento), PRD (gerada no segundo procedimento) e LDR (gerado no terceiro procedimento) que podem ser analisadas juntamente com as informações da base de dados de gerenciamento. Outras

informações podem ser utilizadas, dependendo das tarefas realizadas nesse procedimento, como será visto adiante.

Diversas análises podem ser feitas, dependendo dos objetivos da descoberta de QoS e do propósito do ambiente de gerenciamento e da rede. Por exemplo, um ambiente de gerenciamento pode ser responsável por detectar e corrigir qualquer falha no processo de gerenciamento de rede. Uma tarefa desse ambiente seria analisar o resultado das consultas de identificação de alvos para verificar aquelas que falharam, onde uma possível solução seria verificar se o que foi solicitado a ser identificado, informado na lista PRD, foi realizado e armazenado na lista LDR. Outro ambiente de gerenciamento pode basear-se num princípio onde diferentes tarefas de gerenciamento devem utilizar diferentes bases de informações de gerenciamento. Cada base teria um modelo de dados adequado para acelerar as suas consultas às informações de gerenciamento de QoS. Assim, esse procedimento armazenaria as informações identificadas em diferentes bases de dados, atendendo às necessidades das várias plataformas de gerenciamento. Outras análises e armazenamentos poderiam ser realizados com outros objetivos, se considerarmos entradas de dados não apresentadas aqui.

TABELA 3.3 - Exemplos de tarefas de análise e armazenamento realizadas no quarto procedimento da metodologia.

Informações envolvidas	**Tarefa**
LDR, base de informações de gerenciamento	Análise - Associação dos alvos aos dispositivos. Classificação dos alvos de acordo com sua funcionalidade.
IPD, base de conhecimento, PRD, LDR	Análise – relatar quais os identificadores parciais de IPD não possuíam um recurso de QoS associado para a causa ser determinada. Para corrigir esse problema, ou as técnicas de obtenção de identificadores parciais, ou a base de conhecimento deveriam ser atualizadas.
IPD, base de conhecimento, PRD, LDR	Análise – relatar quais os recursos de QoS de PRD não foram identificados no dispositivo para a causa ser determinada: base de conhecimento desatualizada ou o alvo mal configurado.
PRD e LDR	Análise – relatar se o que foi solicitado a ser consultado em PRD foi obtido em LDR. As informações em LDR não estariam erradas, mas a associação entre o recurso solicitado e o identificado sim. A causa da inconsistência poderia ser: base de conhecimento com associação entre identificador parcial e recurso de QoS errada, consulta de identificação inadequada, dispositivo mal configurado, entre outros.
LDR, base de informações de gerenciamento	Análise – relatar alterações na topologia das várias arquiteturas de fornecimento de QoS presentes na rede. Para tal, os alvos identificados em LDR seriam confrontados com a base de informações de gerenciamento.
LDR, base de informações de gerenciamento	Armazenamento – disponibilizar as informações identificadas em LDR para várias bases de gerenciamento, cada uma com tarefas específicas: gerenciamento de falhas e desempenho.

Como pode ser visto, tanto os dados de saída como os de entrada desse

procedimento podem variar bastante, atendendo necessidades específicas de tarefas de gerenciamento de QoS de cada rede. Por isso, esse procedimento não irá apresentar algoritmo base. Entretanto, para facilitar a compreensão, a TABELA 3.3 lista algumas possíveis tarefas de análise e armazenamento.

Tendo todos os procedimentos da metodologia definidos, as transições e iterações representadas pelas setas na FIGURA 3.3 serão apresentadas. As setas 1.2, 2.3 e 3.4 representam as transições entre cada dois procedimentos. Quando o primeiro procedimento termina, ocorre a transição 1.2, onde os dados gerados no primeiro procedimento são os dados de entrada do segundo procedimento. O mesmo tipo de transição ocorre na passagem do segundo para o terceiro procedimento, representada pela seta 2.3. A seta 3.4 representa a transição do terceiro para o quarto procedimento, onde o quarto procedimento recebe como dados de entrada pelo menos os dados gerados pelo primeiro, segundo e terceiro procedimento (IPD, PRD e LDR respectivamente), além da base de informações de gerenciamento.

As iterações são representadas pelas setas 3.3, 3.2, 3.1, 2.2, 2.1 e 1.1, e permitem o refinamento do resultado da descoberta de QoS. Por exemplo, se o resultado do segundo procedimento fornece um número considerado insuficiente de recursos que um dispositivo implemente, é possível voltar ao primeiro procedimento (seta 2.1) para obter novos identificadores parciais, diferentes dos obtidos na iteração anterior. Assim, a execução do segundo procedimento seria reiniciada com novos dados de entrada, onde possivelmente seriam obtidos novos recursos de fornecimento de QoS para o dispositivo. Outra solução para esse caso seria consultar outra base de conhecimento numa nova iteração, representada pela seta 2.2. Se a iteração da seta 2.1 ou 2.2 não fosse utilizada e o processo de identificação continuasse, seria possível que nenhum recurso de fornecimento de QoS fosse identificado. Uma possível causa, o número insuficiente de recursos de fornecimento de QoS gerados pelo segundo procedimento, seria detectada apenas no último procedimento, após a execução de todos os procedimentos da metodologia, onde se faz análise das informações identificadas. Assim, seria necessário que todo o processo fosse novamente realizado para que os recursos do dispositivo fossem identificados. Analogamente, os dados gerados no primeiro procedimento poderiam ser refinados através da iteração representada pela seta 1.1, onde apenas novas consultas poderiam gerar resultados diferentes. Por fim, os dados do terceiro procedimento poderiam ser refinados através das iterações representadas pelas setas 3.3, 3.2 e 3.1. Assim, as iterações (setas das FIGURA 3.3) representam análise das informações intermediárias (resultado de cada procedimento) que permitem o refinamento dessas informações, e, conseqüentemente, é feita a detecção precoce de problemas relacionados a resultados inadequados de cada procedimento.

Por fim, as setas .1, .2, .3, .4 representam a possibilidade dos procedimentos da metodologia apresentada receberem dados de entrada não gerados pelos procedimentos dessa metodologia. Assim, é possível que outras metodologias e respectivos procedimentos forneçam tais informações. A única ressalva é que tais entradas de dados alternativas devem seguir o formato dos dados de entrada apresentados em cada procedimento. Por exemplo, se o gerente de rede já sabe quais recursos deseja identificar em determinado dispositivo, ele não precisa passar pelo primeiro e segundo procedimentos.

3.4 Arquitetura

Tendo em vista as características consideradas no modelo de descoberta de QoS e a metodologia apresentada na seção anterior (onde todos os procedimentos para a realização da descoberta foram definidos), nesta seção é proposta uma arquitetura para descoberta de QoS. Tal arquitetura é apresentada na FIGURA 3.11.

A arquitetura proposta é baseada nas pesquisas desenvolvidas pelo IETF, através do grupo de trabalho DISMAN (*Distributed Management*) [DIS 2001], onde é definido um modelo para gerenciamento de redes distribuído [LEV 2001]. O modelo do IETF preocupa-se em fornecer mecanismos para implementar gerenciamento distribuído em redes, não restringindo o escopo de sua solução para uma tarefa de gerenciamento específica. Assim, este trabalho utiliza o suporte fornecido pelos trabalhos do IETF como base para propor uma arquitetura de descoberta de QoS. A arquitetura aqui proposta implementa todos os procedimentos previamente definidos na metodologia de descoberta de QoS bem como prevê a integração com o ambiente de gerenciamento de QoS QAME.

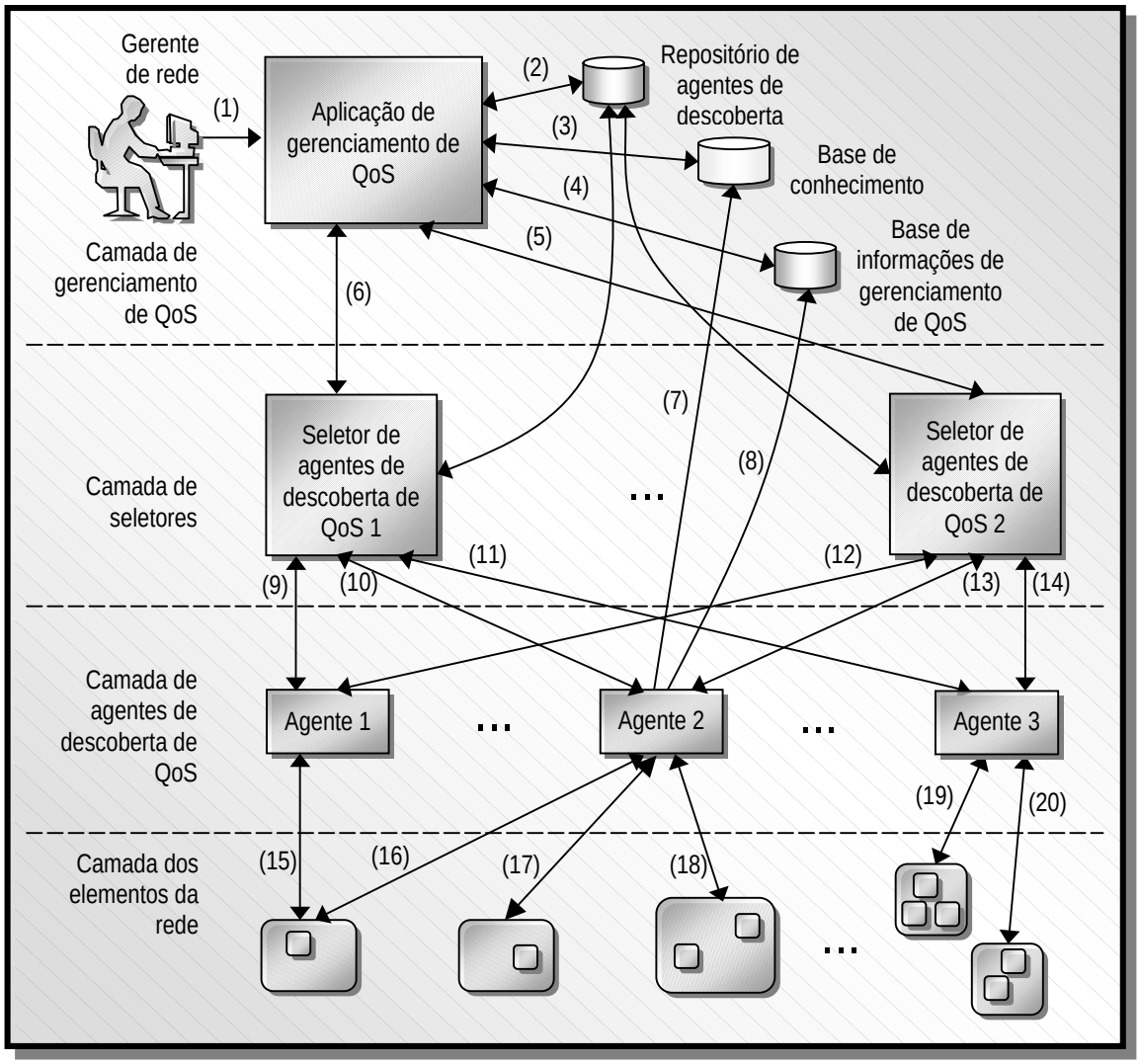

FIGURA 3.11 - Arquitetura de descoberta de QoS

A arquitetura apresenta quatro camadas de elementos funcionais (da camada

inferior para a camada superior): camada dos elementos da rede, camada de agentes de descoberta de QoS, camada de seletores e camada de gerenciamento de QoS. Todos os elementos estão localizados na rede gerenciada e interagem entre si para implementar a descoberta de QoS. A camada dos elementos da rede é constituída de dispositivos e alvos associados. A camada de agentes de descoberta de QoS é composta de agentes de descoberta de QoS. A camada de seletores é constituída de seletores de descoberta de QoS. Por fim, a camada de gerenciamento de QoS é composta pela aplicação de gerenciamento, pelo repositório de agentes de descoberta, pela base de conhecimento e pela base de informações de gerenciamento. As comunicações entre os elementos da arquitetura são representadas na FIGURA 3.11 por setas numeradas, descritas oportunamente. Todos os elementos da arquitetura são agora detalhadamente abordados. Logo em seguida serão apresentados três exemplos da utilização da arquitetura, juntamente com a metodologia proposta neste trabalho.

Alvos

Os **alvos** são elementos ativos que tornam possível o fornecimento de QoS pelas redes de computadores, sendo que cada **dispositivo** pode possuir diversos alvos. Cada alvo fornece um ou mais recursos de fornecimento de QoS. Assim, os alvos são os elementos que a descoberta de QoS pretende identificar. A descoberta de QoS irá classificar o alvo de acordo com a operação que ele executa e irá associá-lo ao respectivo dispositivo de rede.

Agentes de descoberta de QoS

Os elementos da camada de **agentes de descoberta de QoS** (chamados, a partir de agora, simplesmente de agentes de descoberta) são responsáveis por implementar os procedimentos definidos pela metodologia de descoberta de QoS. Tais agentes recebem as solicitações de execução (setas 9-14) de descobertas de QoS de um seletor de agentes. Não existe qualquer comunicação direta entre os agentes de descoberta, como poderia ocorrer num ambiente colaborativo, e por isso não há setas que ligam quaisquer dois agentes na FIGURA 3.11.

TABELA 3.4 - Tarefas dos agentes de descoberta

Agente de descoberta de QoS	Procedimentos
1	Identificação parcial
2	Identificação parcial, seleção parcial dos recursos de fornecimento de QoS, identificação dos alvos e análise e armazenamento das informações obtidas
3	Identificação parcial e identificação dos alvos

Os agentes de descoberta estão livres para implementar quaisquer procedimentos da metodologia. A TABELA 3.4 exemplifica três tipos de agentes de descoberta. O

agente de descoberta 1 implementa apenas um procedimento da metodologia, enquanto os outros agentes (2 e 3) implementam dois ou mais procedimentos. Se um agente de descoberta implementa mais de um procedimento, não é obrigatório que eles sejam consecutivos. Por exemplo, um agente pode implementar tanto o primeiro e o terceiro procedimento da metodologia (como o agente 3), como o segundo e terceiro. Um agente pode ser solicitado a executar qualquer procedimento da metodologia de descoberta, mesmo que internamente ele não possua suporte ao procedimento de metodologia solicitado. Assim, o agente de descoberta poderia receber a solicitação de execução do primeiro e segundo procedimento, ou o segundo, terceiro e quarto procedimento, bem como a solicitação de todos os procedimentos da metodologia de descoberta de QoS. Embora tenham sido apresentados na TABELA 3.4 três agentes de descoberta, outras combinações de procedimentos poderiam ser utilizadas para compor outros agentes. Esta flexibilidade visa atender o requisito de escalabilidade do modelo e será melhor analisada posteriormente.

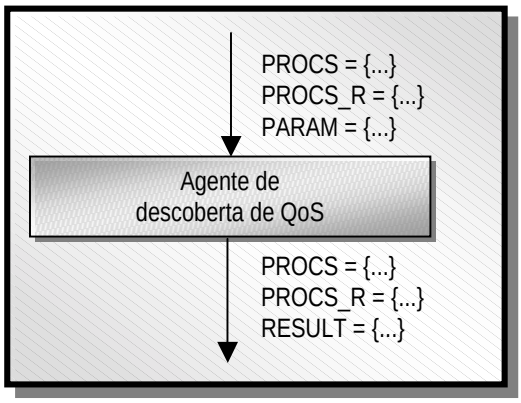

FIGURA 3.12 - Dados de entrada e saída de
um agente de descoberta de QoS

A FIGURA 3.12 ilustra os dados de entrada e de saída de um agente de descoberta. Um seletor de agentes de descoberta solicita a execução dos procedimentos a um agente de descoberta, fornecendo três listas como dados de entrada: PROCS, PROCS_R e PARAM. Em PROCS, o seletor informa a lista de todos os procedimentos que inicialmente foram solicitados a serem executados. Por exemplo, a lista PROCS com valores {1, 2, 3, 4} informa que todos os procedimentos foram solicitados. Em PROCS_R são informados os procedimentos que já foram executados em algum momento. Uma lista PROCS_R {1,2} informa que já foram executados os dois primeiros procedimentos da metodologia de descoberta de QoS. Por fim, em PARAM são fornecidos os dados de entrada dos procedimentos que serão executados. Sabendo quais os procedimentos que devem ser executados e quais já foram executados, o agente de descoberta irá executar aqueles procedimentos que possuírem dados de entrada. Como neste exemplo falta executar o terceiro e quarto procedimentos da metodologia, tipicamente seriam fornecidos os dados de entrada para a execução do terceiro procedimento, ou seja, a lista PRD. Se fossem fornecidos os dados de entrada para a execução do quarto procedimento, o agente não iria executar, por este não ser o procedimento consecutivo em relação ao último procedimento executado.

Após a execução dos procedimentos da metodologia, o agente de descoberta gera três listas como dados de saída: PROCS, PROCS_R e RESULT. PROCS permanece com os mesmos valores inalterados pelo agente de descoberta durante a execução da

descoberta de QoS e, portanto, ele é igual a PROCS dos dados de entrada. A diferença entre PROCS_R dos dados de saída e dos dados de entrada, é que na PROCS_R dos dados de saída foram adicionados os procedimentos executados pelo agente de descoberta. Se no exemplo apresentado no parágrafo anterior o agente executasse apenas o terceiro procedimento da metodologia, PROCS_R dos dados de saída teria o valor {1, 2, 3}. O resultado propriamente dito da execução dos procedimentos da metodologia de descoberta de QoS são armazenados em RESULT, que nesta caso é {LDR} (o terceiro procedimento da metodologia foi executado). É importante salientar que PARAM e RESULT, dados de entrada e saída, respectivamente, dos procedimentos executados pelo agente de descoberta, devem seguir a definição dos dados de entrada e de saída dos procedimentos da metodologia de descoberta de QoS. Por exemplo, se PROCS dos dados de entrada for {1, 2} (o gerente solicitou a execução do primeiro e segundo procedimento da metodologia), PROCS_R for {1} (o primeiro procedimento já foi realizado) e o agente é capaz de executar todos os procedimentos da metodologia da descoberta de QoS, PARAM deve ser {IPD} e RESULT {PRD}.

O resultado da execução de um agente pode ser enviado para dois elementos da arquitetura: de volta para o seletor de agentes de descoberta, ou as informações da descoberta de QoS são armazenadas diretamente na base de informações de gerenciamento e o seletor de agentes de descoberta é informado sobre o fim da execução da tarefa. Embora o agente de descoberta possa enviar os resultados diretamente para a base de informações de gerenciamento, o seletor de agentes sempre deve ser informado do estado final da execução, para que o devido registro de execução seja realizado. A escolha do envio dos resultados, ou para o seletor de agentes de descoberta de QoS ou para a base de informações de gerenciamento, depende das tarefas de cada agente, como será visto adiante. O agente 3, por exemplo, envia (setas 11 ou 14) os dados de saída, na qual o campo RESULT possui a lista LDR de associações entre dispositivos e recursos (caso tenha sido solicitado a execução do terceiro procedimento), para o seletor de agentes de descoberta que solicitou a tarefa: o seletor de agentes de descoberta de QoS 1 ou o 2. Já o agente 2, que implementa todos os procedimentos da metodologia de descoberta de QoS, pode enviar os resultados dos procedimentos (campo RESULT) (seta 8) diretamente para a base de informações de gerenciamento, caso tenha sido executado o quarto procedimento. As informações de estado final de execução (PROCS e PROCS_R) são enviadas (setas 10 ou 13) para o seletor de agentes de descoberta de QoS que solicitou (1 ou 2). Um detalhe importante a respeito do agente 2 é o acesso que ele faz à base de conhecimento (seta 7), pois implementa o procedimento de seleção parcial de recursos dos dispositivos. É importante ressaltar que, embora os elementos da camada de agentes de descoberta de QoS sejam chamados de agentes, não há relação desses com agentes móveis nem com os agentes SNMP e, evidentemente, a implementação pode utilizar um desses conceitos para implementar os agentes de descoberta.

Seletor de agentes de descoberta de QoS

Na camada de seletores, um **seletor de agentes de descoberta de QoS** (chamado, a partir de agora, simplesmente de seletor de agentes) é responsável por escolher e ativar um agente de descoberta que irá executar uma descoberta de QoS. Os seletores de agentes recebem as solicitações de descoberta de QoS (setas 5 e 6) da aplicação de gerenciamento de QoS, e utilizarão parte das informações contidas na

solicitação como parâmetros do critério de decisão da escolha dos agentes que irão executar a tarefa solicitada. Após a escolha, o seletor de agentes envia a solicitação para o agente de descoberta. Não existe comunicação entre seletores de agentes e, por isso, não há setas que ligam quaisquer dois seletores de agentes na FIGURA 3.11. É importante lembrar que todas as solicitações para realização de descoberta de QoS passam pelo seletor de agentes.

TABELA 3.5 - Critérios dos seletores de agentes

Seletor de agentes de descoberta de QoS	Critérios
1	Proximidade entre agente de descoberta e dispositivos a serem identificados
2	Poder de processamento do agente de descoberta. Proximidade entre agente de descoberta e dispositivos a serem identificados. Intensidade do tráfego gerado para realizar a descoberta.

Analogamente à camada de agentes de descoberta de QoS, nessa camada os seletores de agentes estão livres para implementar qualquer conjunto de critérios de escolha dos agentes de descoberta. A TABELA 3.5 exemplifica 2 seletores de agentes, os mesmos ilustrados na FIGURA 3.11, e os respectivos critérios que cada exemplo implementa. Os critérios que cada seletor de agentes implementa podem ser parametrizáveis ou não. Um seletor de agentes que não aceita parâmetros é útil para selecionar um agente de descoberta específico, por exemplo. Na prática, um seletor de agentes desse tipo não executa qualquer escolha de agente de descoberta, apenas repassa a solicitação. Essa possibilidade deixa um mecanismo para o gerente de rede escolher um agente de descoberta específico para executar a descoberta de QoS. Por exemplo, se um gerente de rede deseja que determinado agente de descoberta execute a descoberta de QoS, ele criará um seletor de agentes que repasse as solicitações de descoberta de QoS sempre para o referido agente de descoberta. Por outro lado, os seletores de agentes que aceitam parâmetros permitem uma escolha mais flexível, que possa se adequar ao cenário de descoberta e a critérios do usuário, tal como tempo máximo para executar a descoberta de QoS.

O seletor de agentes 1, por exemplo, implementa um critério que avalia a proximidade entre agente de descoberta e dispositivos a serem consultados, que pode ser medida pela quantidade de saltos entre o agente de descoberta e os dispositivos. Assim, se um gerente de rede deseja que a rede como um todo perceba a menor quantidade de tráfego, que pode ser alcançada pela proximidade física entre um agente de descoberta e os dispositivos a serem identificados, irá solicitar a descoberta de QoS ao seletor de agentes 1. Já o seletor de agentes 2 implementa três critérios de seleção de agentes, onde um deles é idêntico ao implementado pelo seletor de agentes 1. Outro critério que o seletor de agentes 2 implementa é o poder de processamento. Um gerente de rede pode explorar esse critério para diminuir o tempo de execução da descoberta de QoS, solicitando que a tarefa seja executada por agentes de descoberta com mais poder de processamento, caso o processamento nos agentes de descoberta seja um fator considerado importante no tempo de execução da tarefa. O último critério implementado

pelo seletor de agentes 2 é baseado na quantidade de tráfego gerado pelo agente de descoberta para executar a descoberta de QoS. Eventualmente um agente de descoberta pode gerar mais tráfego de rede para realizar as consultas por se basear num princípio de exaustão de possibilidades. Se um gerente precisa executar uma descoberta de QoS em um período de tráfego intenso e não quer interferir no tráfego da rede com fortes rajadas causadas pelas consultas da tarefa, irá solicitar ao seletor de agentes 2, que é capaz de escolher um agente de descoberta com esse perfil. Embora tenham sido apresentados três seletores de agentes de descoberta de QoS, outros critérios podem ser utilizados para formar novos seletores. Esta característica visa a atender o requisito de flexibilidade do modelo e será melhor analisada posteriormente.

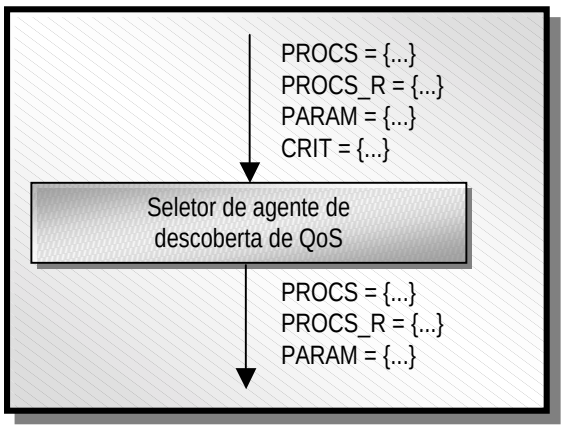

FIGURA 3.14 - Dados de entrada e saída de um seletor de agentes de descoberta de QoS

A FIGURA 3.14 ilustra os dados de entrada e saída de um seletor de agentes. A aplicação de gerenciamento solicita a execução de procedimentos ao seletor de agentes, fornecendo quatro listas como dados de entrada: PROCS, PROCS_R, PARAM e CRIT. Analogamente aos dados de entrada do agente de descoberta, PROCS e PROCS_R possuem a lista de procedimentos a serem executados e a lista de procedimentos já executados, respectivamente. Também análogo ao agente de descoberta, PARAM possui a lista de parâmetros dos procedimentos de descoberta de QoS propriamente ditos. A lista CRIT apresenta os parâmetros dos critérios que o seletor de agentes irá avaliar. Por exemplo, o seletor de agentes 1 poderia receber a seguinte lista CRIT: {PROXIMIDADE='< 2 saltos'}. Já o seletor de agentes 2 poderia receber a seguinte lista CRIT: {PODER_PROC='alto', PROXIMIDADE='< 4 saltos', TRÁFEGO='< 25 kbytes/s'}. O poder de processamento solicitado é 'alto', baseado numa classificação pré-estabelecida. Outro parâmetro é a proximidade entre agente de descoberta e dispositivos, onde é solicitado que o agente de descoberta esteja distante dos dispositivos de rede no máximo 3 saltos. Já no tráfego, o agente deve gerar no máximo 25 kbytes/s.

Após a escolha do agente de descoberta que irá realizar as tarefas solicitadas, o seletor de agentes fornece como dados de saída três listas: PROCS, PROCS_R e PARAM. O conteúdo das três listas é o mesmo das listas PROCS, PROCS_R e PARAM dos dados de entrada. Além disso, os dados de saída do seletor de agentes são utilizados como dados de entrada do agente de descoberta, que já foram explicados anteriormente. Após a execução da descoberta de QoS, o seletor de agentes recebe o resultado do agente de descoberta e repassa o resultado imediatamente para a aplicação

de gerenciamento processar os resultados. É importante lembrar que os resultados recebidos do agente de descoberta podem incluir as informações identificadas dos dispositivos, além dos procedimentos que foram realizados.

Repositório de agentes de descoberta

O **repositório de agentes de descoberta** (chamado, a partir de agora, simplesmente de repositório de agentes) armazena a descrição de todos os agentes de descoberta. Assim, quando um novo agente de descoberta for criado, suas definições serão adicionadas no repositório de agentes. Por outro lado, quando um agente de descoberta se tornar obsoleto, suas informações podem ser retiradas do repositório. O repositório de agentes será acessado ainda para atualizar informações de um agente de descoberta existente, fornecer informações ao seletor de agentes para auxiliar na decisão daquele que será escolhido e fornecer informações para instalar uma nova instância de um agente de descoberta.

Base de conhecimento

A **base de conhecimento** possui as associações entre identificadores parciais e recursos de fornecimento de QoS. Ela é acessada pelos agentes que implementam a seleção parcial dos recursos dos dispositivos e deve ser atualizada sempre que houver alteração nas associações entre identificadores parciais e recursos de fornecimento de QoS. Um detalhe importante da base de conhecimento é que ela (ou parte dela) pode ser enviada junto com um agente de descoberta de QoS que implemente a seleção parcial dos recursos de dispositivos, de maneira a acelerar o acesso à base. Um determinado agente que implemente seleção parcial de dispositivos bem específicos, onde poucos identificadores parciais são manipulados, pode ter a parte da base de conhecimento correspondente a esses identificadores junto ao próprio alvo. Enfim, com a atualização na base de conhecimento e do repositório de agentes, a arquitetura se torna extensível, podendo identificar novos recursos de fornecimento de QoS e alvos, atendendo a mais uma característica do modelo.

Base de informações de gerenciamento

A **base de informações de gerenciamento** é o local de armazenamento de todas as informações de descoberta de QoS, de onde as outras tarefas de gerenciamento de QoS poderão recuperar tais informações. Assim, todo elemento que for responsável por armazenar as informações de gerenciamento, faz acessos à base de informações de gerenciamento.

Aplicação de gerenciamento de QoS

Por fim, o último elemento da arquitetura a ser apresentado é a **aplicação de gerenciamento de QoS** (chamada, a partir de agora, simplesmente de aplicação de gerenciamento). A aplicação de gerenciamento é responsável por fazer a interface entre o gerente de rede e a arquitetura de descoberta de QoS. Tal interface permite ao gerente de rede solicitar (seta 1) tarefas relacionadas à descoberta de QoS propriamente dita,

bem como outras de administração de alguns elementos da arquitetura, como será visto adiante. Assim, as requisições para iniciar e parar uma descoberta de QoS, bem como visualizar o andamento da tarefa, ou o seu resultado, serão solicitadas à aplicação de gerenciamento. Ao receber uma solicitação do gerente de rede para realizar uma descoberta de QoS, a aplicação de gerenciamento repassa a solicitação ao devido seletor de agentes, de acordo com os critérios de seleção de agentes escolhidos. Quando a aplicação de gerenciamento receber o resultado da realização desta tarefa deverá tratá-la adequadamente. Se a descoberta foi realizada com sucesso, a aplicação de gerenciamento pode armazenar os resultados na base de informações de gerenciamento e atualizar o estado da execução da tarefa. Caso seja reportado algum erro à aplicação de gerenciamento, ela deverá gerar os devidos relatórios e alertar o gerente de rede. É importante salientar que a aplicação de gerenciamento não é obrigada a armazenar os resultados da descobertas, ficando esta decisão a cargo do gerente de rede (usuário).

Em relação ao armazenamento das informações, podem ocorrer duas situações: ou o agente de descoberta ou a aplicação de gerenciamento (seta 4) se encarrega de armazenar as informações. Por exemplo, um gerente de rede solicita à aplicação de gerenciamento a realização da descoberta de QoS do terceiro e quarto procedimentos (PROCS={3,4}, PROCS_R={} e PARAM={PRD}) para uma determinada rede, onde o agente se encarrega da tarefa de armazenar as informações geradas pela descoberta de QoS. Assim, a aplicação de gerenciamento irá repassar a solicitação para o devido seletor de agentes, de acordo com os critérios de seleção de agentes (CRIT={PROXIMIDADE='< 5 saltos'}) escolhidos pelo gerente de rede. Enquanto a tarefa não termina, o gerente pode solicitar à aplicação de gerenciamento a visualização do andamento da execução da tarefa, onde poderia ser informado o estado atual PROCS_R={3}, ou seja, apenas o terceiro procedimento foi realizado até o momento da consulta. Após o término da execução da tarefa, o gerente pode visualizar o resultado da execução da tarefa, isto é, quais tarefas foram realizadas com sucesso (PROCS_R={3,4}) e as informações geradas pela descoberta de QoS (RESULT={LDR}).

Como já mencionado, as outras tarefas da aplicação de gerenciamento estão relacionadas à administração de alguns elementos da arquitetura: repositório de agentes, agentes de descoberta e base de conhecimento. A administração do repositório de agentes (seta 2) mantém atualizadas as informações dos agentes de descoberta. Assim, por exemplo, quando um novo agente de descoberta for adicionado ao repositório, os seletores de agentes poderão considerá-lo nos seus algoritmos de escolha sem precisar alterar seus códigos fontes. A administração da base de conhecimento (seta 3) visa manter consistente as associações entre identificadores parciais e recursos de fornecimento de QoS. Ela deverá ser atualizada sempre que um novo recurso de fornecimento de QoS for disponibilizado pela rede ou um identificador parcial for alterado.

3.5 Exemplos de descoberta de QoS baseados na metodologia e na arquitetura

Esta seção apresenta alguns cenários de descoberta de QoS com o objetivo de mostrar as interações dos elementos da arquitetura, considerando a metodologia proposta. Três exemplos são apresentados: execução de todos os procedimentos da metodologia de descoberta de QoS, execução do primeiro e segundo procedimento da

metodologia de descoberta de QoS e, por último, execução do terceiro e quarto procedimento.

Exemplo 1: execução de todos os procedimentos da metodologia de descoberta de QoS

No primeiro exemplo será possível verificar como a arquitetura proposta é utilizada na execução de todos os procedimentos da metodologia de descoberta de QoS. A FIGURA 3.15 apresenta o cenário que será utilizado no primeiro exemplo e a TABELA 3.6 apresenta os elementos presentes no cenário. Normalmente, a execução de todos os procedimentos da metodologia será utilizada em situações onde está se começando as atividades de gerenciamento de QoS numa rede, onde ainda não se têm informações a respeito dos recursos de QoS disponíveis.

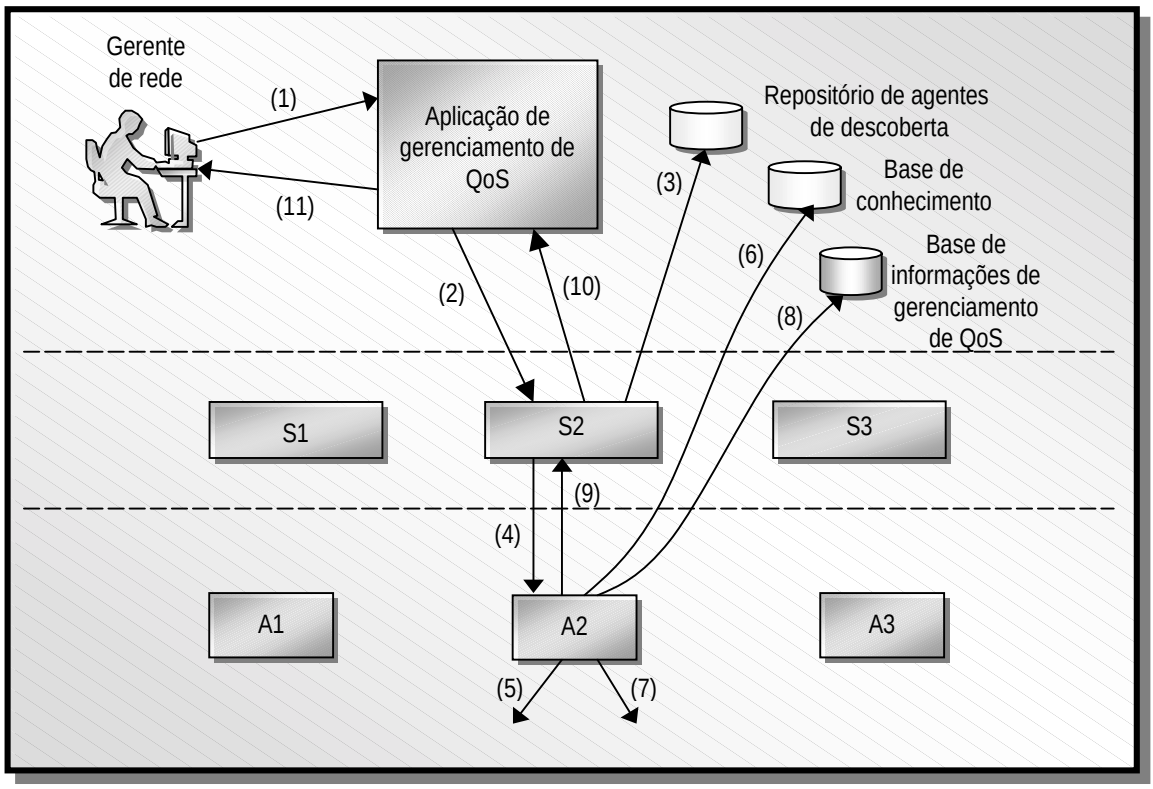

FIGURA 3.15 - Cenário do exemplo 1: execução de todos os procedimentos da metodologia de descoberta de QoS

No cenário específico deste exemplo é imaginada uma arquitetura de descoberta de QoS com três seletores de agentes e três agentes de descoberta, além dos elementos da camada de gerenciamento de QoS: aplicação de gerenciamento, base de conhecimento e base de informações de gerenciamento de QoS. As comunicações envolvidas são representadas por setas, e são numeradas de acordo com sua ordem de execução.

Alguns elementos descritos na TABELA 3.6 merecem destaque. O seletor de agentes S1 implementa um critério de seleção onde o agente escolhido é aquele que implementa exatamente os procedimentos solicitados. Assim, se os procedimentos solicitados para serem executados são {2,3}, o agente escolhido será aquele que

implementa os procedimentos {2,3} preferencialmente e não, por exemplo, um agente que implemente {1, 2, 3, 4}. Quando não existir a correspondência exata, um critério para selecionar um agente de descoberta alternativo é utilizado. Além do critério implementado por S1, o seletor de agentes S2 implementa um critério extra que avalia a proximidade entre o agente de descoberta e os dispositivos envolvidos na descoberta de QoS. Já o seletor de agentes S3 implementa outro critério extra, baseado no desempenho do processador onde o agente de descoberta está executando.

TABELA 3.6 - Elementos da arquitetura de descoberta de QoS dos exemplos 1, 2 e 3

Elemento		Descrição
Aplicação de gerenciamento de QoS		Aplicação de gerenciamento de QoS do exemplo
BC		BC é a base de conhecimento utilizada pelos agentes de descoberta de QoS que implementam o segundo procedimento da metodologia
BIGQ		BIGQ é a base de informações de gerenciamento de QoS
Seletores de agentes	S1	Critério de seleção: correspondência exata entre procedimentos solicitados a serem executados e procedimentos implementados pelo agente de descoberta
	S2	Critérios de seleção: idem ao S1 e proximidade entre agente de descoberta e dispositivos
	S3	Critérios de seleção: idem ao S2 e desempenho do processador
Agentes de descoberta	A1	Implementa o primeiro e o segundo procedimento da metodologia
	A2	Implementa todos os procedimentos da metodologia
	A3	Implementa o terceiro e o quarto procedimento da metodologia

Inicialmente, o gerente de rede solicita (seta 1) a execução da descoberta de QoS informando quais os procedimentos que serão executados, os parâmetros do procedimento inicial e quais critérios para seleção de agentes ele deseja que sejam utilizados. Neste exemplo é suposto que essas informações são as seguintes:

- PROCS={1,2,3,4} // Todos os procedimentos
- PARAM={D}
- CRIT={‘CORRESPONDENCIA_EXATA_PROCS’, PROXIMIDADE=’< 2 SALTOS’}

Assim, é solicitada a execução dos quatro procedimentos da metodologia, informando a lista de dispositivos D em PARAM. Além disso, os critérios para seleção de agentes são a correspondência exata entre procedimentos solicitados pelo gerente de rede e implementados pelo agente de descoberta, e proximidade entre agente de descoberta e dispositivos com menos de dois saltos de distância. Com os critérios solicitados, a aplicação de gerenciamento escolhe o seletor de agentes S2 e envia (seta 2) a solicitação de descoberta de QoS, fornecendo os seguintes dados de entrada do

seletor de agentes:

- PROCS={1,2,3,4}
- PARAM={D}
- PROCS_R={}
- CRIT={‘CORRESPONDENCIA_EXATA_PROCS’, PROXIMIDADE=’< 2 SALTOS’}

Ao receber os dados de entrada, o seletor de agentes S2 irá acessar (seta 3) o repositório de agentes para verificar a descrição dos agentes de descoberta, de maneira a escolher o mais adequado para a solicitação do gerente. Com os agentes disponíveis (A1, A2 e A3), o mais adequado para o primeiro critério (correspondência exata entre procedimentos solicitados e implementados pelo agente) é o agente A2, já que A2 implementa exatamente os procedimentos solicitados. Nesse exemplo é suposto que A2 também está próximo (1 salto) dos dispositivos envolvidos na descoberta de QoS e, por isso, é o agente escolhido pelo segundo critério. Assim, o seletor de agentes S2 envia (seta 4) a solicitação de descoberta de QoS para o agente de descoberta A2, fornecendo os seguintes dados de entrada:

- PROCS={1,2,3,4}
- PARAM={D}
- PROCS_R={}

Para realizar o primeiro procedimento (identificação parcial dos dispositivos), A2 faz as primeiras consultas (seta 5) aos dispositivos de D para obter a lista de identificadores parciais IPD. Após a execução do primeiro procedimento, A2 acessa (seta 6) a base de conhecimento para executar o segundo procedimento (seleção parcial dos recursos dos dispositivos) para obter a lista de associações entre dispositivos e recursos de fornecimento de QoS PRD. Tendo a lista PRD, A2 pode executar o terceiro procedimento (identificação dos alvos), onde os dispositivos em D são novamente consultados (seta 7). Após a consulta aos dispositivos, A2 inicia o quarto procedimento (análise e armazenamento). Após a devida análise, A2 armazena (seta 8) as informações geradas pela descoberta de QoS – que estão contidas no dado de saída RESULT. Então, A2 retorna (seta 9) o resultado da descoberta de QoS para S2, fornecendo as seguintes informações:

- PROCS={1,2,3,4}
- PROCS_R={1,2,3,4}
- RESULT={IPD,PRD,LDR}

S2, ao receber o resultado da descoberta de QoS de A2, repassa (seta 10) as informações para a aplicação de gerenciamento que, por fim, informa (seta 11) o gerente de rede.

Exemplo 2: execução do primeiro e segundo procedimentos da metodologia de descoberta de QoS

No segundo exemplo será verificado como a arquitetura e a metodologia de descoberta de QoS se comportam quando for solicitada a execução dos dois primeiros procedimentos da metodologia, utilizando um critério de seleção de agentes diferente do exemplo anterior. A FIGURA 3.16 apresenta o cenário utilizado e a TABELA 3.6 apresenta os elementos presentes no mesmo cenário. A execução dos dois primeiros procedimentos serve de base para análise das associações entre recursos de QoS e identificadores parciais. Para um gerente de rede, seria possível verificar a correspondência entre as associações de recursos de QoS e identificadores parciais armazenadas na base de conhecimento e os identificadores encontrados na rede.

Inicialmente, o gerente de rede solicita (seta 1) a execução da descoberta de QoS, informando quais os procedimentos que serão executados, os parâmetros do procedimento inicial e quais critérios para seleção de agentes ele deseja que sejam utilizados. Neste exemplo é suposto que essas informações são as seguintes:

- PROCS={1,2}
- PARAM={D}
- CRIT={‘CORRESPONDENCIA_EXATA_PROCS’}

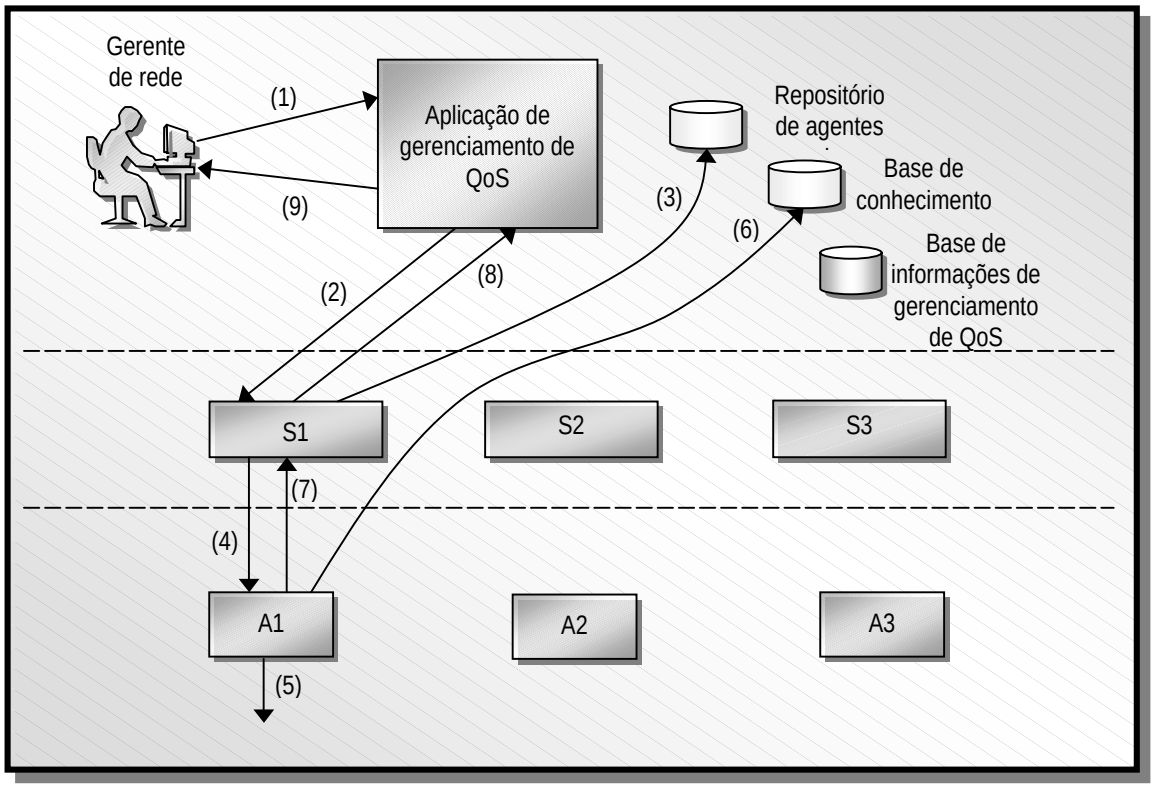

FIGURA 3.16 - Cenário do exemplo 2: execução do primeiro e segundo procedimentos da metodologia de descoberta de QoS

Foi solicitada a execução dos dois primeiros procedimentos da metodologia, informando a lista de dispositivos D em PARAM. Além disso, foi solicitado apenas um

critério para a seleção do agente: correspondência exata entre procedimentos solicitados pelo gerente de rede e implementados pelo agente de descoberta. Com o critério solicitado, a aplicação de gerenciamento escolhe o seletor de agentes S1 e envia (seta 2) a solicitação de descoberta de QoS, fornecendo os seguintes dados de entrada para o seletor de agentes:

- PROCS={1,2}
- PARAM={D}
- PROCS_R={}
- CRIT={‘CORRESPONDENCIA_EXATA_PROCS’}

Ao receber os dados de entrada, o seletor de agentes S1 irá acessar o repositório de agentes (seta 3) para verificar a descrição dos agentes de descoberta, de maneira a escolher o mais adequado para a solicitação do gerente. Com os agentes disponíveis (A1, A2 e A3), o mais adequado para o primeiro critério solicitado (correspondência exata entre procedimentos solicitados e implementados pelo agente) é o agente A1, já que A1 implementa exatamente os procedimentos solicitados. Assim, o seletor de agentes S1 envia (seta 4) a solicitação de descoberta de QoS para o agente de descoberta A1, fornecendo os seguintes dados de entrada:

- PROCS={1,2}
- PARAM={D}
- PROCS_R={}

Para realizar o primeiro procedimento (identificação parcial dos dispositivos), A1 faz as primeiras consultas (seta 5) aos dispositivos em D para obter a lista de identificadores parciais IPD. Após a execução do primeiro procedimento, A2 acessa (seta 6) a base de conhecimento para executar o segundo procedimento (seleção parcial dos recursos dos dispositivos) para obter a lista de associações entre dispositivos e recursos de fornecimento de QoS PRD. Tendo completado todos os procedimentos solicitados, A1 retorna (seta 7) o resultado da descoberta de QoS para S1, fornecendo as seguintes informações:

- PROCS={1,2}
- PROCS_R={1,2}
- RESULT={IPD,PRD}

S2, ao receber o resultado da descoberta de QoS de A2 repassa (seta 8) as informações para a aplicação de gerenciamento que, por fim, informa (seta 9) o gerente de rede.

Exemplo 3: execução do terceiro e quarto procedimentos da metodologia de descoberta de QoS

O terceiro e último exemplo apresentado irá demonstrar o comportamento da arquitetura quando a execução do terceiro e quarto procedimentos da metodologia for solicitada, utilizando mais critérios de seleção que os exemplos anteriores. Esse exemplo é a continuação do exemplo anterior, onde o gerente desejava utilizar critérios diferentes para escolher o agente de descoberta que executa o primeiro e segundo

procedimentos da metodologia. Assim, o parâmetro PRD contém valores fornecidos como dado de saída do exemplo anterior. A FIGURA 3.17 apresenta o cenário que será utilizado pelo terceiro exemplo e a TABELA 3.6 apresenta novamente os elementos presentes no cenário, os mesmos utilizados nos exemplos anteriores.

Inicialmente, o gerente de rede solicita (seta 1) a execução da descoberta de QoS, informando quais os procedimentos que serão executados, os parâmetros do procedimento inicial e quais critérios para seleção de agentes ele deseja que sejam utilizados. Neste exemplo é suposto que essas informações são as seguintes:

- PROCS={3,4}
- PARAM={PRD}
- CRIT={‘CORRESPONDENCIA_EXATA_PROCS’, PROXIMIDADE=’< 4 saltos’, DESEMPENHO_PROCESSADOR=’médio’}

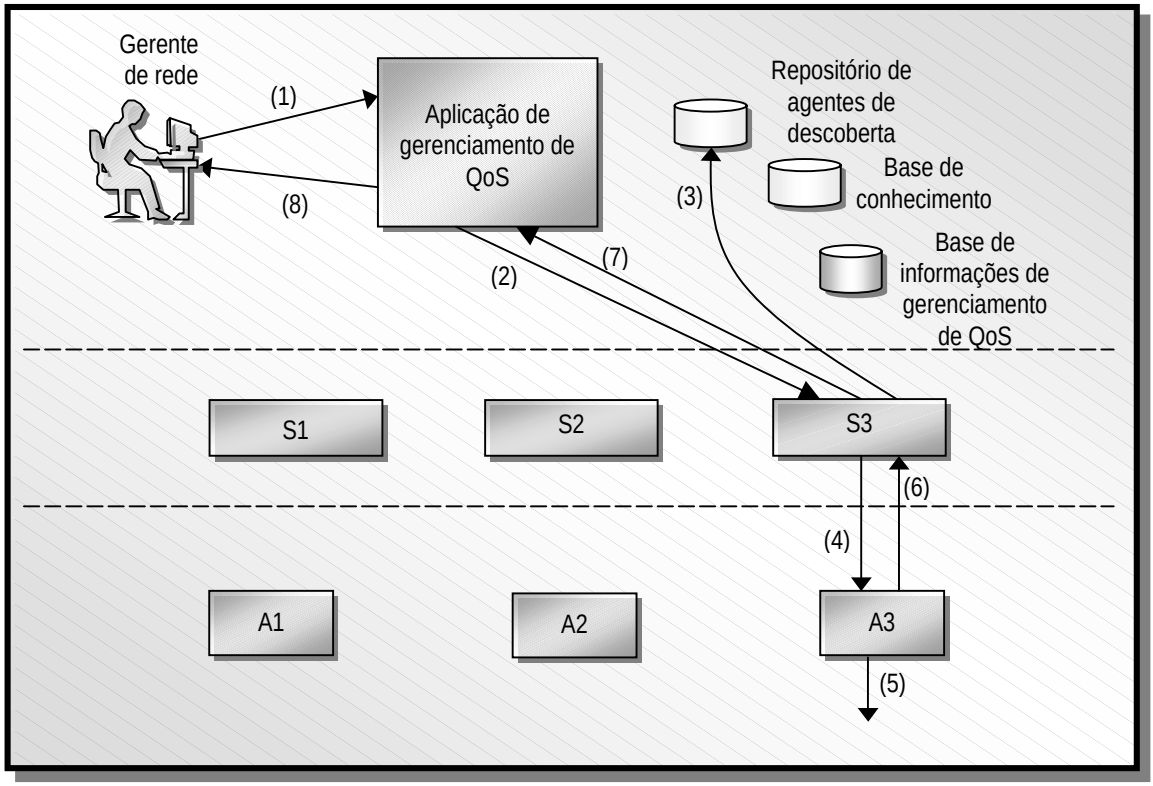

FIGURA 3.17 - Cenário do exemplo 3: execução do terceiro e quarto procedimento da metodologia de descoberta de QoS

Como pode ser visto, o gerente de rede solicitou a execução do terceiro e quarto procedimentos da metodologia, informando a lista de associações entre dispositivos e recursos de QoS PRD em PARAM. Além disso, foram solicitados três critérios para a seleção do agente:

(1) Correspondência exata entre procedimentos solicitados pelo gerente de rede e implementados pelo agente de descoberta.

(2) Distância entre agente de descoberta e dispositivos envolvidos menor que quatro saltos.

(3) Desempenho do processador onde o agente é executado com desempenho

médio.

Com os critérios solicitados, a aplicação de gerenciamento escolhe o seletor de agentes S3 e envia (seta 2) a solicitação de descoberta de QoS, fornecendo os seguintes dados de entrada para o seletor de agentes:

- PROCS={3,4}
- PARAM={PRD}
- PROCS_R={}
- CRIT={‘CORRESPONDENCIA_EXATA_PROCS’, PROXIMIDADE=’< 4 saltos’, DESEMPENHO_PROCESSADOR=’médio’}

Ao receber os dados de entrada, o seletor de agentes S3 irá acessar o repositório de agentes para verificar a descrição dos agentes de descoberta, de maneira a escolher o mais adequado para a solicitação do gerente. Com os agentes disponíveis (A1, A2 e A3), o mais adequado para o primeiro critério solicitado (correspondência exata entre procedimentos solicitados e implementados pelo agente) é o agente A3, já que A3 implementa exatamente os procedimentos solicitados. Pelo critério PROXIMIDADE, é suposto neste exemplo que o agente de descoberta mais adequado é o agente A2. Por fim, pelo critério DESEMPENHO_PROCESSADOR, é suposto neste exemplo que o agente de descoberta mais adequado é A3. Utilizando um critério de desempate (que não será descrito nem analisado por não fazer parte do foco deste exemplo), o agente S3 foi escolhido. Um possível critério de desempate é a ordem em que os agentes estão armazenados no repositório de agentes, onde o último seria o escolhido.

Assim, o seletor de agentes S3 envia (seta 4) a solicitação de descoberta de QoS para o agente de descoberta A1, fornecendo os seguintes dados de entrada:

- PROCS={3,4}
- PARAM={PRD}
- PROCS_R={}

Para realizar o terceiro procedimento (identificação dos alvos), A3 faz as consultas (seta 5) aos dispositivos da lista D para obter a lista de recursos de fornecimento de QoS LDR. Após a execução do terceiro procedimento, A3 realiza a análise das informações obtidas. Tendo completado todos os procedimentos solicitados, A3 retorna (seta 6) o resultado da descoberta de QoS para S3, fornecendo as seguintes informações:

- PROCS={3,4}
- PROCS_R={3,4}
- RESULT={LDR}

S3, ao receber o resultado da descoberta de QoS de A3, repassa (seta 7) as informações para a aplicação de gerenciamento que, por fim, informa (seta 8) o gerente de rede (usuário). É importante ressaltar que as informações obtidas através da execução dos agentes não foram armazenadas na base de informações de gerenciamento de QoS, deixando claro que a tarefa não é obrigatória. Além disso, tal tarefa pode ser executada posteriormente.

4 Implementação do modelo de descoberta de QoS

O capítulo anterior apresentou uma proposta de modelo de descoberta de QoS, bem como os requisitos que guiaram a elaboração do mesmo. Como foi visto, tal modelo é composto de uma metodologia e uma arquitetura, separadas dessa maneira para facilitar possíveis evoluções. Esse capítulo irá apresentar uma implementação que demonstra as principais características do modelo: (1) possibilidade de estender as funcionalidades de descoberta de QoS, (2) distribuir as tarefas em nodos de processamento fisicamente separados e (3) executar a descoberta em diferentes etapas, cada uma representada por um procedimento da metodologia.

Além de seguir as características do modelo, a implementação realizada possui outras características mais específicas. São elas:

- **Rede alvo**. O modelo não define o tipo de rede dos alvos a serem identificados nem o da ferramenta de descoberta de QoS. Essa implementação irá considerar apenas redes IP como infraestrutura lógica dos alvos e da ferramenta de descoberta de QoS. Isso pode incluir dispositivos de redes ATM, desde que tais dispositivos possam ser gerenciados através de MIBs, como a AToMIB [AHM 94], e que os nodos dessa rede possam ser acessados via IP.
- **Plataforma para delegação de agentes de descoberta**. O modelo define que a descoberta de QoS deve ser realizada de maneira distribuída. Como não é objetivo deste trabalho desenvolver um mecanismo de distribuição de tarefas, a plataforma Jasmin (*A Java Script-MIB Implementation*) [TU 2000] é utilizada para tal finalidade. Como será visto oportunamente, cada agente de descoberta será implementado por um *script* de gerenciamento da plataforma Jasmin.
- **Dispositivos gerenciados**. Os dispositivos gerenciados devem possuir prioritariamente agentes SNMP, para que seus alvos sejam identificados. Além de ser um protocolo padronizado e amplamente implementado nos dispositivos de rede, as informações fornecidas pelas MIBs atuais têm auxiliado bastante na identificação dos alvos, como será visto na seqüência desse capítulo. Embora as MIBs sejam a maior fonte de informação na identificação dos alvos, outros mecanismos também foram estudados, tal como um baseado no serviço TELNET [POS 83].
- **Autonomia dos módulos**. Em ambientes distribuídos colaborativos é comum haver *overhead* de comunicação, o que pode ser um problema para as aplicações que executam nesses ambientes. Em primeiro lugar, é desejado que o tráfego de mensagens de aplicações de gerenciamento não interfira no desempenho do tráfego de usuários. Em segundo lugar, o excesso de mensagens pode degradar o desempenho global do sistema de gerenciamento, inviabilizando sua execução. Assim, essa implementação irá favorecer a autonomia de seus módulos com o objetivo de evitar tráfego. É importante salientar que essa implementação não tem o objetivo de obter alto desempenho na descoberta, apenas pretende que a comunicação entre seus módulos não seja um transtorno para sua execução. Em alguns casos, a autonomia pode gerar a duplicação de funcionalidades entre os módulos.

- **API de descoberta de QoS**. Embora as GUIs possam facilitar a interação com o sistema, essa implementação pretende disponibilizar também uma interface extra para linha de comando (CLI) para que os gerentes de rede possam realizar suas principais tarefas em um *shell* de comando, ou por outros aplicativos capazes de manipular *shell scripts*. Tal facilidade pode auxiliar o gerente de rede a automatizar suas tarefas mais específicas, não cobertas por esta implementação.

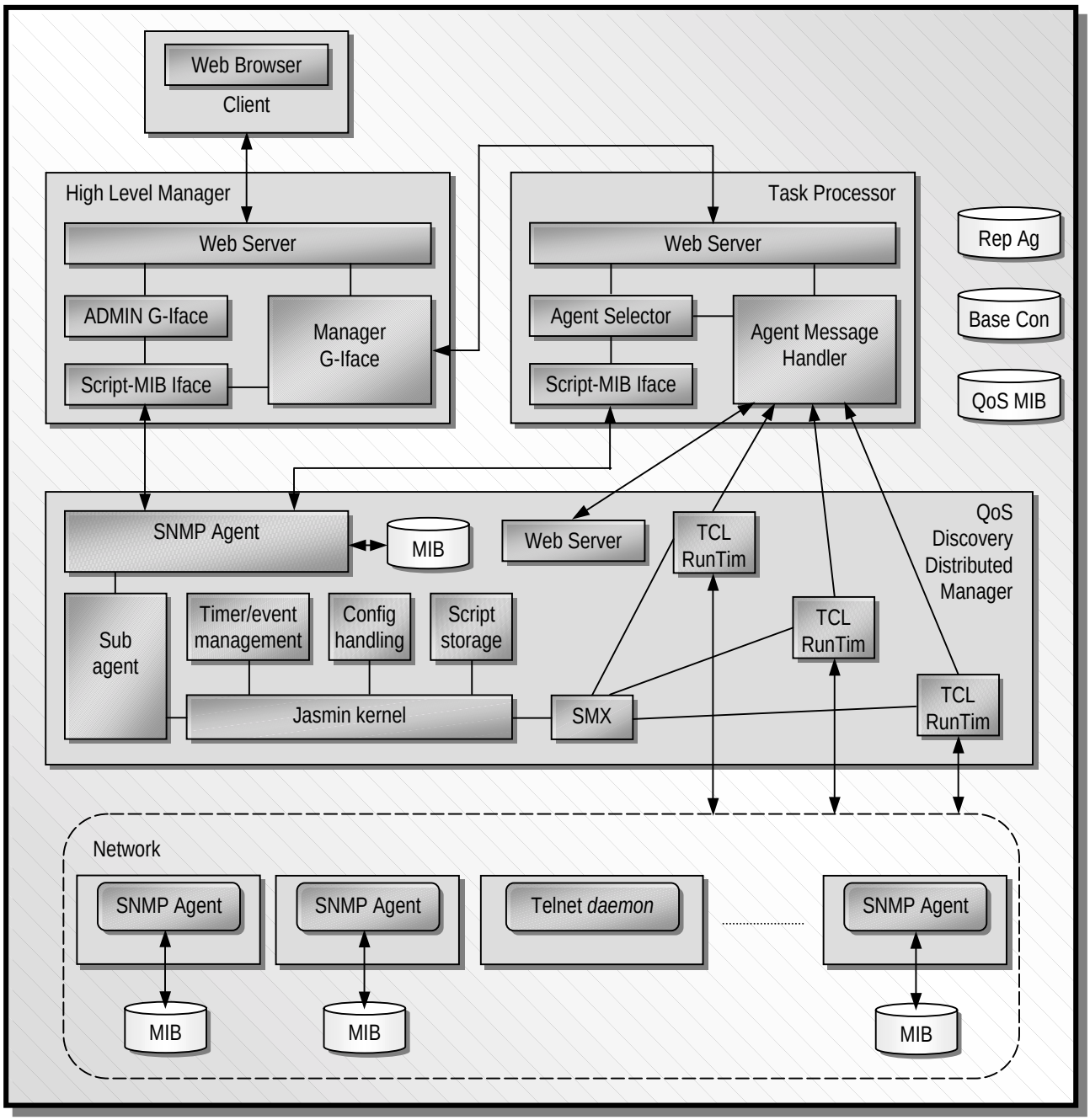

FIGURA 4.1 - Arquitetura da implementação do modelo

Tendo base no modelo de descoberta de QoS e nas características específicas da implementação apresentadas anteriormente, a FIGURA 4.1 apresenta a arquitetura do sistema de descoberta de QoS proposto. Em linhas gerais, a arquitetura é composta de cinco módulos principais, cada módulo dividido em sub-módulos, e cada sub-módulo

podendo ser dividido em grupos funcionais. É importante salientar que toda a implementação é executada exclusivamente na plataforma GNU/Linux e utiliza pacotes de *software* aberto. Por outro lado, como o sistema é baseado na Web, o gerente de rede pode ter acesso às funcionalidades da implementação através de um navegador Web instalado em qualquer plataforma.

4.1 Módulo *Network*

O módulo *Network* representa dispositivos reais de rede, dos quais a descoberta de QoS irá tentar identificar os alvos. Entre esses dispositivos há aqueles que possuem alvos e outros que não são elementos que podem fornecer serviços com QoS. Assim, nesse módulo encontraremos equipamentos de redes tais como roteadores, *switches* e *hubs*.

4.2 Módulo *Qos Discovery Distributed Manager* (QoSD-DM)

O módulo *QoS Discovery Distributed Manager* (QoSD-DM) tem por objetivo manter *scripts* de gerenciamento, que são os responsáveis pela descoberta de QoS propriamente dita. Assim, o *script* de gerenciamento representa o agente de descoberta de QoS da arquitetura do modelo, e ao longo desse texto estes termos serão utilizados como sinônimos. Nessa implementação o QoSD-DM é implementado com base em um agente SNMP, que implementa a Script-MIB [LEV 2001]. O agente SNMP utilizado é o NET-SNMP [BAE 2002], enquanto a implementação da Script-MIB utiliza o Jasmin, uma espécie de "*add-on*" ao NET-SNMP. A ativação de uma tarefa de descoberta de QoS é realizada através de mensagens SNMP que instalam um *script* de gerenciamento, o respectivo *launch button* e, por último, executam este *launch button*. Um *launch button* é uma instância de um *script* de gerenciamento com parâmetros de execução e parâmetros de processamento. Para sua execução é necessário apenas um *set* de objeto na Script-MIB. Mais detalhes sobre a Script-MIB e o Jasmin podem ser encontrados no *site* do projeto [TU 2000]. Ao término da tarefa, o *script* de gerenciamento envia uma *trap* SNMP ao sub-módulo *Agent Message Handler* do módulo *Task Processor*, descrito adiante. Para realizar tal tarefa, o QoSD-DM possui os seguintes sub-módulos: *Web Server*, *SNMP Agent*, *sub agent*, *Jasmin kernel*, *Timer/event management*, *config handling*, *script storage*, *SMX* e *TCL RunTime*.

Web Server

O sub-módulo *Web Server* tem como principal objetivo gerenciar as conexões HTTP/HTTPS. Através dessas conexões, o QosD-DM pode fornecer ao sub-módulo *Agent Message Handler* do módulo *Task Processor*, ambos descritos adiante, o arquivo com o resultado da descoberta de QoS realizada por um de seus agentes de descoberta. A descrição do mecanismo de obtenção do resultado de uma descoberta de QoS será apresentada em detalhes junto com a descrição do sub-módulo *Agent Message Handler*.

Sub-módulos SNMP Agent, Jasmin kernel, Timer/Event management, Config handling, Script Storage e *SMX*

O sub-módulo *SNMP Agent* implementa o agente SNMP, recebendo todas as solicitações de descoberta de QoS em forma de mensagens SNMP e repassando para o sub-módulo *sub agent* aquelas mensagens destinadas a objetos que o *SNMP Agent* não implementa. Quando uma mensagem é destinada a algum objeto da Script-MIB, o *sub agent* repassa a mensagem para o sub-módulo J*asmin kernel*, que implementa as funções *core* da Script-MIB. As principais funções do *Jasmin kernel* incluem iniciar, manter e terminar a execução da Script-MIB. O sub-módulo *Timer/event management* basicamente controla eventos programados do Jasmin, tal como a validade temporal de alguns objetos da Script-MIB. Já o sub-módulo *Config handling* controla todas as configurações do Jasmin, tal como uma lista dos *scripts* que possuem armazenamento não volátil. O sub-módulo *Script storage* armazena os *scripts* de gerenciamento que são carregados na ativação do agente SNMP, bem como aqueles que são carregados enquanto o agente SNMP está executando. Aliás, essa última característica merece destaque, pois evita que o agente SNMP precise ser reiniciado a cada alteração de funcionalidade nos *scripts* de gerenciamento. O sub-módulo SMX implementa o protocolo de mesmo nome SMX (Script-MIB *Extensibility Protocol Version 1.1*) [SCH 2001], utilizado para comunicação entre o sistema de execução específico de linguagem de programação com a implementação da Script-MIB independente de linguagem. Neste caso, o sistema de execução específico de linguagem de programação é implementado pelo TCL *RunTime*, descrito a seguir.

Sub-módulo TCL RunTime

O sub-módulo *TCL Runtime* é o ambiente de execução de *scripts* de gerenciamento que implementam as funções da descoberta de QoS. Os *scripts* de gerenciamento são totalmente implementados em TCL. O ambiente de execução gerencia, entre outros aspectos, perfis de segurança de execução, onde é possível restringir determinadas instruções. Embora essa implementação não se preocupe com aspectos de segurança, uma futura abordagem pode utilizar estas facilidades para tornar o sistema mais completo. Para poder executar os *scripts* de gerenciamento de descoberta de QoS, o perfil de política de segurança precisou ser alterado para que algumas instruções não permitidas pela configurações padrão fossem adicionadas ao *TCL Safe Base* [OUS 2000].

Para cada *script* de gerenciamento a ser executado, um novo módulo *TCL Runtime* deve ser instanciado. A FIGURA 4.1 apresenta três *TCL Runtimes*, podendo haver outros executando. Ainda é possível controlar o número de instâncias de *TCL Runtime* por *script* de gerenciamento registrado na Script-MIB. Os agentes implementados são: SnmpCiscoClassMIB, SnmpDiffServ, SnmpAToMIB e SnmpFull. O agente SnmpCiscoClassMIB identifica alvos que implementam a CISCO-CLASS-BASED-QOS-MIB (apenas alvos que implementem a tabela cbQosServicePolicyTable) [CIS 2001a], suportada em alguns dispositivos da Cisco. O agente SnmpDiffServ identifica alvos que implementam DiffServ, enquanto o SnmpAToMIB identifica alvos que implementam ATM. O agente SNMPFull agrega todas as funcionalidades dos outros agentes.

É importante salientar que todos os agentes de descoberta possuem uma interface e código mínimo em comum, o que facilita a padronização da parametrização do *script*. A TABELA 4.1 apresenta a interface mínima de parâmetros que um agente de descoberta de QoS deve implementar. A interface mínima está relacionada basicamente com parâmetros relacionados ao controle da tarefa de descoberta de QoS, bem como com a indicação dos procedimentos da metodologia.

TABELA 4.1 - Interface mínima de um agente de descoberta de QoS.

Parâmetro	Descrição	Exemplo de valores
-h	Indica os endereços IPs dos dispositivos a serem identificados.	"200.132.73.34", ou "200.132.73.34 200.228.138.20"
-net	Indica o endereço IP da sub-rede dos dispositivos a serem identificados.	"200.132.73.0"
-mask	Máscara de rede dos dispositivos a serem identificados. Juntamente com o parâmetro –net, esse parâmetro indica como obter a lista de endereços IPs dos dispositivos a serem identificados.	"255.255.255.0"
-pid	Indica o identificador da tarefa de descoberta de QoS. Será utilizado para armazenar o resultado da descoberta de QoS.	"666"
-tp	Indica o endereço IP do módulo Task Processor, o qual o agente de descoberta deve informar o resultado da descoberta de QoS.	"200.132.73.102"
-phase	Indica as fases da metodologia a serem executadas.	"234" (está sendo solicitado a execução do segundo, terceiro e quarto procedimento)
-pph2	Indica o arquivo de parâmetros a ser utilizado na fase 2.	"http://soyuz.metropoa.tche.br/folder/file2.txt"
-pph3	Indica o arquivo de parâmetros a ser utilizado na fase 3.	"http:// soyuz.metropoa.tche.br /folder/file3.txt"
-pph4	Indica o arquivo de parâmetros a ser utilizado na fase 4.	"http:// soyuz.metropoa.tche.br /folder/file4.txt"

4.3 Módulo *Task Processor* (TP)

Os objetivos do módulo *Task Processor* (TP) estão relacionados à monitoração da execução propriamente dita das tarefas de descoberta de QoS. Ele possui duas funções principais: (1) selecionar agentes de acordo com as características da solicitação da descoberta de QoS e (2) processar os eventos/mensagens recebidas dos agentes de descoberta de QoS. Para tal, o TP distribui suas tarefas em quatro sub-módulos: *Web Server*, *Agent Selector*, *Agent Message Handler* e *Script-MIB Iface*.

Sub-módulo Web Server

O sub-módulo *Web Server* tem por principal objetivo gerenciar as conexões HTTP/HTTPS. Através dessas conexões as funcionalidades dos sub-módulos *Agent Selector* e *Agent Message Handler* são acessadas. O *Web Server* pode ainda, através de configurações de servidor, restringir os usuários a uma lista pré-definida e o número de acessos simultâneos. A instância de *Web Server* escolhido para essa implementação é o Apache [APA 99] versão 1.3.X. Além disso, o Apache está configurado para suportar *scripts* PHP, base tecnológica para implementação de outros sub-módulos deste sistema.

Sub-módulo Agent Selector

O sub-módulo *Agent Selector* (AS) tem por objetivo escolher um agente de descoberta de acordo com os parâmetros de solicitação de descoberta de QoS. Para realizar a seleção de um agente de descoberta, esse sub-módulo recebe a solicitação do sub-módulo *web server*, processa os parâmetros, escolhe o agente e invoca o sub-módulo *Script-MIB Iface*, que realizará de fato a ativação do agente de descoberta. Implementado em *script* PHP, esse sub-módulo é parametrizado através de formulários que utilizam o método HTTP/POST [FIE 99]. Os parâmetros suportados nesta implementação são: (1) procedimentos (etapas) da descoberta de QoS e (2) tipo de alvos que se deseja identificar. Os possíveis valores dos parâmetros (que representam os critérios de seleção) implementados são:

- **Procedimento a ser executado**. É possível selecionar a execução dos procedimentos individualmente ou todos em uma tarefa de descoberta.
- **Tipo de alvo a ser identificado**. Os tipos de alvos que podem ser identificados são: DiffServ, CISCO-CLASS-BASED-QOS-MIB e AToMIB.

Quando o sub-módulo não consegue identificar um agente de descoberta adequado, o agente de descoberta padrão é selecionado, neste caso o SnmpFull. O agente de descoberta padrão agrega todas as funcionalidades dos agentes de descoberta em um só. O tipo de agente está fortemente ligado ao tipo de alvo que ele pode identificar.

Sub-módulo Agent Message Handler

O sub-módulo *Agent Message Handler* (AMH) é responsável por tratar todos os eventos originados pelos agentes de descoberta. Tipicamente tais eventos informarão o término de uma tarefa de descoberta bem como o resultado da operação, e são implementados através de *traps* SNMP. Como esta tarefa demanda que o tratador das *traps* esteja rodando continuamente, optou-se por implementar o sub-módulo AMH como um *daemon* TCL que fica esperando exclusivamnte por *traps*. É importante ressaltar que o emissor da *trap* é o agente de descoberta (*script* de gerenciamento), ao invés do Jasmin ou o agente SNMP. Tanto o Jasmin como o agente SNMP não têm primitivas implícitas que informem a um *host* remoto o término da execução de um *script* de gerenciamento em forma de evento assíncrono. Então, para obter o resultado da execução do *script* de gerenciamento, poderia ser utilizado um mecanismo do tipo

polling, que ficasse consultando a referida MIB periodicamente. Como a quantidade de agentes SNMP rodando simultaneamente pode aumentar razoavelmente, e considerando que um dos principais objetivos desse trabalho é reduzir o tráfego de gerenciamento, optou-se em sinalizar os eventos de agentes de descoberta (scripts de gerenciamento) através de *traps* SNMP. A TABELA 4.2 apresenta os eventos e ações que são implementadas pelo AMH. Como pode ser visto na TABELA 4.2, dois eventos são identificados: um trata as tarefas que tiveram êxito, enquanto o outro trata aquelas que não conseguiram completar a operação. O formato da mensagem que identifica o evento é bastante simples. No primeiro campo é informado o endereço IP do QoSD-DM que estava executando o agente de descoberta, enquanto no segundo, o estado do resultado da descoberta de QoS. Se o campo contiver o valor QOSDOK, a descoberta terminou normalmente. Caso o campo apresente o valor QOSDFAIL, a descoberta de QoS não chegou ao final. O terceiro campo é deixado livre para o agente de descoberta colocar uma mensagem qualquer. No último campo, o agente de descoberta fornece a URL do arquivo gerado pela realização da tarefa de descoberta de QoS.

TABELA 4.2 - Eventos e ações implementadas pelo sub-módulo AMH

Evento	Mensagem	Ação
Fim de execução de descoberta de QoS com sucesso.	[IP QoSD-DM][QOSDOK][DESCR][URL]	Baixar o arquivo com o resultado da descoberta na url URL. Atualizar estado da tarefa no respectivo arquivo de controle.
Fim de execução de descoberta de QoS com erro.	[IP QoSD-DM][QOSDFAIL][DESCR][URL]	Baixar o arquivo com o resultado da descoberta na url URL. Atualizar estado da tarefa no respectivo arquivo de controle e registrar a falha.

Sub-módulo Script-MIB Iface

O sub-módulo *Script-MIB Iface* (SM-I), juntamente com o sub-módulo *Agent Message Handler*, é um dos mais importantes sub-módulos, devido à função que desempenha na arquitetura: implementa funcionalidades que fazem parte do *core* do sistema. Seu principal objetivo é fornecer uma interface de comunicação com os *scripts* de gerenciamento. Além disso, fornece uma API que pré-processa informações geradas pelos *scripts* de gerenciamento e apresenta tais informações de maneira mais estruturada, facilitando o desenvolvimento de *scripts* PHP para as GUIs. Este módulo é composto de vários *scripts* totalmente desenvolvidos em TCL. A TABELA 4.3 apresenta a lista dos principais *scripts* TCL implementados. Como os agentes de descoberta são executados em agentes SNMP que implementam a Script-MIB, este sub-módulo possui primitivas internas que acessam os objetos desta MIB. Por fim, como este módulo é implementado através de *scripts* TCL, o gerente de rede pode utilizá-los para implementar outras tarefas mais específicas, não abordadas nessa implementação, ou por tarefas não ligadas à descoberta de QoS que poderiam utilizar essa interface, tal

como uma aplicação de gerenciamento que deseja distribuir *scripts* de gerenciamento que implementam monitoração de QoS.

TABELA 4.3 - Scripts implementados.

Arquivo	Função
getInstalled.tcl	Obtém a lista dos scripts de gerenciamento instalados num QoSD-DM.
getLaunchs.tcl	Obtém a lista dos launch buttons instalados num QoSD-DM.
getRunningScripts.tcl	Obtém a lista dos scripts de gerenciamento em execução num QoSD-DM.
installScript.tcl	Instala um script de gerenciamento num QoSD-DM.
installLaunch.tcl	Instala um launch button num QoSD-DM.
runLaunch.tcl	Inicia a execução de um launch button num QoSD-DM.

4.4 Módulo *High Level Manager* (HLM)

Como pode ser percebido na descrição dos módulos anteriores, as funções realizadas estavam mais relacionadas com o *core* do sistema, ora realizando comunicação entre entidades da arquitetura, ora processando o resultado de tarefas de descoberta de QoS. Como será visto, o módulo HLM possui funcionalidades relacionadas com a interação com o usuário, ora em forma de relatórios, ora em forma de formulários. O módulo *High Level Manager* (HLM) tem por objetivo controlar a ativação de todas as operações realizadas pela descoberta de QoS, bem como apresentar seus resultados. Além disso, este módulo é encarregado de administrar a configuração do sistema como um todo. Para tal, ele é composto de quatro sub-módulos: *Web Server*, *Manager G-Iface*, *ADMIN G-Iface* e *Script-MIB Iface*.

Sub-módulo Web Server

Este sub-módulo é equivalente ao sub-módulo *Web Server* do módulo TP em relação ao seu objetivo primário: gerenciar conexões HTTP. A principal diferença é que, nesse caso, ele gerenciará conexões HTTP/HTTPS que são destinadas ao *Manager G-Iface* ou ao *ADMIN G-Iface*. Tanto a implementação quanto a versão escolhida aqui é a mesma utilizada pelo sub-módulo do TP: Apache versão 1.3.X. As instâncias são preferencialmente diferentes, pois isso melhora a escalabilidade do sistema.

Sub-módulo Manager G-Iface (MG-I)

O sub-módulo *Manager G-Iface* (MG-I) tem por objetivo fornecer ao gerente de rede mecanismos para poder realizar solicitações de descoberta de QoS, bem como obter estados de andamento e relatórios destas solicitações. Para tal, este sub-módulo é

separado em três grupos funcionais: *QoS request*, *request status* e *request report*. Por possuir características mais relacionadas de interação com o usuário, os grupos funcionais deste sub-módulo implementam basicamente GUIs e pré-processam informações para serem repassadas a outras entidades da arquitetura que implementam funcionalidades que fazem parte do *core* da arquitetura, como será visto.

TABELA 4.4 - Propriedades registradas numa tarefa de descoberta de QoS.

Propriedade	Função
script-name	Nome do arquivo do *script* a ser executado.
Script-instance	Instância registrada na Script-MIB.
script-par	Parâmetros específicos do script de gerenciamento.
distr-man	Endereço IP do QoSD-DM.
date-time	Instala um script de gerenciamento num QoSD-DM.

A seguir os grupos funcionais são descritos:

- *QoS request*. Este grupo é responsável por fornecer ao usuário meios para solicitar tarefas de descoberta de QoS bem como ativá-las. Para solicitar uma descoberta de QoS o usuário pode utilizar o **modo manual** ou o **assistente,** ambos implementados inteiramente em *scripts* PHP. O assistente de descoberta de QoS tem a vantagem de facilitar a escolha do agente de descoberta através de uma GUI mais amigável. O modo manual permite ao gerente de rede escolher o agente de descoberta a ser executado e onde ele deve ser executado. Após a solicitação do gerente, a tarefa é registrada e a solicitação é repassada para o sub-módulo *Script-MIB Iface*, que ativa de fato a descoberta de QoS. A TABELA 4.4 lista as propriedades que são registradas para uma tarefa de descoberta de QoS. Através do assistente de solicitação de descoberta de QoS, o gerente de rede solicita que tipo(s) de alvo deseja identificar e o próprio assistente escolhe o agente a ser executado. Para se ativar uma descoberta de QoS o usuário deve parametrizar o assistente, que escolherá um seletor de agentes, que por sua vez se encarregará de registrar e ativar a tarefa propriamente dita.
- *Request status*. Este item é responsável por obter o estado de uma tarefa de descoberta de QoS. Para tal são utilizados dois formulários de GUI, baseados em *scripts* PHP: o primeiro lista todas as tarefas não terminadas, enquanto o segundo detalha o estado da tarefa. As informações fornecidas por esse módulo são as propriedades listadas na TABELA 4.4 e estado da execução do agente.
- *Request report*. Este item é responsável por apresentar o relatório final de execução de uma tarefa de descoberta de QoS. De maneira análoga ao *request status*, tal funcionalidade é implementada por dois *scripts* PHP: o primeiro lista todas as tarefas, enquanto o segundo detalha o estado da tarefa.

As informações fornecidas incluem as propriedades da TABELA 4.4, resultado da execução, uma mensagem de saída do script de gerenciamento e a URL do arquivo que contém o resultado da tarefa.

Sub-módulo ADMIN G-Iface

O sub-módulo *ADMIN G-Iface* tem como principal objetivo administrar os agentes de descoberta de QoS em todos os aspectos relacionados à plataforma Jasmin. Nessa implementação, entende-se que administrar os agentes de descoberta de QoS significa instalar, alterar e remover agentes. Além disso, cada agente de descoberta é representado por um *script* de gerenciamento no Jasmin. Assim, o sub-módulo *ADMIN G-Iface* implementa as GUIs necessárias para o fornecimento de suas funcionalidades através de *scripts* PHP. O *core* das funcionalidades é implementado pelo sub-módulo *Script-MIB Iface* e é acessado por uma interface semelhante à linha de comando. Além disso, idealmente esse módulo é responsável por administrar a configuração do sistema como um todo. Entretanto, o atual estágio de implementação não possui tal funcionalidade e deverá ser revisto em futuras abordagens.

Sub-módulo Script-MIB Iface

Este sub-módulo é equivalente ao sub-módulo SM-I do módulo QosD-DM, sendo apenas uma instância diferente. Analogamente ao sub-módulo *Web Server*, dependendo do cenário de instalação esta instância do sub-módulo SM-I pode ser a mesma presente no módulo HLM, isto é, ambos os módulos estariam sendo executados no mesmo nodo de processamento.

4.5 **Módulo** *Client*

O módulo *Client*, apresentado na FIGURA 4.1, implementa o suporte à interface gráfica do usuário (GUI), na qual o gerente de rede irá acessar todas as funcionalidades das ferramentas de descoberta de QoS. É composto basicamente por navegador *web*, que precisa interpretar javascript e HTML 4.0, tecnologias utilizadas por alguns *scripts* PHP dos módulos *High Level Manager* (HLM) e *Task Processor* (TP), descritos a seguir. Devido à natureza do acesso (cliente/servidor) de um navegador *web* a um servidor *web*, o gerente de rede pode acessar remotamente as funcionalidades de descoberta de QoS. Isto é uma vantagem, pois é possível desassociar fisicamente a solicitação de descoberta com a sua execução, agilizando o gerenciamento da rede propriamente dito. Com a grande disseminação de navegadores *web*, praticamente tem sido garantida a presença desse aplicativo na maioria dos sistemas operacionais mais difundidos. Por isso, a utilização de uma ferramenta desse tipo facilita ao gerente de rede acessar urgentemente o sistema caso seja preciso, o que comumente ocorre em caso de pane. Em situações que demandem urgência no acesso ao sistema de gerenciamento de descoberta de QoS, o acesso via navegador *web* pode acelerar o processo se comparado com um sistema que possua interface proprietária, *fat-client*, etc.

Não é objetivo desse trabalho modificar o protocolo padrão de comunicação entre um navegador *web* e um servidor *web*, tão pouco se demonstrou necessário tal

fato. Assim, o protocolo utilizado entre o módulo Client e o servidor *web* do HLM é HTTP ou HTTPS, não havendo diferença sob o ponto de vista funcional dessa implementação. A utilização do HTTPS, nesse contexto, aumenta a privacidade das informações transferidas, mas não é um requisito obrigatório. Além disso, o uso do HTTP auxilia bastante no processo de depuração do sistema.

5 Conclusões e Trabalhos Futuros

Essa dissertação de mestrado apresentou um modelo para descoberta de QoS em redes de computadores. Como visto no capítulo1, a principal motivação para esse trabalho vem do fato de que atualmente não existem suporte e metodologia adequados para a investigação das facilidades que ajudam no fornecimento de QoS dentro dos dispositivos de uma rede. O capítulo 2 definiu o problema investigado, contextualizando o mesmo em relação ao estado da arte no gerenciamento de redes com QoS. A primeira contribuição desse trabalho foi apresentada no capítulo 3, que introduziu a proposta de um modelo para descoberta de QoS, composto por uma metodologia e uma arquitetura que define as entidades que implementam as funcionalidades da metodologia. Por fim, no capítulo 4, uma implementação baseada no modelo foi descrita. O presente capítulo irá apresentar conclusões sobre o modelo proposto e o protótipo implementado. Finalmente, esse capítulo encerra essa dissertação, apresentando um conjunto de possíveis trabalhos futuros.

5.1 Análise do modelo

Como mencionado anteriormente, o modelo de descoberta de QoS é composto por uma metodologia e uma arquitetura. Na metodologia são definidas as principais funcionalidades para a realização da descoberta de QoS, enquanto na arquitetura tais funcionalidades são atribuídas a entidades. A metodologia proposta, entretanto, pode ser implementada utilizando-se outras arquiteturas de gerenciamento, já que as funcionalidades definidas na metodologia não dependem de elementos específicos da arquitetura utilizada, e podendo ser implementados por elementos diferentes de arquiteturas distintas. Por um lado, tal distinção permite que a metodologia possa ser aplicada à outras arquiteturas. Por outro, a arquitetura pode agregar funcionalidades não definidas na metodologia e disponibilizar novos serviços não previstos anteriormente, possibilitando, assim, a evolução da proposta. Por exemplo: a metodologia poderia ser aplicada a uma arquitetura que utilize outros paradigmas de gerenciamento de rede, tal como o gerenciamento centralizado.

Alguns aspectos merecem destaque na metodologia. Primeiramente, sua divisão em diferentes procedimentos permite maior flexibilidade no modelo, tanto do ponto de vista da implementação quanto do ponto de vista do melhor aproveitamento das informações geradas pelos procedimentos intermediários. Além disso, futuras evoluções em um determinado procedimento, desde que mantidas as interfaces, não afetariam a definição dos outros procedimentos. Por exemplo: a execução dos dois primeiros procedimentos da metodologia (identificação parcial dos dispositivos e seleção parcial de recursos dos dispositivos) fornece para o gerente de rede uma lista de dispositivos, onde cada dispositivo está associado a possíveis recursos de QoS. Se os identificadores parciais apresentarem boa precisão e a base de conhecimentos estiver devidamente atualizada, essas informações podem ser utilizadas para uma busca rápida dos recursos de rede disponíveis. Tais características podem ser alcançadas com manutenção adequada do sistema de gerenciamento de QoS e da configuração dos dispositivos. A manutenção do sistema de gerenciamento de QoS diz respeito principalmente à base de

conhecimento: se ela estiver sempre atualizada, poderá fornecer uma lista mais precisa e completa dos possíveis recursos de QoS implementados pelo dispositivo. A manutenção da configuração dos dispositivos diz respeito à atualização das informações dos dispositivos relacionadas à geração do identificador parcial. Por exemplo: se uma atualização no *firmware* de um dispositivo for realizada para adição de funcionalidades, é importante que os identificadores parciais correspondentes a essas novas funcionalidades sejam disponibilizados, refletindo a atualização.

Se a execução dos dois primeiros procedimentos da metodologia já oferece informações úteis para o gerente de rede, o resultado da execução de todos os procedimentos pode ser utilizado por outras tarefas complementares do gerenciamento de rede, já que os outros procedimentos acabam apresentando informações ainda mais precisas e refinadas. Por exemplo: a descoberta de QoS pode ser utilizada para auxiliar no planejamento dos serviços de rede, onde a descrição dos alvos presentes nos dispositivos da rede permite ao gerente melhor definir quais serviços podem ser oferecidos.

Quando a metodologia foi elaborada, o maior desafio foi definir um mecanismo que diminuísse o tráfego de dados gerado pela descoberta de QoS. Assim, foi introduzido o conceito de identificadores parciais, uma informação que permite deduzir quais recursos de QoS o dispositivo pode possuir. Um aspecto fundamental nesse mecanismo é o baixo tráfego gerado na obtenção dos identificadores parciais, pois as consultas são realizadas através de trocas de mensagens bastante “leves” com os dispositivos.

Uma solução mais simplificada de identificação de dispositivos pode apresentar baixa eficiência. Por exemplo: ferramentas tradicionais que possuem facilidades para descoberta de características internas aos dispositivos acabam gerando muitas mensagens desnecessárias (normalmente, nesses casos se faz um *walk* completo na MIB do dispositivo, mas apenas algumas informações retornadas são realmente utilizadas). Nesse trabalho, entretanto, o uso de identificadores parciais reduz o número de consultas desnecessárias aos dispositivos, já que se parte de consultas mais específicas que tem uma maior chance de sucesso. Entretanto, caso sejam obtidos identificadores parciais inválidos isso irá gerar consultas ineficazes e, ao mesmo tempo, ineficientes, pois tal fato causa o desperdício de largura de banda disponível.

Embora o uso de identificadores parciais auxilie bastante na redução do tráfego da descoberta de QoS, um cenário onde o dispositivo de rede esteja muito longe do agente de descoberta nem sempre seria beneficiado: o tráfego de mensagens geradas pela descoberta de QoS poderia até ser pequeno, mas poderia passar por vários segmentos de rede, desperdiçando largura de banda. Assim, a arquitetura foi elaborada seguindo um paradigma distribuído, de maneira que os agentes de descoberta pudessem ficar mais próximos dos dispositivos de rede. Entretanto, essa abordagem aumenta a complexidade da distribuição das tarefas de descoberta de QoS, como será visto nas considerações sobre a implementação.

Outro aspecto considerado na elaboração da arquitetura foi a definição de um elemento que escolhesse, de maneira transparente para o gerente de rede, o agente de descoberta de QoS, baseado apenas nos parâmetros da solicitação do usuário. Esse elemento, que não existia na metodologia, foi chamado de seletor de agentes e permitiu que o gerente de rede abstraísse aspetos intrínsecos à distribuição de tarefas. Entretanto, tal solução implica num aumento de complexidade da implementação do seletor de

agentes, justificada pelo benefício propiciado. A complexidade está associada ao fato de que, para selecionar o agente de descoberta de QoS, o seletor de agentes deve conhecer todas as características de todos os agentes de descoberta, o que pode não ser uma tarefa muito simples em ambientes com recursos de memória e processamento limitados. Além disso, o seletor de agentes deve estar com as informações consistentes em relação ao estado atual dos agentes que compõem a arquitetura instalada, o que historicamente não é uma tarefa simples de se resolver, pois envolve a utilização de esquemas de comunicação distribuída para sincronização de estados. Em alguns casos não é desejável que o seletor de agentes escolha um agente de descoberta que não esteja ativo.

Além do seletor de agentes, outro elemento da arquitetura que merece destaque é o agente de descoberta de QoS, responsável por identificar os alvos. Um dos requisitos na descoberta de QoS é o de que a mesma seja extensível, de maneira que novos tipos de alvos possam ser identificados. Tal característica é implementada através da criação de novos agentes de descoberta: quando um novo tipo de alvo é solicitado a ser identificado, um novo agente de descoberta de QoS é implementado e então executado.

Como pode ser visto na análise do modelo, a metodologia e a arquitetura não se preocuparam com o mecanismo de distribuição das tarefas de descoberta de QoS propriamente dita. Tal aspecto é abordado na implementação e analisado na próxima seção.

5.2 Análise da implementação

A arquitetura da implementação foi organizada em módulos, cada um representando uma entidade autônoma de processamento. Um mecanismo implementado, que não é representado por um módulo específico, é o de distribuição de agentes de descoberta de QoS. O mecanismo de distribuição de agentes é responsável por ativar, monitorar e finalizar tarefas de descoberta de QoS executadas pelos agentes. Por possuir funcionalidades em praticamente todos os módulos da arquitetura, menos no módulo *Client*, tal mecanismo foi o que mais gerou código fonte, sendo, portanto, o mais extenso no desenvolvimento da implementação do protótipo. Esse mecanismo está presente em praticamente todos os módulos da arquitetura, pois a autonomia de execução foi privilegiada. Para garantir autonomia nesse mecanismo, o sub-módulo *Script-MIB Iface* foi replicado nos módulos *High Level Manager* e *Task Processor*. Além disso, o módulo *QoS Discovery Distributed Manager* precisa implementar as funcionalidades necessárias para trocar mensagens com o *Task Processor*, como, por exemplo, a mensagem de ativação de descoberta de QoS. Como não é objetivo desse trabalho desenvolver um novo mecanismo para delegar tarefas de descoberta de QoS, a solução adotada foi utilizar a Script-MIB, através da implementação do Jasmin. A Script-MIB apresenta várias vantagens na delegação de tarefas de gerenciamento, entre elas a possibilidade de aumentar as funcionalidades de um agente de descoberta sem parar a execução do nodo que suporta a execução do agente. É desejado que o módulo responsável pela monitoração dos *scripts* em execução, o *Task Processor*, seja informado de maneira assíncrona sobre o resultado da descoberta de QoS. Entretanto, a solução existente demanda consultas do tipo *polling* num objeto específico na Script-MIB. Um esquema baseado em *polling* não é interessante, pois pode inserir uma quantidade razoavelmente grande de mensagens. Diante desse problema, foi implementada uma solução própria, na qual o *script* de gerenciamento (na forma de um

agente de descoberta de QoS) envia uma *trap* SNMP ao seletor de agentes que ativou a descoberta de QoS, numa abordagem mais próxima de sinalização por eventos assíncronos.

A escolha do Jasmin não favoreceu apenas a implementação da delegação de tarefas de descoberta de QoS. As linguagens suportadas pelo Jasmin para a implementação dos *scripts* de gerenciamento possuem alta portabilidade. É possível executar *scripts* de gerenciamento em Java (a primeira linguagem suportada e que deu origem ao nome Jasmin) e TCL. Como seria necessário integrar vários *scripts* de gerenciamento, tanto os agentes de descoberta quanto os de propósitos gerais, tal como *scripts* PHP, Java deixou de ser uma alternativa, devido à dificuldade de integração. Com o uso de TCL, entretanto, o desenvolvimento de agentes de descoberta ficou mais simples devido à versatilidade da linguagem e à quantidade de funcionalidades relacionadas à descoberta de QoS oferecidas, representada pela extensão *Scotty* Tnm (*TCL Extensions for Network Management Applications*). Embora o TCL seja interpretado, todas as funções relacionadas à comunicação de rede são implementadas num módulo desenvolvido em C/C++.

Apesar do mecanismo de distribuição de tarefas ter sido a tarefa mais extensa durante o desenvolvimento do protótipo, a implementação dos agentes de descoberta foi a mais complexa, devido à diversidade dos agentes SNMP (versão da MIB e protocolo SNMP implementado, por exemplo) que podem ser encontrados numa rede. A maior dificuldade encontrada foi a obtenção dos identificadores parciais dos dispositivos. Seria interessante que os dispositivos disponibilizassem uma lista dos recursos implementados, o que facilitaria a obtenção dos identificadores parciais. O IETF define na MIB-II a tabela *sysORTable*, que lista as "capacidades" do agente SNMPv2 em relação às MIBs suportadas. Por exemplo: se um agente pudesse listar todos os mecanismos de fornecimento de QoS na *sysORTable*, tais informações poderiam ser interpretadas como identificadores parciais. Entretanto, vários dispositivos testados implementam apenas SNMPv1 (*switches* ATM), e outros utilizam soluções proprietárias que, apesar de implementarem a *sysORTable*, implementam objetos de MIB proprietárias que possuem descrições mais detalhadas sobre as "capacidades" implementadas pelo agente. No caso do fabricante CISCO foi desenvolvido a CISCO-SYSTEM-MIB [CIS 2001a], que apresenta informações mais específicas que o grupo *system* da MIB-II, que não são apresentadas na *sysORTable*.

5.3 Considerações finais

As redes de computadores com QoS tiveram um grande crescimento não só devido à demanda crescente de aplicações de rede que requerem qualidade de serviços em seus fluxos de dados, mas também ao aumento da diversidade de soluções para o fornecimento de QoS e de ferramentas e propostas para o gerenciamento dessas funcionalidades. Várias propostas de grupos de pesquisas e do próprio IETF surgiram para resolver os problemas de fornecimento e gerenciamento de QoS. Uma das tarefas identificadas no gerenciamento de QoS é a descoberta de QoS, que tem por objetivo identificar os alvos da rede. Devido à heterogeneidade e ao atual crescimento das redes de computadores, a proposta de modelo de descoberta de QoS apresentada nesse trabalho considerou que a tarefa deveria ser extensível, escalável e distribuída. Baseado nas análises realizadas sobre o modelo e a implementação, alguns aspectos mais gerais

foram observados.

Em primeiro lugar, o modelo se demonstrou extensível, pois a adição de novos agentes garantiu tal característica. Tradicionalmente, para se adicionar um novo módulo de software num agente SNMP é necessário parar a execução do sistema. Na Script-MIB isso não é necessário, devido ao próprio propósito dessa MIB, o que se demonstrou uma vantagem. O Jasmin, no início, se demonstrou um pouco instável para a adição de *scripts* de gerenciamento, quando era tentado a instalação de mais de 4 agentes ao mesmo tempo. Após algumas consultas aos desenvolvedores do aplicativo, algumas mudanças na configuração foram realizadas e o problema foi resolvido. Outro detalhe interessante sobre a Script-MIB é que ela resolve o problema de delegação dos *scripts* de gerenciamento, e não a distribuição, ou seja, a decisão de como os *scripts* de gerenciamento irão ser distribuídos é responsabilidade do aplicativo de gerenciamento. Nessa implementação ou a distribuição era realizada pelo gerente de rede, no modo manual, ou automaticamente, pelo assistente de distribuição de tarefas.

5.4 Trabalhos futuros

Esse trabalho não teve a pretensão de esgotar todos os temas relacionados à descoberta de QoS. Naturalmente, foram identificados alguns trabalhos futuros a partir das conclusões realizadas. Em primeiro lugar, o desenvolvimento de novos agentes de descoberta poderia aumentar o número de tipos de alvos que a ferramenta identifica. Além disso, seria interessante que os agentes relatassem todos os identificadores parciais que não geraram uma lista de possíveis recursos de fornecimento de QoS implementado pelo dispositivo, de maneira que a base de conhecimento fosse devidamente atualizada.

Como foi relatado, o agente de descoberta envia uma *trap* SNMP para informar o *Task Processor* sobre o fim da execução da tarefa de descoberta de QoS. Embora essa solução não seja padronizada, foi a maneira encontrada para resolver o problema. Além disso, a implementação do modelo prevê que numa futura abordagem sejam utilizados esquemas padronizados. O grupo DISMAN do IETF já definiu a Event-MIB, um *superset* dos alarmes da RMON-MIB, onde há a vantagem de a sinalização do evento ser realizada numa entidade remota, enquanto a RMON-MIB apenas informa eventos no *host* local. Seria possível configurar um *trigger* que fosse acionado ao término da execução do *script* de gerenciamento.

Um outro trabalho interessante seria pesquisar o desempenho do uso do TCL como linguagem dos *scripts* de gerenciamento. A literatura descreve algumas técnicas que embutem o interpretador TCL em códigos C/C++, ou através da utilização de *profiling* de execução do TCL. Uma funcionalidade ainda não estável na implementação do *engine* TCL é uma abordagem *multithreading*, que poderia melhorar o desempenho de procedimentos que hoje têm sua execução serializada, pois foi utilizada a versão não *multithread* do interpretador.

Por fim, uma evolução nas definições da metodologia e da arquitetura parece ser um caminho natural para abordar aspectos não considerados no estado atual do modelo, tal como o mecanismo de delegação de tarefas de descoberta de QoS. Assim, o autor espera com esse trabalho tenha contribuído para o avanço do estado da arte na área de gerenciamento de QoS.

Anexo 1 Instalação do ambiente NET-SNMP e Jasmin

Esse anexo tem por objetivo traçar um roteiro de instalação das plataformas Jasmin/NET-SNMP com o intuito de que as experiências realizadas nesse trabalho possam ser reproduzidas. Cabe ressaltar que esse breve manual de instalação não procura de forma alguma ser um "manual de instalação do Jasmin e do NET-SNMP". Serão detalhados os passos de instalação e apenas os parâmetros de configuração utilizados para esse trabalho, desconsiderando as várias possibilidades diferentes de configuração destes softwares.

Assim, os passos a seguir descrevem a instalação do ambiente NET-SNMP\Jasmin numa estação Conectiva Linux kernel 2.2.14, considerando alguns aspectos particulares desse trabalho.

1. Obtendo os pacotes

Esse trabalho utilizou a versão 4.2 do NET-SNMP, 0.9.6 do Jasmin, 1.5.3 do wget, que podem ser obtidos de ftp://ftp.ibr.cs.tu-bs.de/pub/local/jasmin/. A versão do Apache foi a 1.3.X (http://www.apache.org), do Java 1.1.8 e 1.2 (http://www.blackdown.org) e do TCL e TK a 8.3 (http://www.tcl.tk) e o Scotty Tnm. O Apache e o Java tiveram sua instalação facilitada, pois foram utilizados os respectivos pacotes RPM. Entretanto, os outros pacotes tiveram de ser compilados a partir do código fonte.

2. Descompactação e compilação

Como usuário root descompacte os pacotes do NET-SNMP, Jasmin, wget, TCL e o TK em diretórios diferentes, mas no mesmo nível da árvore de diretórios. Assim, siga os seguintes passos para a compilação dos pacotes:

TCL/Tk

No diretório "./unix", localizado no diretório raiz dos arquivos fontes do TCL, execute os seguintes comandos:

- *./configure --enable-shared (o ./ é importante!)*
- *make*
- *make install*

Então , adicione ao arquivo /etc/profile a seguinte variável de ambiente:

TCL_LIBRARY="/usr/local/lib/tcl8.3"

Enfim, crie um *soft link* para o interpretador TCL ficar disponível ao PATH com o seguinte comando (só terá efeito se a variável de ambiente PATH contém "/usr/local/bin"):

ln -s /usr/local/bin/tclsh8.3 /usr/local/bin/tclsh

Para compilar o Tk, os passos são semelhantes. No diretório "./unix", localizado no diretório raiz dos arquivos de instalação do Tk, execute os seguintes comandos:

- *./configure --enable-shared --with-tcl=../tcl8.3.2/unix*
- *make*
- *make install*

Então, adicione ao arquivo /etc/profile a seguinte variável de ambiente:

TK_LIBRARY="/usr/local/lib/tk8.3"

Enfim, crie um *soft link* para o interpretador Tk ficar disponível ao PATH com o seguinte comando (só terá efeito se a variável de ambiente PATH contém "/usr/local/bin"):

ln -s /usr/local/bin/wish8.3 /usr/local/bin/wish

Scotty Tnm

No diretório ".\unix" localizado no diretório raiz do Scotty, execute as seguintes linhas de comando:

- *./configure --with-tcl=/usr/local/lib/ --with-tk=/usr/local/lib/*
- *make*
- *make install*

Siga as últimas instruções apresentadas pela instalação do scotty.

Anexo 2 Interface gráfica do protótipo

Esse anexo tem por objetivo apresentar a interface gráfica do protótipo. Como a interface foi baseada em páginas da Web, utilizando tecnologia PHP, serão apresentadas imagens de um navegador Web apresentando as páginas do protótipo.

Página Inicial

A página inicial, ilustrada na Fig. 1, apresenta rapidamente o protótipo e possui *links* HTTP para todos os serviços implementados: solicitação de descoberta de QoS, estado de um processo de descoberta de QoS, relatório dos processos de descoberta de QoS já realizados e tarefas administrativas. Como o protótipo está inserido no ambiente QAME, à esquerda da página está presente o menu desse ambiente.

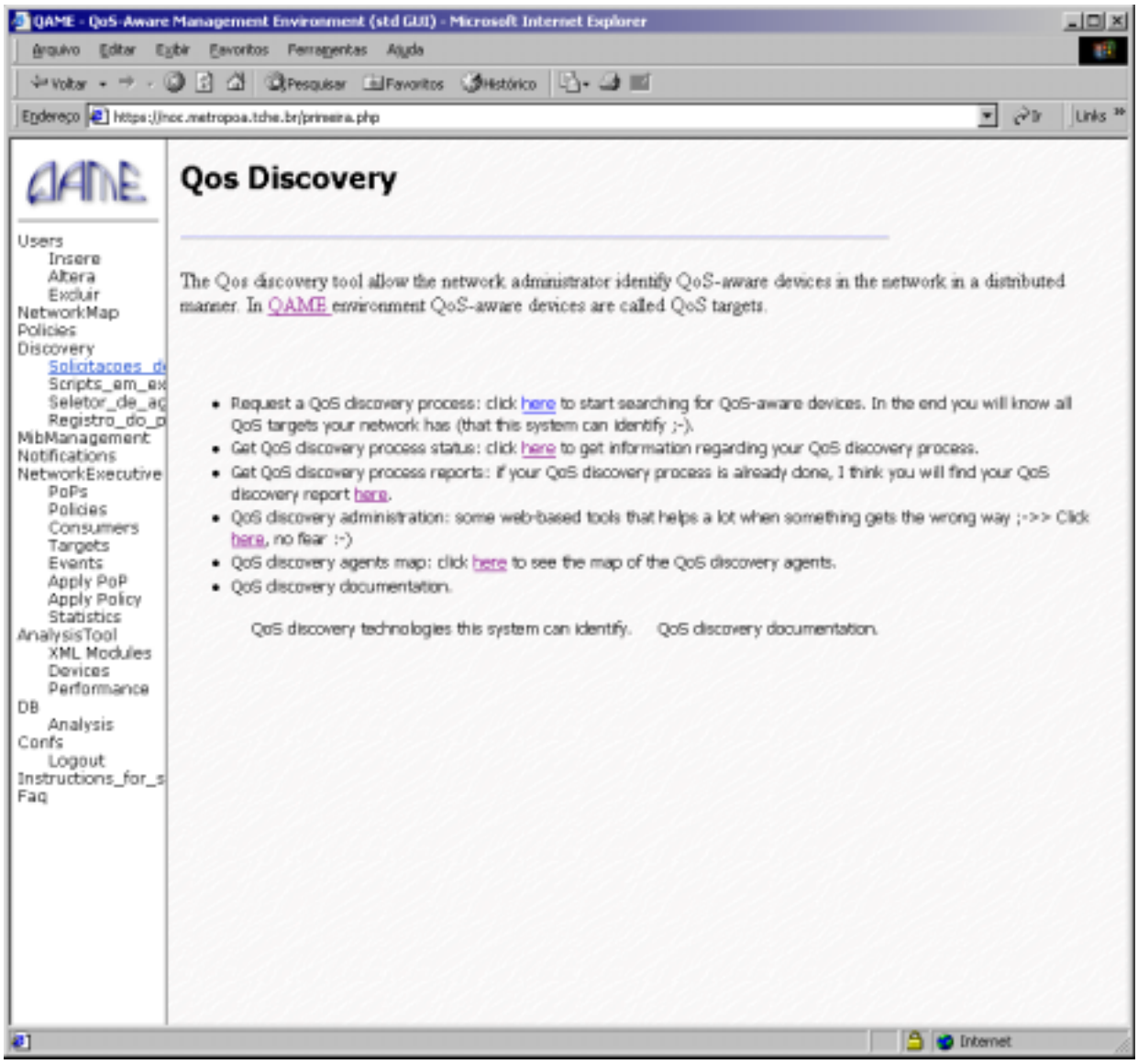

Fig. 1: Página inicial do protótipo.

Página de solicitação de descoberta de QoS

A página de solicitação de descoberta de QoS apresenta uma interface de parametrização dos procedimentos da metdologia de descoberta de QoS. As figuras Fig. 2 e Fig. 3 mostram a página de solicitação de descoberta de QoS.

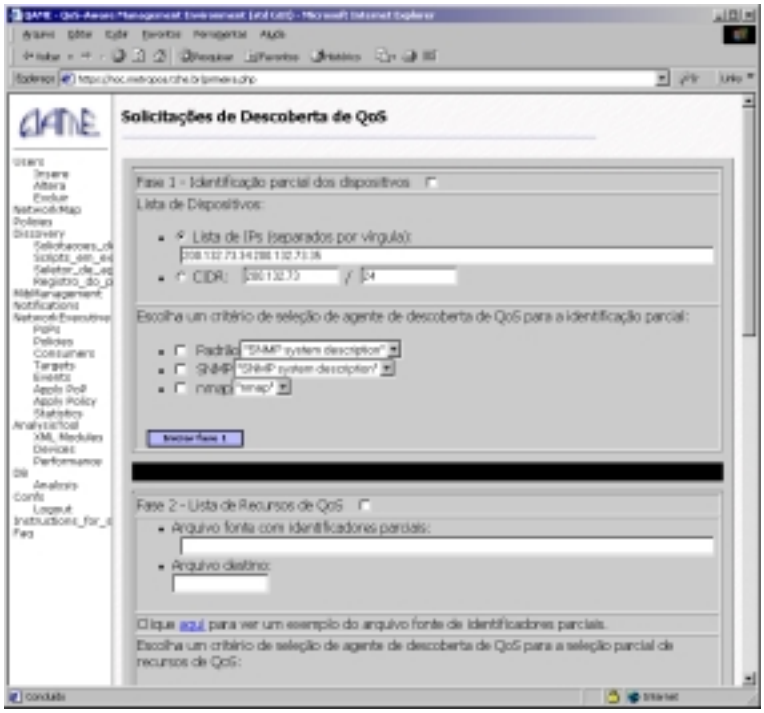

Fig. 2: Página inicial do protótipo.

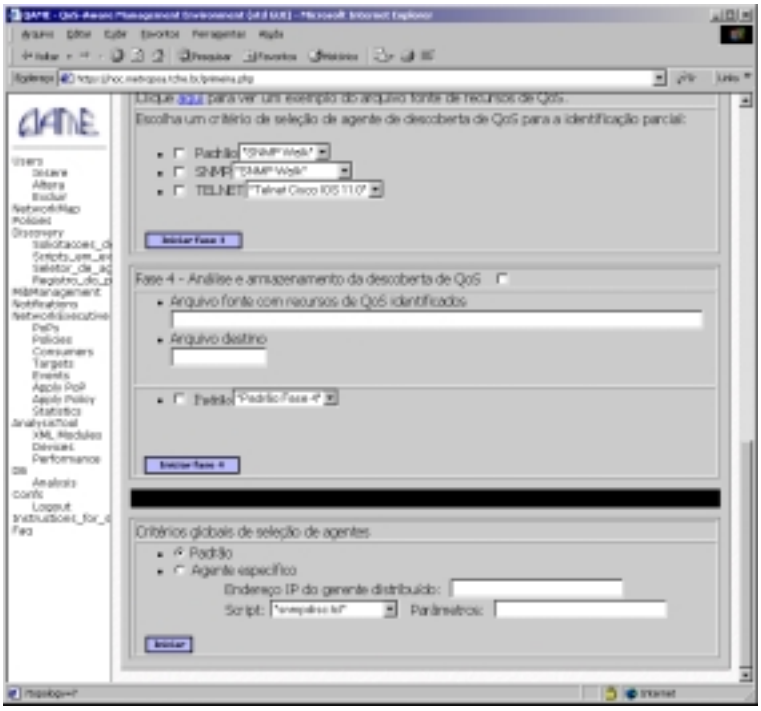

Fig. 3: Página inicial do protótipo.

Como pode ser visto, é possível executar cada procedimento da metodologia individualmente, pois há um botão que inicia a execução de cada procedimento.

Página de verificação de estado de um processo de descoberta de QoS

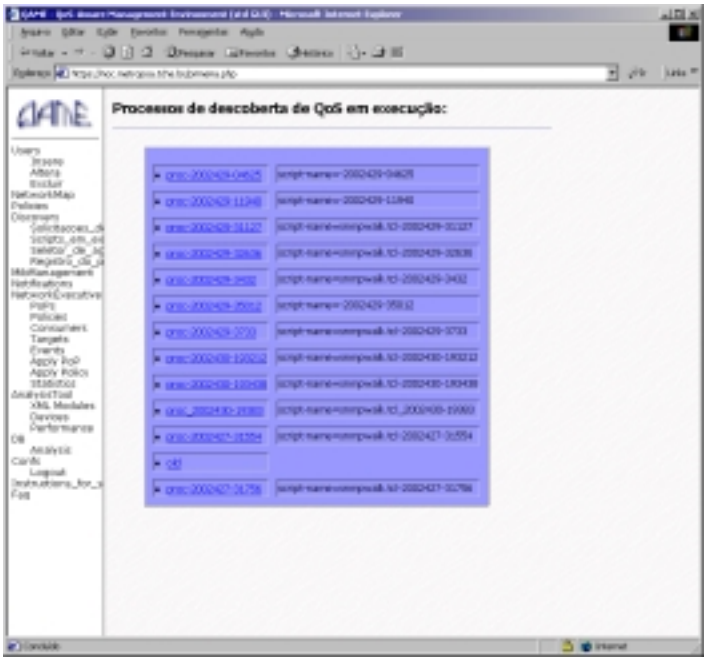

Fig. 4: Página de listagem dos processos de descoberta de QoS em execução.

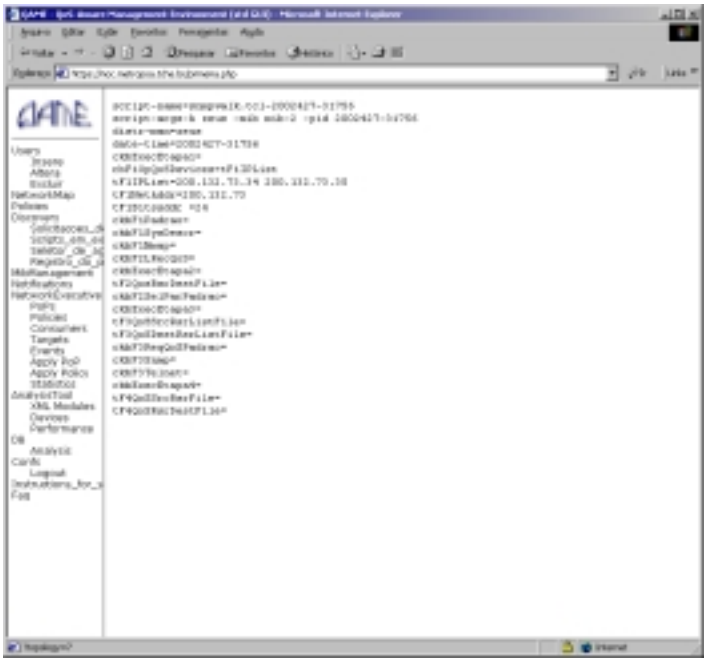

Fig. 5: Página de detalhamento do estado de execução de um processo de descoberta de QoS.

As figuras Fig .4 e Fig. 5 apresentam as páginas que permitem visualizar o estado de execução dos processos de descoberta de QoS. Na Fig. 4 é apresentada a lista dos processos de descoberta de QoS, enquanto que na Fig. 5 são apresentados os atributos do processo de descoberta de QoS. Note que a apresentação dos atributos de

um processo de descoberta de QoS é textual.

Páginas de administração

As páginas de administração da ferramenta fornecem ferramentas para manipular manualmente os *scripts* de descoberta de QoS.

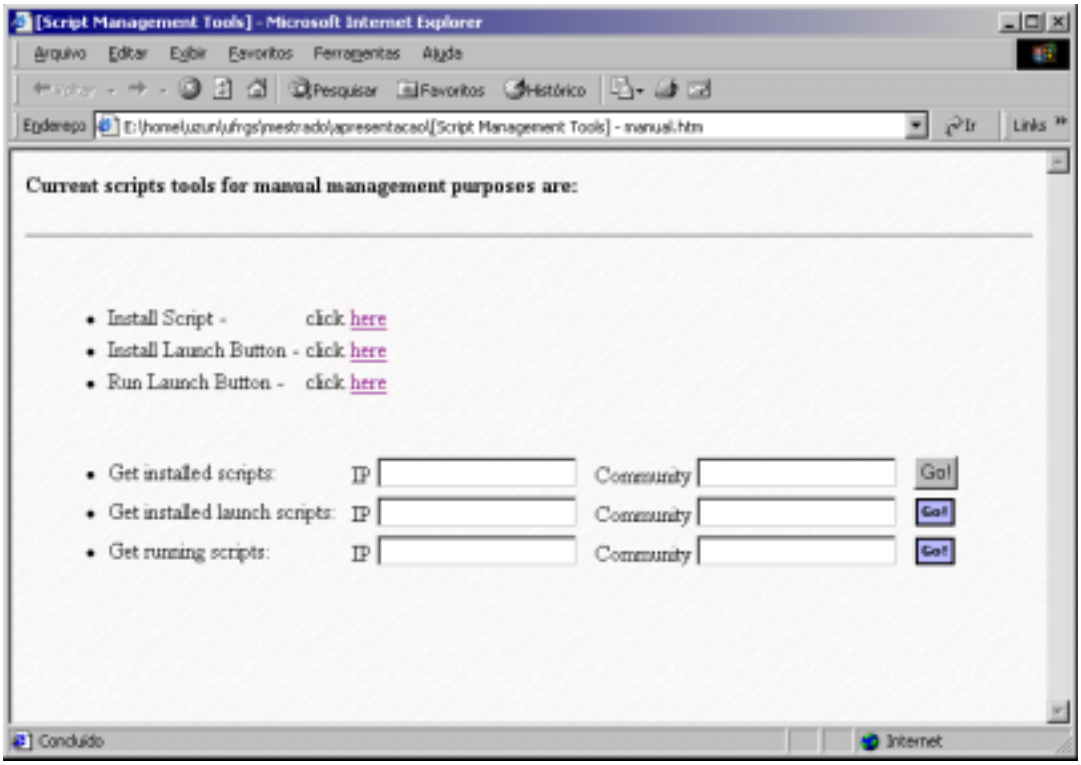

Fig. 6: Página inicial de administração

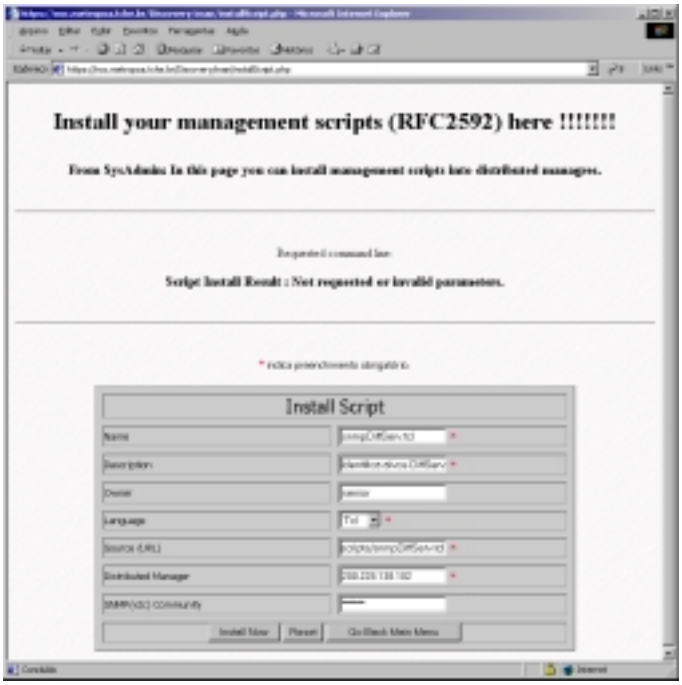

Fig. 7: Página de instalação de um script de descoberta de QoS

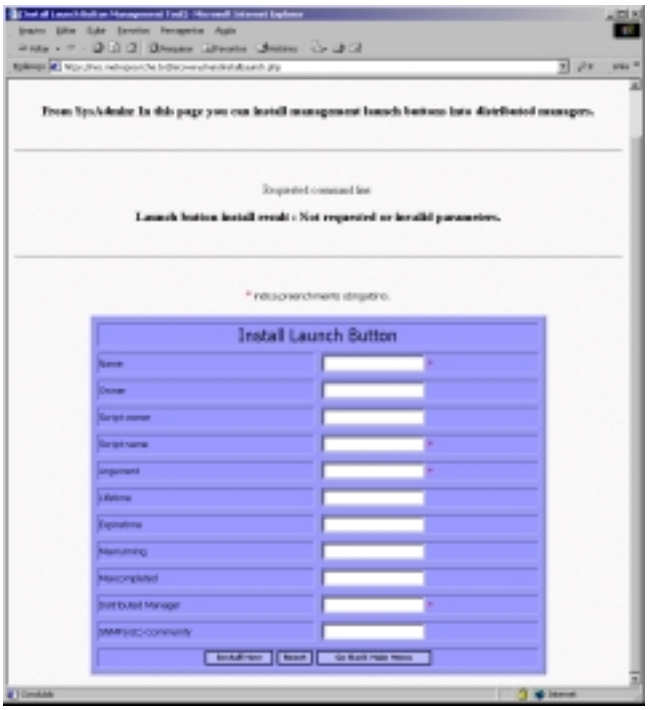

Fig. 8: Página de instalação de um *launch button*.

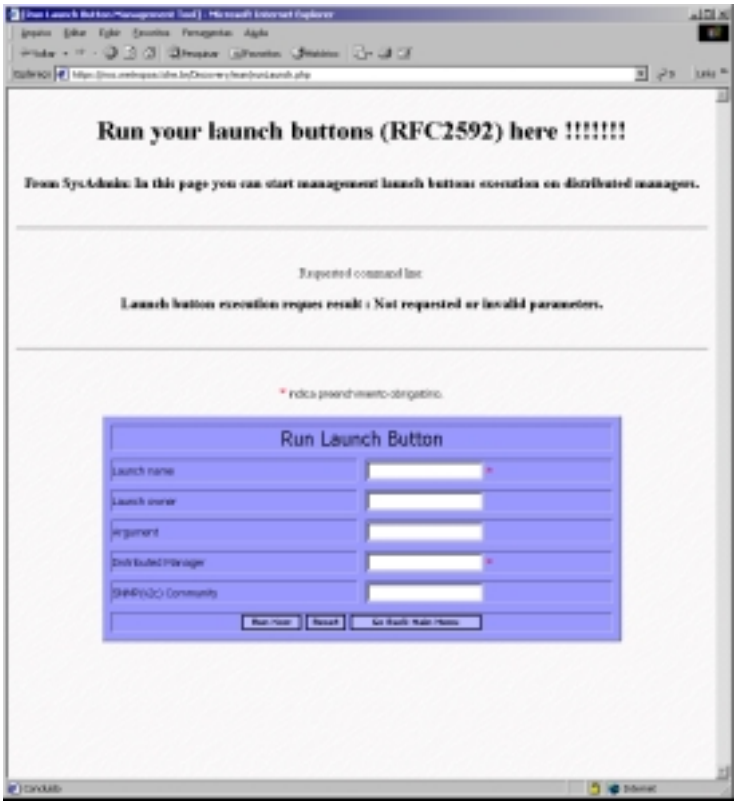

Fig. 9: Página de solicitação de execução de um *launch button*.

A figura Fig. 6 apresenta a página inicial de administração, onde é possível ter acesso a todas ferramentas através de *links* HTTP. A Fig. 7 apresenta a página que possui a interface de instalação de um *script* de descoberta de QoS. A Fig. 8 apresenta a página que possui a inteface de instalação de um *launch button* de um *script* de

descoberta de QoS. O *launch button* pode ser considerado como uma instância parametrizada de um *script* de descoberta de QoS, e está ligado a características da plataforma de delegação Jasmin. Para maiores informações sobre a plataforma de delegação Jasmin veja a documentação relacionada à plataforma. Por fim, a Fig 9 apresenta a página que possui a interface de solicitação de execução de um *launch button* de descoberta de QoS.

Exemplo de resultado do processo de descoberta de QoS

A Fig. 10 apresenta o exemplo do resultado de execução do primeiro e segundo procedimentos da descoberta de QoS. Como pode ser visto, o resultado é textual e orientado a atributo e resposta, agrupados em seções de interesse dentro do arquivo. Além disso, todos os resultados sempre apresentam na primeira seção os atributos de descrição do processo de descoberta, onde destacamos a presença de informações como parâmetros de execução do *script* (atributo PARAM) e o identificador da tarefa (atributo PID).

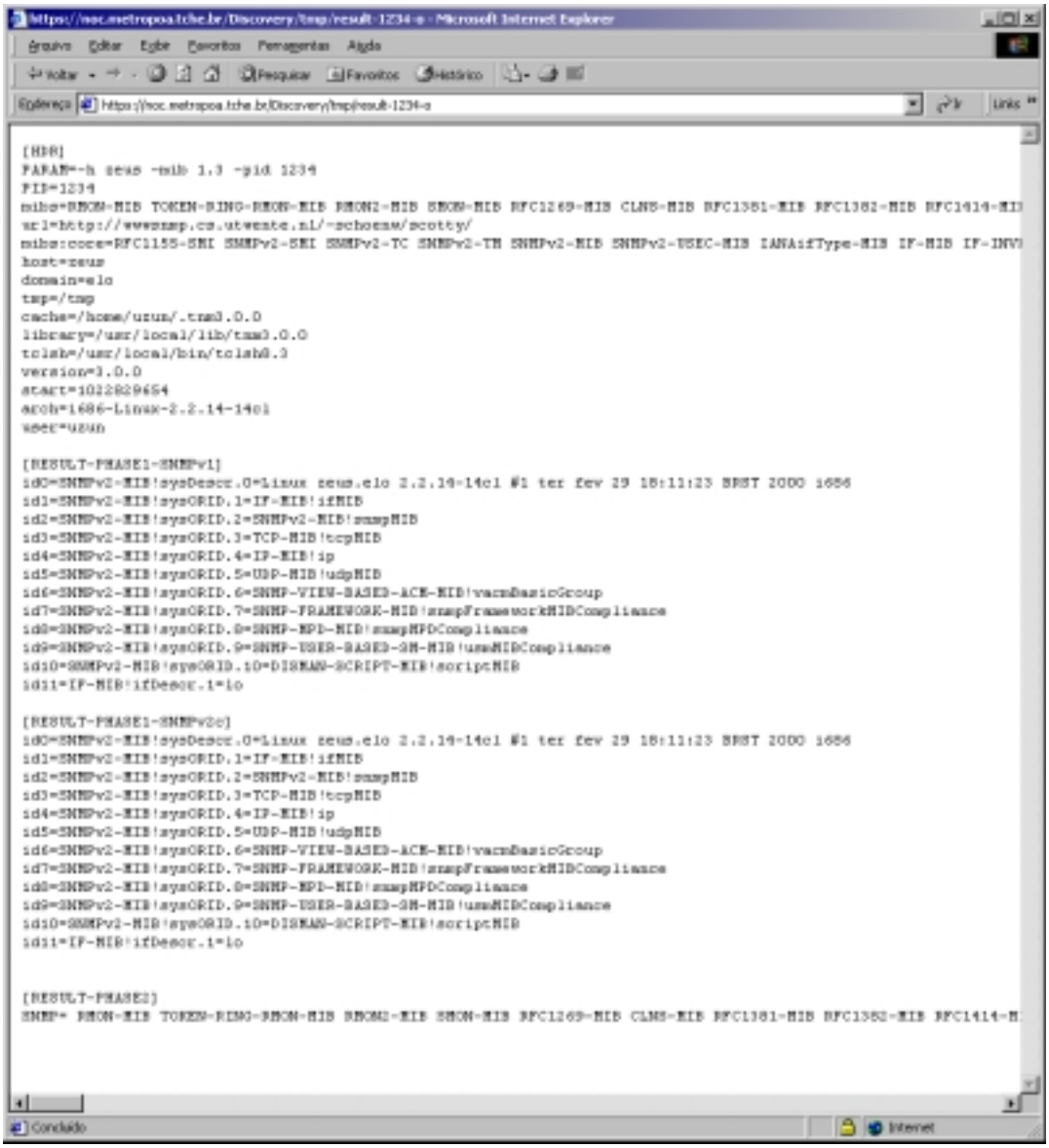

Fig. 10: Exemplo do resultado da execução do primeiro e segundo procedimentos.

Anexo 3 Implementado um agente de descoberta de QoS como um *script* de gerenciamento da Scrip-MIB

Esse anexo tem por objetivo fornecer as informações necessárias para o desenvolvimento de um agente de descoberta de QoS como um *script* de gerenciamento da Scrip-MIB para ser executado dentro do ambiente do protótipo. Tais informações são necessárias, pois a API de comunicação utilizada entre os *scripts* PHPs (que implementam a aplicação de gerenciamento da descoberta de QoS) e os *scripts* TCL (que implementam a descoberta de QoS) deve ser conhecida. A seguir serão apresentadas as figuras com o código fonte TCL mínimo que um agente de descoberta de QoS deve implementar.

```
##
## qos_discovery_sample.tcl - sample.
##
package require Tnm 3.0
##
## script variables
##
set ag_sc "1"; ## agente script jasmin
set ag_sh "2";  ## agente para shell
set ag_t $ag_sc; ## tipo do agente
## Script parameters
set host ""; ## host IP list
set net ""; ## ip net address
set mask ""; ## ip netmask
set g_pid ""; ## process identifier
set task_processor ""; ## task processor IP do send script status message.
set phase ""; ## phase do be executed
set ph1_param ""; ## phase 1 parameters
set ph2_param ""; ## phase 2 parameters
set ph3_param ""; ## phase 3 parameters
set ph4_param ""; ## phase 4 parameters
## script scpecific variables
set label "";
set rootDir "/usr/local/apache/htdocs/";
set webSubDir "./";
```

Fig. 1: Declaração das variáveis globais.

A Fig.1 apresenta a declaração das variáveis globais do *script* de descoberta de QoS, das quais destacamos as que armazenam os parâmetros enviados pelo cliente. A variável *label* foi declarada apenas para deixar claro que o *script* pode ter parâmetros específicos. Além disso, é sugerido que os parâmetros de cada procedimento sejam armazenados em variáveis diferentes, para facilitar sua manipulação no código. No início do código uma diretiva TCL é declarada, para que seja utilizado a extensão Tnm do TCL. Como a implementação foi baseada principalmente em consultas SNMP dos dispositivos, o uso da biblioteca TNM é essencial para a implementação dos agentes de descoberta.

```
## Send usage Message to client
proc usage {} {
        ::Tnm::smx result "Error: invalid script argument";
        ::Tnm::smx exit  invalidArgument;
}
## Parse user parameters
proc parseCmd {} {
        global host label g_pid phase argv;
        set parsing_options 1;
        while {([llength $argv] > 0) && $parsing_options} {
                set arg [lindex $argv 0];
                set argv [lrange $argv 1 end];
                if {[string index $arg 0] == "-"} {
                        switch -- $arg  {
                                "-h"    { ## host
                                        set host [lindex $argv 0];
                                        set argv  [lrange $argv 1 end]; }
                                "-net"    { ## ip net address
                                        set net [lindex $argv 0];
                                        set argv  [lrange $argv 1 end];  }
                                "-mask"    { ## ip netmask
                                        set mask [lindex $argv 0];
                                        set argv  [lrange $argv 1 end]; }
                                "-pid"  { ## process identifier
                                        set g_pid [lindex $argv 0];
                                        set argv    [lrange $argv 1 end]; }
                                "-tp"       { ## task processor IP do send script status message.
                                        set task_processor [lindex $argv 0];
                                        set argv    [lrange $argv 1 end]; }
                                "-phase"    { ## phase do be executed
                                        set phase [lindex $argv 0];
                                        set argv    [lrange $argv 1 end]; }
                                "-pph1"     { ## phase 1 parameters
                                        set ph1_param [lindex $argv 0];
                                        set argv    [lrange $argv 1 end]; }
                                "-pph2"     { ## phase 2 parameters
                                        set ph2_param [lindex $argv 0];
                                        set argv    [lrange $argv 1 end]; }
                                "-pph3"     { ## phase 3 parameters
                                        set ph3_param [lindex $argv 0];
                                        set argv    [lrange $argv 1 end]; }
                                "-pph4"     { ## phase 4 parameters
                                        set ph4_param [lindex $argv 0];
                                        set argv    [lrange $argv 1 end];
                                }
                                "--"    { set parsing_options 0;  }
                        }
                } else {
                        set parsing_options 0;
                        lappend newargv $arg;
                }
        }
}
```

Fig. 2: Funções de mensagem e análise dos parâmetros fornecidos pelo usuário.

A Fig.2 apresenta duas funções: a primeira (*usage*) envia uma mensagem de uso dos parâmetros do *script*, enquanto a segunda (*parseCmd*) analisa os parâmetros de entrada.

```
# Get local IP
proc getLocIP {} {
        set locIPs [Tnm::netdb hosts];
        foreach item $locIPs {
                foreach {name ip} $item {
                        if {$ip != "127.0.0.1"} return $ip;
                }
        }
        return "127.0.0.1";
}
## start execution
set argv_bk $argv;
parseCmd;
if {($host == "") || ($label == "") || ($g_pid == "")} {
        usage;
} else {
        if [catch {::Tnm::mib oid $label} label] {
                usage;
        }
   if [catch {set rtn [start $host $net $mask $label $phase $ph1 $ph2 $ph3 $ph4 ]} rtn] {
                ::Tnm::smx result "Script failed in start command.";
                ::Tnm::smx exit  runtimeError;
        }
        set len [llength $rtn];
        if {$len < 1} {
                ::Tnm::smx result "Script failed.";
                ::Tnm::smx exit  genericError;
        } else {
                set ph1 [string match "*1*" $phase]
                set ph2 [string match "*2*" $phase]
                set ph3 [string match "*3*" $phase]
                set ph4 [string match "*4*" $phase]
                ## Setting global results for storage - RUF
                set g_rtn $rtn;
                set g_arg $argv_bk;
                set fName "discovery/tmp/result-$g_pid"
                set g_resultFile $rootDir$fName;
                createFile $g_resultFile;
                saveHdr $g_arg $g_pid $g_resultFile;
                if {$ph1 == "1"} {  saveResultPh1 $g_rtn $g_resultFile phase1_rtn; }
                if {$ph2 == "1"} { saveResultPh2 phase1_rtn $g_resultFile phase2_rtn; }
                if {$ph3 == "1"} { saveResultPh3 $g_rtn $g_resultFile; }
                if {$ph4 == "1"} { saveResultPh4 $g_rtn $g_resultFile; }
        }
}
sendFinalizationMessage $task_processor;
set locIP [getLocIP];
::Tnm::smx result "http://$locIP/$webSubDir$fName";
::Tnm::smx exit;
```

Fig. 3: Função auxiliar e execução principal do script de descoberta de QoS.

A Fig. 3 apresenta uma função *getLocIP* e o corpo do código fonte. A função *getLocIP* retorna o endereço IP da estação onde o *script* de descoberta está sendo executado e é utilizado para gerar a mensagem de término de execução do *script*, enviada pelo protocolo SMX. Alguns aspectos são interessantes destacar no corpo do código fonte.

A primeira tarefa do *script* é analisar os parâmetros de entrada, para então chamar a função *start*, responsável por executar os procedimentos de descoberta de QoS. Logo, todos os parâmetros relacionados à descoberta de QoS são passados como argumento dessa função. A seguir, o resultado da função é avaliado e a resposta é armazenada, havendo uma função para cada procedimento (funções *saveResultPh1*, *saveResultPh2*. *saveResultPh3* e *saveResultPh4*). Se for necessário, outras informações podem ser armazenadas, ficando a cargo do desenvolvedor do *script* criar novas funções para realizar tal tarefa. Por fim, o *script* deve enviar uma mensagem para o *Task Processor*, informando o término da tarefa, para que esse possa processar o resultado. Além disso, é necessário informar o agente SNMP, através da sessão SMX, o resultado da execução do *script*. A mensagem direcionada ao agente SMX está relacionada ao armazenamento dos estados internos da Script-MIB.

Bibliografia

[3CO 2001] 3COM Homepage. Disponível em: <http://www.3com.com>. Acesso em: out. 2001.

[3CO 2001a] 3COM switches Homepage. Disponível em: <http://www.3com.com/switches.html>. Acesso em: out. 2001.

[AIE 2000] AIELLO, W. A. et al. Competitive Queue Policies for Differentiated Services. In: IEEE INFOCOM, 2000. **Proceedings** … Tel-Aviv: IEEE Inc., 2000.

[AHM 94] AHMED, M.; TESINK, K. **Definitions of Management Objects for ATM Management Version 8.0 using SMIv2**. [S.l.]: IETF, Aug. 1994. (Request for Comments 1695). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[APA 99] THE APACHE SOFTWARE FOUNDATION. **The Apache Sotfware HomePage**. 1999. Disponível em: < http://www.apache.org/>. Acesso em: 2000.

[ASH 2001] ASH, G. **Traffic Engineering & QoS Methods for IP-, ATM-, & TDM-Based Multiservice Networks**. [S.l.]: IETF, Oct. 2001. Work in Progress. Disponível em: <http://www.ietf.org>. Acesso: 2001.

[AUR 97] AURRECOECHEA, C.; CAMPBELL, A.; HAUW, L. A survey of QoS architectures. **ACM/Springer Multimedia Systems Journal**, [S.l.], v.6, n.3, p. 138-151, May 1997. Special Issue on QoS Architectures.

[BAE 2002] BAER, Michael et al. **NET-SNMP Project Homepage**. Disponível em: < http://net-snmp.sourceforge.net/>. Acesso em: 2002.

[BLA 98] BLAKE, S. et al. **An Architecture for Differentiated Services**. [S.l.]: IETF, Dec. 1998. (Request for Comments 2475). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[BRA 94] BRADEN, R. et al. **Integrated Services in the Internet Architecture:** an Overview. [S.l.]: IETF, June 1994. (Request for Comments 1633). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[BRA 97] BRADEN, R. et al. **Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification**. [S.l.]: IETF, Sept. 1997. (Request for Comments 2205). Disponível em: <http://www.ietf.org>. Acesso em: 6 jan. 2000.

[BRE 2000] BREITBART, Y. et al. A Topology Discovery in Heterogeneous IP Networks. In: IEEE INFOCOM, 2000. **Proceedings** … Tel-Aviv: IEEE Inc., 2000.

[CAI 2001] CAIDA Homepage. Disponível em: <http://www.caida.org/>. Acesso em: 2001.

[CAI 2001a] CAIMIS Inc Homepage. Disponível em: <http://www.caimis.com/>.

Acesso em: 2001.

[CIS 2001] CISCO 12000 Series Internet Routers. Disponível em: <http:// www.cisco.com/warp/public/cc/pd/rt/12000/ >. Acesso em out. 2001.

[CIS 2001a] CISCO Homepage. Disponível em: <http://www.cisco.com>. Acesso em: 2001.

[CIS 2001b] CISCO QPM Homepage. Disponível em: <http://www.cisco.com/warp/public/cc/pd/wr2k/qoppmn/index.shtml>. Acesso em: 2001.

[CIS 2001c] CISCO Service-Level Manager Homepage. Disponível em: <http://www.cisco.com/warp/public/cc/pd/wr2k/svmnv1/>. Acesso em: 2001.

[CIS 2001d] CISCO Internetwork Performance Monitor Homepage**.** Disponível em: <http://www.cisco.com/warp/public/cc/pd/wr2k/nemo/>. Acesso em: 2001.

[CIS 2001e] CISCO QoS Device Manager Homepage. Disponível em: <http://www.cisco.com/warp/public/cc/pd/nemnsw/qodvmn/index.shtml >. Acesso em: 2001.

[CLA 88] CLARK, D. The Design Philosofy of the DARPA Internet Protocol. In: ACM SIGCOMM, 1988. **Proceedings…** [S.l.:s.n.], 1988.

[CRA 98] CRAWLEY, E. et al. **A Framework for QoS-based Routing in the Internet**. [S.l.]: IETF, Sept. 1998. (Request for Comments 2386). Disponível em: <http://www.ietf.org>. Acesso em: 2001.

[DIS 2001] DISTRIBUTED Management (disman) Charter Homepage. Disponível em: <http://www.ietf.org/html.charters/disman-charter.html>. Acesso em: 2002.

[DOV 99] DOVROLIS, C.; RAMANATHAN, P. A Case for Relative Differentiated Services and the Proportional Differentiation Model. **IEEE Network**, New York, v13, n.5, Sept./Oct. 1999.

[DRA 2000] DRACINSCHI, A.; FDIDA, S. Congestion Avoidance Mechanism for Unicast and Multicast. In: EUROPEAN CONFERENCE ON UNIVERSAL MULTISERVICE NETWORKS, ECUMN, 2000. **Proceedings…** Colmar: IEEE Inc., 2000.

[EDE 2001] EDER, M.; NAG, S. **Service Management Architectures Issues and Review**. [S.l.]: IETF, Jan. 2001. (Request for Comments 3052). Disponível em: <http://www.ietf.org>. Acesso em: 19 fev. 2001.

[EXT 2000] EXTENSIBLE Markup Language (XML) 1.0 (Second Edition). World Wide Web Consortium, 2000. Disponível em: <http://www.w3.org/TR/2000/REC-xml-20001006>. Acesso em: 2001.

[FIE 99] FIELDING, R. et al. **Hypertext Transfer Protocol -- HTTP/1.1**. [S.l.]: IETF, June 1999. (Request for Comments 2616). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[FOS 98] FOSTER, I.; KARONIS, N. A Grid-Enabled MPI: Message Passing in

Heterogeneous Distributed Computing Systems. In: ACM SC. **Proceedings…** Orlando: ACM Press, 1998.

[FSF 99] THE FREE SOFTWARE FOUNDATION. **The GNU Project**. 1999. Disponível em: <http://www.gnu.org/>. Acesso em: 2000.

[GON 99] GONÇALVES, A. R.; ROCHOL, J. Metodologia para Avaliação de Desempenho de Backbone ATM. In: SEMANA ACADÊMICA DO PPGC, 4., 1999, Porto Alegre. **Anais...** Porto Alegre: PPGC da UFRGS, 1999. p. 111-114.

[GRA 2000] GRANVILLE, L. Z.; FLEISCHMANN, R. U; TAROUCO, L; ALMEIDA, M. J. B. Managent of Differentiated Services through QoS-Enabled Hosts. In: ASIAN-PACIFIC NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, APNOMS, 2000, Nara-Japan. **Proceedings…** [S.l.:s.n.], 2000.

[GRA 2001] GRANVILLE, L. Z.; TAROUCO, L. QAME - An Environment to Support QoS Management Related Tasks on IP Networks. In: INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, ICT, 2001, Bucharest-Romania. **Proceedings…** [S.l.:s.n.], 2001.

[GRA 2001a] GRANVILLE, L. Z. et al. An Approach for Integrated Management of Networks with Quality of Service Support Using QAME. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS: OPERATIONS & MANAGEMENT, DSOM, 12., 2001 Nancy-France. **Proceedings…** [S.l.]: INRIA, 2001.

[GRA 2001b] GRANVILLE, L. Z.; ZANIN, F. A.; ALMEIDA, M. J. B. ADAPT- A Low-Cost Videoconference Model for Personal Computers Running on IP Networks. In. SYMPOSIUM ON APPLIED COMPUTING, SAC 2001, 16., 2001, Las Vegas. **Proceedings…** New York: ACM, 2001.

[HUS 2000] HUSTON, G. **Next Steps for the IP QoS Architecture**. [S.l.]: IETF Nov. 2000. (Request for Comments 2990). Disponível em: <http://www.ietf.org>. Acesso em: 27 fev. 2001.

[HUT 94] HUTCHISON, D. et al. Quality of Service Management in Distributed Systems. In: **Network and Distributed Systems Management**. Wokingham: Addison Wesley, 1994.

[IBM 2001] IBM Homepage. Disponível em: <http://www.ibm.com>. Acesso em: 2001.

[INT 2001] INTERNET ENGINEERING TASK FORCE. **IETF Homepage**. Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[JAI 92] JAIN, R. Myths about Congestion Management in High Speed Networks. **Internetworking: Res. and Exp.**, [S.l.], v.3, p. 101-113, 1992.

[JUN 2001] JUNIPER M160 Series. Disponível em: <http://www.juniper.net/products/m160-l2.html>. Acesso em: out. 2001.

[KAN 86] KANTOR, B.; LAPSLEY,P. **Network News Transfer Protocol:** A Proposed Standard for the Stream-Based Transmission of News. [S.l.]: IETF Feb. 1986. (Request for Comments 977). Disponível em:

<http://www.ietf.org>. Acesso em: 2001.

[KAV 2000] KAVASSERI, R.; STEWART, B. **Event MIB**. [S.l.]: IETF, Oct. 2000. (Request for Comments 2981). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[KAV 2000a] KAVASSERI, R.; STEWART, B. **Distributed Managemet Expression MIB**. [S.l.]: IETF, Oct. 2000. (Request for Comments 2981). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[LEO 2000] LEONEL, N. A. **Teoria dos grafos & análise combinatória**. [S.l.:s.n.], 2000. 48 f.

[LEV 2001] LEVI, D.; SHCOENWAELDER, J. **Definitions of Managed Objects for the Delegation of Management Scripts**. [S.l.]: IETF, Aug. 2001. (Request for Comments 3165). Disponível em: <http://www.ietf.org>. Acesso em: 2001.

[LOW 2001] LOWEKAMP, B; HALLARON, D.; GROSS, T. Topology Discovery for Large Ethernet Networks. In: ACM SIGCOMM, 2001, San Diego- USA. **Proceedings…** [S.l.:s.n.], 2001.

[LUC 2001] LUCCENT GRF Series Router. Disponível em: <http:// www.lucent.com/>. Acesso em: out. 2001.

[LUI 2000] LUI, K.; NAHRDTEDT, K.; CHEN, S. Hierarchical QoS Routing in Delay-Bandwidth Sensitive Networks. In: IEEE ANNUAL CONFERENCE ON LOCAL COMPUTER NETWORKS, LCN, 2000, Tampa-Florida-USA. **Proceedings…** [S.l.:s.n.], 2000.

[MCC 91] MCCLOGHRIE, K; ROSE, M. **Management Information Base for Network Management of TCP/IP-based internets:** MIB-II. [S.l.]: IETF, Mar. 1991. (Request for Comments 1213). Disponível em: <http://www.ietf.org>. Acesso em: 2001.

[MCD 99] McDYSAN, David. **Qos and Traffic Management in IP and ATM Networks**. NewYork: McGraw Hill, 1999. 456p.

[MIT 2000] MITZEL, D**. Overview of 2000 IAB Wireless Internetworking Workshop**. [S.l.]: IETF, Dec. 2000. (Request for Comments 3002). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[NET 2001] NETWORK Node Manager HP-Openview Homepage. Disponível em: <http://www.openview.hp.com/products/nnm/index.asp>. Acesso em: 2001.

[NIC 98] NICHOLS, K. et al. **Definition of the Differentiated Services Field (DS Field) in the Ipv4 and Ipv6 Headers.** [S.l.]: IETF, Dec. 1998. (Request for Comments 2474). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[NIC 99] NICHOLS, K.; JACOBSON, V.; ZHANG, L. **A Two-bit Differentiated Services Architecture for the Internet**. [S.l.]: IETF, Nov. 1999. (Request for Comments 2638). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[OMG 2001] OMG - Object Management Group. **The Common Object Request**

Broker: Architecture and Specification v. 2.5. Disponível em: <http://www.omg.com>. Acesso em: 2001.

[OPE 2001] OPENNMS Project Homepage. Disponível em: <http://www.opennms.org/>. Acesso em: 2001.

[OPE 2001a] THE OPEN Source Homepage. Disponível em: <http://www.opensource.org/>. Acesso em: 2001.

[ORC 2001] ORCHESTREAM Homepage. Disponível em: <http://www.orchestream.com/>. Acesso em: 2001.

[OUS 2000] OUSTERHOUT, John. **TCL Safe Manpage**. Disponível em: <http://www.tcl.tk/>. Acesso em: 2000.

[POS 83] POSTEL, J.; REYNOLDS, J. **TELNET Protocol Specification**. [S.l.]: IETF, May 1984. (Request for Comments 854). Disponível em: <http://www.ietf.org>. Acesso em: 2001.

[POS 85] POSTEL, J.; REYNOLDS, J. **File Transfer Protocol (FTP)**. [S.l.]: IETF, Oct. 1985. (Request for Comments 959). Disponível em: <http://www.ietf.org>. Acesso em: 2001.

[ROS 2000] ROSENBERG, J.; SCHULZRINNE, H. **A Framework for Telephony Routing over IP**. [S.1.]: IETF, June. 2000. (Request for Comments 2871). Disponível em: <http://www.ietf.org>. Acesso em: 18 dez. 2000.

[ROS 2001] ROSEN, E. C.; VISWANATHAN, A.; CALLON, R. **Multiprotocol Label Switching Architecture**. [S.l.]: IETF, Jan. 2001. (Request for Comments 3031). Disponível em: <http://www.ietf.org>. Acesso em: 28 fev. 2001.

[SCH 96] SCHULZRINNE, H. et al. **RTP: A Transport Protocol for Real-Time Applications**. [S.l.]: IETF, Jan. 1996. (Request for Comments 1889). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[SCH 2001] SCHOENWAELDER, J.; QUITTEK, J. **Script MIB Extensibility Protocol Version 1.1**. [S.l.]: IETF, Oct. 2001. (Request for Comments 2593). Disponível em: <http://www.ietf.org>. Acesso em: 2001.

[SCO 2001] SCOTTY Projetct Homepage. Disponível em: <http://snmp.cs.utwente.nl/~schoenw/scotty/ >. Acesso em: 2001.

[SHE 97] SHENKER, S.; WROCLAWSKI, J. **General Characterization Parameters for Integrated Service Network Elements**. [S.l.]: IETF, Sept. 1997. (Request for Comments 2215). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[SHE 97a] SHENKER, S.; PARTRIDGE, C.; GUERIN, R. **Specification of Guarateed Quality of Service**. [S.l.]: IETF, Sept. 1997. (Request for Comments 2212). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[SIA 99] SIAMWALLA, R.; SHARMA, R.; KESHAV, S. **Discovering Internet Topology**. Relatório Técnico: Projeto Octopus. Disponível em: <http://www.cs.cornell.edu/skeshav/papers/discovery.pdf>. Acesso em: 2001.

[SKI 2001] SKITTER Project Homepage. Disponível em: <http://www.caida.org/tools/measurement/skitter/>. Acesso em: 2001.

[SKI 2001a] SKITTER Monitors Homepage. Disponível em: <http://www.caida.org/tools/measurement/skitter/monitors/>. Acesso em: 2001.

[STA 98] STALLINGS, W. **SNMP, SNMPv2, SNMPv3, RMON and RMON2 - Practical Network Management**. 3rd ed. Reading: Addision-Wesley, 1998.

[STE 97] STEEGMANS, F. et al. **Tina Network Resource Architecture Version 3**. TINA-C Work in Progress, Feb. 1997. Disponível em: <http://www.tinac.com>. Acesso em: 2000.

[SUN 2001] SUN SOFT. **The Java Virtual Machine Specification Second Edition**. Disponível em: <http://java.sun.com/docs/books/vmspec/index.html>. Acesso em: 2001.

[TAN 96] TANENBAUM, A. S. **Computer Networks**. 3rd ed. Upper Saddler River: Prentice Hall, 1996.

[TOU 99] TORVALDS, L. **The Kernel Linux Archives**. Disponível em: <http://www.kernel.org/>. Acesso em: 2000.

[TU 2000] TU BRAUNSCHWEIG; NEC C&C RESEARCH LABORATORIES. **Jasmin Project Homepage**. Disponível em: <http://www.ibr.cs.tu-bs.de/projects/jasmin/>. Acesso em: 2002.

[UNI 2001] UNICENTER TNG Homepage. Disponível em: <http://www.cai.com/unicenter/>. Acesso em: 2001.

[VAN 2000] VANDERMEULEN, F. et al. An End to End QoS Discovery Architecture Embedded in a TINA based Multimedia Plataform. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, ISCC, 5., 2000, Antibes-France. **Proceedings…** [S.l.:s.n.], 2000.

[WHA 2001] WHATIS Online IT Technical Definitions. Disponível em: <http://whatis.techtarget.com/>. Acesso em: 2001.

[WRO 97] WROCLAWSKI, J. **Specification of the Controlled-Load Network Element Service**. [S.l.]: IETF, Sept. 1997. (Request for Comments 2211). Disponível em: <http://www.ietf.org>. Acesso em: 2000.

[XIA 99] XIAO, X.; NI, L. Internet QoS: A Big Picture. **IEEE Network**, New York, v.13, n.2, p. 8-18, Mar./Apr. 1999.

[XU 2001] Xu, D.; NAHRSTEDT, K.; WICHADAKUL, D. QoS-Aware Discovery of Wide-Area Distributed Services. In: IEEE/ACM INT'L SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, CCGRID, 2001. **Proceedings…** [S.l.:s.n.], 2001.

[YAV 2000] YAVATKAR, R.; PENDARAKIS, D.; GUERIN, R. **A Framework for Policy Based Admission Control**. [S.l.]: IETF, Jan. 2000. (Request for Comments 2753). Disponível em: <http://www.ietf.org>. Acesso em: 29 out. 2000.