Universidade Federal
de Santa Catarina

Programa de Pós-
Graduação em
Ciência da Computa-
ção

Campus Universi-
tário, Florianópolis-
SC

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, como requisito para obtenção do título de Mestre em Ciência da Computação: Sistemas de Computação.

Orientador: Prof. Roberto Willrich, Dr.

Florianópolis, 2010.

2010

Uma abordagem semântica para a especificação de Qualidade de Serviço em Redes de Computadores.

Achilles Colombo Prudêncio

Dissertação de Mestrado

Uma abordagem semântica para a especificação de Qualidade de Serviço em Redes de Computadores.

Achilles Colombo Prudêncio

Nesta dissertação é abordado o tema da especificação de Qualidade de Serviço (QoS) em serviços de rede. Enquanto trabalhos propõem o uso de uma lista fixa de parâmetros de qualidade, soluções que oferecem listas extensíveis de parâmetros são mais adequadas para lidar com a heterogeneidade de aplicações, serviços de rede, e arquiteturas de QoS adotadas pelos provedores. Esta dissertação propõe o uso de ontologias como uma maneira formal e extensível de especificação de qualidade em serviços de rede. Uma modelagem, chamada NetQoSOnt, é proposta, exemplificada, e validada através de um protótipo.

Orientador: Prof. Roberto Willrich, Dr.

Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Ciência da Computação

Achilles Colombo Prudêncio

Uma abordagem semântica para a especificação de Qualidade de Serviço em Redes de Computadores

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, como requisito para obtenção do título de Mestre em Ciência da Computação: Sistemas de Computação.

Orientador:
Roberto Willrich

Florianópolis, Santa Catarina
Fevereiro de 2010

Catalogação na fonte pela Biblioteca Universitária
da
Universidade Federal de Santa Catarina

P971u Prudêncio, Achilles Colombo
Uma abordagem semântica para a especificação de Qualidade de Serviço em Redes de Computadores [dissertação] / Achilles Colombo Prudêncio ; orientador, Roberto Willrich. - Florianópolis, SC, 2010.
134 p.: il., tabs.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da computação. 2. Qualidade dos serviços. 3. Redes de computadores. 4. Ontologias (Sistema de recuperação da informação). 5. Especificação de QoS. I. Willrich, Roberto. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 681

Uma abordagem semântica para a especificação de Qualidade de Serviço em Redes de Computadores

Achilles Colombo Prudêncio

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação: Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Mario Antonio Ribeiro Dantas, Dr.
Coordenador

Banca Examinadora

Roberto Willrich, Dr.
Orientador, UFSC

Luiz Fernando Rust da Costa Carmo, Dr.
UFRJ/INMETRO

Fernando Álvaro Ostuni Gauthier, Dr.
UFSC

Jean Marie Farines, Dr.
UFSC

"Tudo o que temos de decidir é o que fazer com o tempo que nos é dado."

J.R.R Tolkien, em *O Senhor dos Anéis: A Sociedade do Anel*, capítulo *A Sombra do Passado.*

Dedico esta dissertação a meus pais, meus irmãos e a minha mulher, que sempre acreditaram na minha capacidade, me deram apoio e “puxões de orelha” quando necessário.

AGRADECIMENTOS

Ao professor Roberto Willrich, pela orientação;

Aos professores Mário Antônio Ribeiro Dantas e Renato Fileto, pelo auxílio nos momentos de dúvida;

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo auxílio financeiro;

À secretaria do curso de Mestrado em Ciência da Computação da UFSC, pelo auxílio nos assuntos acadêmicos e sempre disponibilidade;

Aos colegas do Laboratório de Pesquisas em Sistemas Distribuídos — LaPeSD, pelo apoio nos momentos de trabalho e pela descontração nos momentos de lazer;

A todos os colegas do Mestrado em Ciência da Computação da UFSC.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AF	*Assured Forwarding*,	p. 41
AIMD	*Additive-Increase, Multiplicative-Decrease*,	p. 48
AQUILA	*Adaptive Resource Control for QoS Using an IP-based Layered Architecture*,	p. 26
ARQ	*Automatic Repeat reQuest*,	p. 48
BE	*Best Effort*,	p. 34
CoS	*Class of Service*,	p. 38
DCCP	*Datagram Congestion Control Protocol*,	p. 48
DF	*Default Forwarding*,	p. 41
DSCP	*DiffServ CodePoint*,	p. 41
EF	*Expedited Forwarding*,	p. 42
ertPS	*Extended Real-time Polling Service*,	p. 45
FL	*Frame Logic*,	p. 75
HTTP	*HyperText Transfer Protocol*,	p. 27
IP	*Internet Protocol*,	p. 26
IPDV	*IP Packet Delay Variation*,	p. 40
IPPM WG	*IP Performance Metrics Working Group*,	p. 39
MAC	*Medium Access Control*,	p. 43
MOS	*Mean Opinion Score*,	p. 35
MUO	*Measurement Units Ontology*,	p. 90
nrtPS	*Non-real-time Polling Service*,	p. 46
NSP	*Network Service Provider*,	p. 25
PHB	*Per Hop Behaviour*,	p. 41
QDINE	*Managing Quality of experience Delivery In New generation telecomunication networks with E-negotiation*,	p. 81

QoE	*Quality of Experience*,	p. 35
QoS	*Quality of Service*,	p. 25
RFC	*Request For Comments*,	p. 39
rtPS	*Real-time Polling Service*,	p. 46
SDU	*Service Data Unit*,	p. 44
SLA	*Service Level Agreement*,	p. 25
SLS	*Service Level Specification*,	p. 25
SOAP	*Simple Object Access Protocol*,	p. 27
SWRL	*Semantic Web Rule Language*,	p. 72
SWS	*Semantic Web Services*,	p. 28
TCP	*Transmission Control Protocol*,	p. 26
TEQUILA	*Traffic Engineering for Quality of Service in the Internet, at Large Scale*,	p. 26
TFRC	*TCP Friendly Rate Control*,	p. 48
UDDI	*Universal Description, Discovery and Integration*,	p. 27
UGS	*Unsolicited Grant Service*,	p. 45
UML	*Unified Modeling Language*,	p. 29
UNA	*Unique Name Assumption*,	p. 58
URI	*Uniform Resource Identifier*,	p. 58
VoD	*Video on Demand*,	p. 27
VoIP	*Voice over IP*,	p. 25
WiMAX	*Worldwide Interoperability for Microwave Access*,	p. 43
WS	*Web Services*,	p. 27
WSDL	*Web Services Description Language*,	p. 28
XML	*eXtensible Markup Language*,	p. 27

RESUMO

Nesta dissertação é abordado o tema da especificação de Qualidade de Serviço (QoS) em serviços de rede. Em operações relacionadas ao gerenciamento de QoS, sempre é necessário especificar os parâmetros de qualidade sendo negociados. Atualmente não existem padrões que descrevem formalmente como deve ser feita a especificação de QoS. Enquanto trabalhos propõem o uso de uma lista fixa de parâmetros de qualidade, soluções que oferecem listas extensíveis de parâmetros são mais adequadas para lidar com a heterogeneidade de aplicações, serviços de rede, e arquiteturas de QoS adotadas pelos provedores. Inspirada na Web Semântica e em problemas similares enfrentados por Serviços Web, esta dissertação propõe o uso de ontologias como uma maneira formal e extensível de especificação de qualidade em serviços de rede. Uma modelagem para especificação, chamada NetQoSOnt, é proposta, exemplificada, e validada através de um protótipo. Esta modelagem tem o objetivo de ser utilizada como base para sistemas de QoS, permitindo a especificação formal de parâmetros de qualidade. Os resultados da implementação do protótipo demonstram a viabilidade da utilização de NetQoSOnt em sistemas automatizados de gerenciamento de QoS.

Palavras-chave: Qualidade de Serviço, Redes de Computadores, Especificação de QoS, Ontologias, Tomada Automatizada de Decisões.

ABSTRACT

In this thesis, the topic of Quality of Service (QoS) specification in network services is approached. In operations related to QoS management, it is always necessary to specify the quality parameters being negotiated. Currently there are no formal standards for QoS specification. While work propose the use of a fixed list of quality parameters, solutions that offer a extensible list of parameters are more adequate to deal with the heterogeneity of applications, network services, and QoS architectures adopted by providers. Inspired by the Semantic Web and by similar problems faced by Web Services, this thesis proposes the use of ontologies as a formal and extensible way of specifying quality in network services. A specification modeling, called NetQoSOnt, is proposed, exemplified, and validated through a prototype. This modeling has the objective of serving as basis to QoS systems, allowing the formal specification of quality parameters. The results of the implementation of the prototype show the viability of use of NetQoSOnt in automated QoS management systems.

Keywords: Quality of Service, Computer Networks, QoS Specification, Ontologies, Automated Decision Making.

1 INTRODUÇÃO

O aumento do uso de aplicações de tempo real (VoIP, videoconferência, jogos, entre outros) e a alta disponibilidade requerida por clientes empresariais e governamentais aumentaram também a necessidade de garantir Qualidade de Serviço (QoS). QoS, por sua vez, compreende o conjunto de requisitos de uma aplicação que um sistema deveria satisfazer para obter a qualidade desejada (LU, 1996). Para garantir QoS em serviços de rede, foram propostas diversas arquiteturas, tais como IntServ, DiffServ, etc.

Mas, antes de utilizar qualquer mecanismo de rede com suporte a QoS os clientes, usuários ou aplicações precisam especificar seus requisitos de qualidade. A especificação de QoS é utilizada em várias operações de gerenciamento de QoS, entre elas: a seleção do provedor, a invocação do serviço, a negociação de qualidade e o monitoramento do serviço.

No momento da subscrição em um serviço de rede, o Provedor de Serviços de Rede (NSP — *Network Service Provider*) e o cliente estabelecem um contrato, denominado Acordo de Nível de Serviço (SLA — *Service Level Agreement*). O SLA permite que o NSP e o cliente definam parâmetros de qualidade mínimos que o NSP deve garantir ao cliente, assim como o preço pelo fornecimento dos serviços e punições caso o provedor falhe com esse fornecimento, ou caso o cliente faça mau uso dos serviços fornecidos.

Um conjunto de parâmetros de qualidade, seus valores e suas métricas formam a especificação técnica de um SLA, também chamada de Especificação de Nível de Serviço (SLS — *Service Level Specification*) (TEQUILA, 2002). Um SLA pode conter um ou mais SLSs e cada SLS pode variar de acordo com o tipo do serviço prestado. Por exemplo, um provedor de acesso de redes sem fio pode garantir que, dentro de sua área de cobertura, a potência do seu sinal nunca será

menor que um certo percentual. Outro exemplo seria o de um provedor de serviços de Internet que pode garantir que, dentro de seu domínio, a transmissão de pacotes nunca terá um atraso maior que 100 milissegundos.

Provedores diferentes do mesmo tipo de serviço também podem ter nomes e métricas diferentes para o mesmo parâmetro de qualidade. Isso ocorre devido ao uso de diferentes terminologias e também ao uso de nomes fantasia. Por exemplo, enquanto um provedor pode chamar o atraso de transmissão de "atraso" em seus serviços, outro provedor pode preferir o termo "latência".

Alguns trabalhos na área de QoS tentaram tratar essa falta de normalização em termos de parâmetros de QoS definindo um conjunto padrão de parâmetros a serem adotados. Uma destas tentativas foi feita pelo projeto TEQUILA (***T****raffic* ***E****ngineering for* ***Qu****ality of Service in the* ***I****nternet, at* ***La****rge Scale*) (TEQUILA, 2002). Semelhante a maioria dos outros trabalhos na área, (TEQUILA, 2002) adota parâmetros da camada de rede referentes ao protocolo IP, como *One Way Delay* (ALMES; KALIDINDI; ZEKAUSKAS, 1999a), *Packet Delay Variation* (DEMICHELIS; CHIMENTO, 2002) e *Packet Loss Rate* (ALMES; KALIDINDI; ZEKAUSKAS, 1999b), que são independentes das camadas de rede inferiores. Entre outros exemplos de propostas que definem um conjunto de parâmetros padrão está a do projeto AQUILA (***A****daptive Resource Control for* ***Q****oS* ***U****sing an* ***I****P-based* ***L****ayered* ***A****rchitecture*) (SALSANO et al., 2000).

Entretanto, a adoção de uma lista pré-definida de parâmetros não é uma solução ideal para especificar a QoS. Existe uma grande quantidade de provedores oferecendo soluções de QoS nas diferentes camadas da pilha TCP/IP, como provedores em nível de enlace, que alugam trechos de fibra ótica; provedores em nível de rede, que oferecem conectividade IP; provedores de serviços aplicativos, que fornecem serviços de

VoIP, vídeo sob demanda (VoD — *Video on Demand*), entre outros.

Obviamente, uma quantidade limitada e de difícil expansão de parâmetros não serviria para atender a todos os provedores existentes. De maneira ideal, uma especificação de QoS deveria ser de fácil expansão e ser capaz de descrever parâmetros quantitativos e/ou qualitativos de forma declarativa (sem expressar como eles devem ser impostos ou mantidos). Para ser adequada a diversos cenários, onde possam ser adotadas diferentes tecnologias, soluções de gerenciamento de QoS e níveis de serviço, uma abordagem de especificação de QoS deveria permitir o uso de um conjunto flexível de parâmetros e métricas. Deveria também permitir que especificações de camadas diferentes possam ser relacionadas, de modo que a escolha da configuração de certos parâmetros em uma camada superior implique na configuração de outros parâmetros nas camadas inferiores, de maneira transparente ao usuário.

Uma especificação de QoS utilizando parâmetros em diferentes níveis exige um mapeamento eficiente: parâmetros qualitativos e quantitativos devem ser compreendidos e mapeados em parâmetros e soluções dependentes de tecnologia. Ou seja, a descrição deve poder ser traduzida pelo provedor em ações de configuração nos mecanismos e políticas do sistema (JINGWEN; NAHRSTEDT, 2004).

Problemas similares aos discutidos ocorrem na área de Serviços Web (*Web Services* — WS). WS são sistemas computacionais que possuem uma interface de aplicação acessível através da Web, utilizando tecnologias padrão, como HTTP e XML (DAVIES; STUDER; WARREN, 2006). Eles são publicados em um registro centralizado utilizando um padrão baseado em XML, denominado UDDI (*Universal Description, Discovery and Integration*). Uma busca é feita no registro (utilizando SOAP — *Simple Object Access Protocol*) para descobrir serviços e suas descrições (como funcionam).

Cada serviço é descrito utilizando a linguagem WSDL (*Web Services Description Language*), que define as mensagens suportadas pelo serviço, seus parâmetros e a resposta fornecida (DAVIES; STUDER; WARREN, 2006).

Os padrões descritos oferecem interoperabilidade entre serviços apenas no nível sintático: a competência de um desenvolvedor é necessária para realizar a busca no registro de modo a recuperar a lista de serviços, descobrir os que fazem a computação desejada, e desenvolver um cliente que utilize estes serviços de acordo. Para superar estas limitações, pesquisas em WS sugerem maneiras de embutir semântica no fluxo de trabalho de WS, de modo a automatizar a interoperação entre serviços e minimizar a necessidade de interação humana. Estas pesquisas formam uma nova área de interesse, chamada Serviços Web Semânticos (*Semantic Web Services* — SWS) (DAVIES; STUDER; WARREN, 2006).

Um dos aparatos semânticos mais utilizados em SWS são as ontologias. Uma ontologia pode ser definida como "uma especificação explícita e formal da conceitualização de um domínio de interesse" (DAVIES; STUDER; WARREN, 2006). Formal, nesse sentido, significa que esta especificação permite o processamento automático por computadores. Em SWS, elas são utilizadas para descrever semanticamente os serviços. Um conjunto padrão de conceitos é estendido de modo a descrever os vários tipos de serviços possíveis em WS. Além de permitir a descrição funcional de WS, as ontologias permitem a descrição de requisitos não funcionais, incluindo QoS, tornando-se um mecanismo formal que pode amenizar o problema de diferentes terminologias e métricas. A Web Ontology Language (OWL) (W3C, 2004a) é a linguagem padrão da W3C para a codificação de ontologias. Na área de SWS, várias ontologias de QoS foram criadas utilizando OWL para especificar qualidade e diferenciar WS além do nível funcional ((DOBSON; LOCK, 2005), (ZHOU; CHIA; LEE, 2005), (BLEUL; WEISE, 2005)).

Para permitir a flexibilização em termos de parâmetros de qualidade para especificar QoS, é necessário um formalismo que atenda uma série de requisitos: permitir a especificação de parâmetros e métricas de QoS; permitir a especificação dos relacionamentos/mapeamentos entre parâmetros e métricas; e permitir a interpretação automática destes parâmetros.

Existem alguns formalismos que permitem expressar estes problemas. A *Unified Modeling Language* (UML), por exemplo, é uma linguagem visual bastante rica e que permite especificar alguns dos requisitos citados. A semântica de UML, no entanto, não é restrita o suficiente de modo a permitir a inferência automatizada (OMG, 2009), de modo que UML pode ser considerado um recurso interessante somente para visualizar graficamente modelos semânticos. Ontologias, por outro lado, são modelos semânticos com toda a expressão necessária para atender aos requisitos citados, e o modelo formal utilizado para definir ontologias possibilita a inferência automatizada.

1.1 OBJETIVO GERAL

Considerando o problema da especificação e mapeamento de QoS em redes, e inspirado nos progressos na área de SWS, esta dissertação propõe tratar o problema de transparência/extensibilidade de parâmetros de QoS em serviços de rede utilizando uma abordagem semântica. Através desta abordagem, é possível não só definir um vocabulário comum, mas permitir a extensão deste vocabulário. A modelagem proposta é projetada para ser a base para abordagens de gerenciamento de QoS baseado em semântica, incluindo seleção, negociação, invocação e monitoramento de serviço.

A modelagem utilizada também permitirá a comparação automatizada de diferentes parâmetros de QoS. Os meios para estender a modelagem e realizar essa comparação serão

exemplificados e demonstrados através de exemplos. Esses exemplos serão a base para validação da proposta.

1.2 OBJETIVOS ESPECÍFICOS

Como objetivos específicos, esta dissertação tem como pretensão de:

1. Apresentar os principais conceitos envolvidos na especificação de parâmetros de QoS de modo geral, e mais profundamente de QoS em redes;

2. Apresentar os principais conceitos envolvidos na modelagem semântica, com foco em ontologias codificadas em OWL;

3. Mostrar e comparar ontologias de QoS já existentes, tanto para a área de redes, quanto de WS;

4. Definir a ontologia para especificação de QoS em serviços de rede, chamada de NetQoSOnt, e demonstrar o seu uso através de exemplos;

5. Validar a proposta utilizando um protótipo.

1.3 JUSTIFICATIVA

Na área de redes de computadores, não existe um padrão amplamente adotado de especificação de QoS. Esta falta de normalização leva ao conhecido problema de interoperabilidade: clientes e provedores descrevem seus requisitos e garantias utilizando definições e terminologia próprias.

A negociação da qualidade entre domínios é prejudicada, pois muitas vezes estas definições são incompatíveis, o que dificulta a comparação de parâmetros.

O estabelecimento de um contrato com um provedor restringe o cliente ao mesmo, exigindo um processo de migração e adaptação dos dados caso haja a necessidade de mudança ou mesmo de interação com outro provedor.

Para um desenvolvedor de aplicativos, ocorre um problema similar quanto a como especificar parâmetros de qualidade. Utilizar os parâmetros e as métricas de um provedor pode tornar difícil a interação com um provedor diferente. Há também o problema da camada em que o serviço está sendo oferecido. Apesar de estar trabalhando em um nível mais alto, é necessário especificar a qualidade em nível de rede, ao negociar com um NSP.

1.4 ORGANIZAÇÃO DESTA DISSERTAÇÃO

O restante desta dissertação está organizada da seguinte maneira: no capítulo 2, é realizada uma revisão sobre os conceitos envolvidos na especificação de QoS; no capítulo 3, são revisados os conceitos relacionados à semântica computacional e ao desenvolvimento de ontologias; no capítulo 4, são apresentados e discutidos trabalhos relacionados ao tema desta dissertação; no capítulo 5, é apresentada NetQoSOnt e sua modelagem de especificações de qualidade de serviço, modelagem esta exemplificada através de um cenário de uso e validada através do desenvolvimento de um protótipo; finalmente, no capítulo 6, são tiradas conclusões acerca da pesquisa realizada e sugeridos trabalhos futuros.

2 QUALIDADE DE SERVIÇO

Neste capítulo é realizada uma revisão dos conceitos envolvidos na especificação de QoS de modo geral e um aprofundamento sobre a especificação de QoS em redes de computadores.

2.1 SOBRE QUALIDADE DE SERVIÇO

A junção de serviços de tempo real à rede de dados aumentou consideravelmente a sua complexidade. Estes serviços englobam não somente a comunicação por voz, que foi somente o início, mas a comunicação por vídeo, sincronização entre sistemas (por exemplo em jogos com múltiplos jogadores jogando a mesma partida em redes separadas por grandes distâncias), alta disponibilidade (por exemplo em troca de informações sensíveis entre governos, empresas), entre outros.

Recentemente, houve um grande aumento da necessidade de garantir a qualidade do serviço em redes de comutação de pacotes. Uma das primeiras aplicações a ter esta demanda foi a junção das redes de comunicação de dados e das redes de comunicação de voz (PARK, 2005).

Realizado em uma rede de comutação de circuitos, o serviço de comunicação de voz oferece alta qualidade de chamada para seus clientes, pois estabelece uma linha dedicada para cada chamada. Obviamente, esta qualidade vem com um preço, uma vez que a linha dedicada ocupa uma grande quantidade de recursos do sistema, mesmo que não esteja sendo usada (quando as duas pessoas ficam em silêncio durante uma chamada telefônica, por exemplo).

As redes de comutação de pacotes são redes projetadas para o envio de dados, o que na maioria das ocasiões não é um serviço de tempo real. Todos os fluxos de dados são tratados de maneira igual, e não existem garantias de tempo

e entrega de pacotes, em um esquema de operação chamado de Melhor Esforço (*Best Effort* — BE) (PARK, 2005).

A operação de envio de pacotes em BE não é suficiente para serviços de tempo real. São necessários mecanismos para oferecer garantias em termos de desempenho do serviço, pois a perda ou o atraso destes é altamente prejudicial para várias aplicações/serviços (por exemplo, uma mensagem de voz não pode ser reconstruída se os pacotes forem perdidos ou recebidos totalmente fora de ordem). E para realizar esta classificação dos dados que trafegam na rede, é necessário definir e especificar QoS de maneira clara.

A classificação é apenas o primeiro componente da garantia da QoS. Existem quatro princípios que toda arquitetura de QoS deve seguir para garantir qualidade (KUROSE; ROSS, 2006): o primeiro deles é a correta classificação e diferenciação dos fluxos de dados; o segundo diz respeito ao isolamento: cada fluxo pode utilizar somente os recursos que lhe forem alocados; o terceiro diz respeito à eficiência: os recursos do sistema como um todo devem ser divididos entre os fluxos passantes, e ao mesmo tempo devem ser utilizados de maneira eficiente; o quarto e último princípio diz respeito à admissão de fluxos: cada fluxo deve declarar seus requisitos antes de utilizar um sistema, e se este sistema julgar que não consegue atendê-los, pode bloquear o fluxo.

2.2 DEFINIÇÃO DE QOS

Dentre as várias definições de Qualidade de Serviço trazidas por (MARCHESE, 2007) e (PARK, 2005), a considerada mais simples e completa para este trabalho é a de que QoS é o *grau com o qual um conjunto de características de um sistema atende a determinados requisitos.*

Considerando esta definição, é possível dividir QoS em dois tipos, como feito em (PARK, 2005):

QoS do Ponto de Vista do Sistema, medida e descrita em termos de parâmetros quantitativos. Estes parâmetros podem representar valores mínimos, valores máximos, ou mesmo intervalos de valores onde determinada característica de qualidade do sistema é garantida.

QoS do Ponto de Vista do Usuário, também chamada de Qualidade de Experiência (*Quality of Experience* — QoE) ou Qualidade Percebida. Geralmente é descrita em parâmetros subjetivos como "chamada de voz sem eco/atraso", "chamada de vídeo sem imagens borradas e áudio sincronizado", embora existam métricas mais objetivas, como a *Mean Opinion Score* (MOS) (ITU-T, 1996).

MOS consiste em uma "média de opinião" dos usuários, que avaliam o serviço dando notas de 1 a 5, sendo que, quanto mais baixa a nota, pior é a qualidade do serviço. Embora seja uma métrica descrita de forma objetiva, MOS corresponde na verdade a uma avaliação subjetiva, que depende também do tipo do serviço. Estudos descritos em (MARCHESE, 2007) fazem a correspondência entre parâmetros de QoS Intríseca e MOS, para o serviço de conferência de vídeo e áudio.

2.2.1 Definição de QoS para Aplicações em Redes de Computadores

Sobre a QoS do sistema para aplicações em redes de computadores, é possível afirmar que ela possui vários níveis, relacionados às camadas da pilha TCP/IP:

Qualidade em Nível de Aplicação é aquela expressa em termos de parâmetros de aplicativo, como a taxa de quadros de um fluxo de vídeo, a taxa de bits de um fluxo de áudio, a disponibilidade de um WS, entre outros.

Qualidade em Nível de Transporte é a qualidade expressa em termos da configuração do protocolo de transporte, como controle de congestionamento, tamanho da janela de envio, entre outros.

Qualidade em Nível de Rede é aquela expressa em termos de parâmetros de configuração do protocolo de rede, em geral relacionados ao protocolo IP, como taxa de perda, atraso, variação de atraso e vazão.

Qualidade em Nível de Enlace envolve parâmetros em redes cabeadas ou sem fio, como priorização de tráfego, potência de sinal e latência.

2.3 ESPECIFICAÇÃO DE QOS

Uma especificação de QoS pode ser definida como um conjunto de parâmetros que, junto com seus respectivos valores, definem a qualidade oferecida por uma aplicação ou serviço (GROSSMAN, 2002). Parâmetros, por sua vez, são tipos de medição de qualidade, e valores são manifestações ou medições dos parâmetros. Especificações de QoS podem ser utilizadas tanto para descrever a oferta de qualidade de um provedor quanto os requisitos de qualidade de um cliente.

A especificação da QoS é importante em várias operações de gerenciamento de qualidade em redes, como a subscrição com o provedor, a negociação de qualidade, a invocação do serviço e o monitoramento do serviço.

A subscrição com o provedor envolve a negociação de um contrato de serviço, denominado Acordo de Nível de Serviço ou SLA. O SLA inclui uma parte não técnica (identificação das partes, preço do fornecimento dos serviços e punições para ambas as partes caso o contrato não seja honrado) e uma parte técnica, chamada de Especificação de Nível de Serviço ou SLS, que inclui os parâmetros de qualidade (em

conjunto com seus valores e suas métricas) que devem ser garantidos pelo provedor ao fornecer o serviço para o cliente.

Em geral, os provedores adotam provisionamento estático de QoS, onde toda a negociação é realizada manualmente, o que resulta em uma longa atividade offline, e tem como produto final os chamados SLAs Estáticos.

A negociação da qualidade também pode ser feita de maneira dinâmica. Para isso, é necessário que cliente e provedor acordem também na sintaxe do SLA a ser negociado, de modo que a negociação possa ser feita de maneira automática. SLAs dinâmicos devem ser vistos como "documentos vivos" no sentido de que eles podem ser modificados frequentemente, de acordo com as necessidades atuais do cliente e a disponibilidade do provedor (MESCAL, 2005). Este tipo de SLA possui uma validade de curta duração e pode ser negociado com base em fluxos mínimos de utilização. Vários trabalhos de pesquisa propõem protocolos de negociação, como por exemplo COPS-SLS (NGUYEN et al., 2002), DSNP (CHEN et al., 2006), SLS IPCP (CLERCQ et al., 2000) e SrNP (GODERIS et al., 2002).

A Invocação do Serviço é aquela feita anteriormente a utilização do mesmo. Ela pode ser implícita ou explícita (TEQUILA, 2002). Na invocação implícita, a QoS a ser garantida é definida em um SLA negociado previamente e às vezes requere protocolos de sinalização de QoS para reserva de recursos. Em uma invocação explícita, o usuário seleciona o nível de QoS no momento da invocação. Neste caso, o usuário ou aplicação especifica seus requisitos de QoS para o provedor, que pode aceitar ou não o pedido, dependendo da carga do sistema.

O Monitoramento do Serviço assegura a detecção de violações no SLA. Idealmente, um terceiro (uma empresa de segurança, por exemplo) de comum acordo entre cliente e provedor é designado para monitorar o uso dos recursos pelo cliente e o provimento destes pelo provedor, sinalizando ao

encontrar violações (MORAES et al., 2008). Quando encontradas, estas devem ser tratadas no modo previamente estabelecido no SLA.

As operações de gerenciamento de qualidade só são possíveis devido à existência de parâmetros claros de desempenho em redes, descritos na seção 2.5.

2.4 CLASSES DE SERVIÇO

Uma Classe de Serviço, também referida como CoS (do inglês *Class of Service*), compreende uma classificação dada pelo provedor a aplicações com características de tráfego e desempenho semelhantes. Por exemplo, aplicações com requisitos de tempo real como chamadas de voz e outros serviços de telefonia podem ser classificadas usando a mesma CoS (BABIARZ; CHAN; BAKER, 2006).

E os requisitos de cada CoS de um provedor, por sua vez, são declarados utilizando especificações de QoS. Os valores ideais dos parâmetros garantidos são escolhidos de modo que para o agregado de tráfego que a CoS representa a utilização do serviço seja satisfatória.

2.5 ESPECIFICAÇÃO DE QOS NA CAMADA INTERNET DA PILHA TCP/IP

Existem provedores oferecendo serviços em todas as camadas da pilha TCP/IP. No entanto, uma das camadas que possuem maior importância é a camada de rede, justamente pelo fato dessa camada fornecer a conectividade entre os vários serviços. Em decorrência do sucesso e da amplitude da Internet, o padrão IP ganhou grande importância, gerando interesse dos provedores em trabalhar na camada de rede. Há uma quantidade significativa de provedores, e os provedores de outras camadas por vezes moldam seus serviços aos requisitos de rede.

Com a necessidade de garantir qualidade na Internet, foram desenvolvidos diversos RFCs (*Request For Comments*, ou "Requisição de Comentários") para padronizar os parâmetros de desempenho na camada de rede. O grupo de trabalho IPPM (*IP Performance Metrics Working Group*) lançou a RFC 2330 (PAXSON et al., 1998), que define um arcabouço geral para o desenvolvimento de parâmetros de desempenho em uma rede IP. Esta RFC também define um vocabulário comum a ser usado na definição de parâmetros e metodologias para a medição destes. A partir das definições da RFC 2330, foram criados os principais parâmetros de desempenho de redes:

***One-way Delay*:** o atraso em um sentido, também chamado simplesmente de atraso, é definido na RFC 2679 (ALMES; KALIDINDI; ZEKAUSKAS, 1999a). Na sua forma mais simples, é o tempo em que um pacote leva para sair da fonte e chegar ao destino. Esta medição simples é realizada várias vezes em um tempo determinado (seguindo uma distribuição de Poisson), e a média destes valores é o valor a ser realmente considerado para definir o atraso. A RFC 2679 discute vários requisitos para medir corretamente este tempo (por exemplo, sincronização dos relógios entre as entidades), mas que não entrarão em discussão neste trabalho.

***One-way Packet Loss*:** a perda de pacotes em um sentido, também chamada de taxa de perda de pacotes, é definida na RFC 2680 (ALMES; KALIDINDI; ZEKAUSKAS, 1999b). Quando um pacote é enviado da fonte ao destino, o tempo que este pacote leva para chegar ao destino (o atraso) é monitorado. Se este tempo ultrapassar um limiar mínimo, o pacote é considerado perdido. Da mesma forma que é feita a medição de atraso, várias amostras de envio de pacotes são tomadas dentro de um intervalo de tempo seguindo uma

distribuição Poisson, e o percentual de pacotes perdidos configura o parâmetro de perda de pacotes.

***One-way IP Packet Delay Variation*,** ou *One-way* IPDV. A variação do atraso de pacotes em um sentido é definida na RFC 3393 (DEMICHELIS; CHIMENTO, 2002). A medição deste parâmetro é feita escolhendo dois pacotes de um fluxo de pacotes, utilizando uma função pré-definida (que depende da entidade que faz a medição e não é especificada na RFC) e seus atrasos são medidos. A diferença entre estes atrasos é a variação de atraso. Assim como ocorre com os parâmetros atraso e perda, essa medição é feita várias vezes e a média quantifica realmente o parâmetro.

***Network Capacity*,** ou *throughput*. A capacidade de trágego, também chamado de vazão de tráfego, é definida na RFC 5136 (CHIMENTO; ISHAC, 2008). Esta RFC contém várias definições para capacidade de tráfego em uma rede IP. Destas definições, a mais utilizada é a capacidade de tráfego em um enlace, que é definida como número máximo de bits na camada IP (que correspondem a oito vezes o número de octetos em todos os pacotes IP recebidos corretamente) que podem ser transmitidos corretamente entre dois *hosts* em um intervalo de tempo. Vale lembrar que o enlace corresponde a uma conexão direta entre dois *hosts*. Quando se fala de um caminho completo entre a fonte e o destino, a capacidade do caminho é a equivalente à capacidade do enlace de menor capacidade deste caminho, que pode variar com o tempo devido a variações no nível de congestionamento dos enlaces.

Vários trabalhos de pesquisa que tratam QoS de rede focam-se nestes parâmetros (por exemplo (TEQUILA, 2002),

(ALÍPIO; NEVES; CARVALHO, 2007), (EXPOSITO; SÉNAC; DIAZ, 2004)). No entanto, os parâmetros de qualidade das outras camadas da pilha TCP/IP, e sua relação com os parâmetros de qualidade de rede também são importantes, e são tratados na próxima seção.

O IPPM, como dito anteriormente, é um esforço voltado para a definição de parâmetros de desempenho em redes IP. Logo, não é feito, dentro deste grupo a criação de Classes de Serviço. Outros padrões da IETF, como por exemplo IntServ e DiffServ, utilizam os parâmetros definidos pelo IPPM na criação de arquiteturas de QoS completas, incluindo a definição de CoS.

A arquitetura DiffServ (*Differentiated Services*) (BLAKE et al., 1998) é uma das mais difundidas e implementadas atualmente. Ela define um conjunto padrão de CoS, chamados na RFC de *Per Hop Behaviour* (PHB) cujos tipos de tráfego que elas devem comportar, e quais os meios que podem ser utilizados para classificar e moldar o tráfego (configuração de filas, entre outros). O conjunto de CoS definidas por DiffServ, identificado pelo campo DSCP (*DiffServ CodePoint*) do pacote IP, é apresentado a seguir (BABIARZ; CHAN; BAKER, 2006):

***Default Forwarding* (DF):** é a CoS padrão, correspondente ao serviço de melhor esforço ou *Best Effort*. Uma vazão mínima de tráfego é garantida, mas não há garantia de ordem ou entrega de pacotes;

***Assured Forwarding* (AF):** é na verdade um conjunto de CoSs, com diferentes níveis de garantia em atraso, variação de atraso, perda e vazão. Todos os pacotes que excedem os limites são descartados. Existem quatro classes, e três níveis de descarte para cada classe. Os serviços que as classes AF devem oferecer garantia abrangem conferência multimídia (classe 4), *strea-*

ming multimídia (classe 3), transações cliente-servidor (classe 2) e aplicações *store and forward* (classe 1);

***Expedited Forwarding* (EF):** é a CoS de maior prioridade, designada para serviços com baixa tolerância a atraso, variação de atraso, perda e vazão. O classe principal de serviço que EF deve oferecer garantia é a de telefonia;

Como dito, a arquitetura DiffServ especifica quais as classes de serviço e os mecanismos que podem ser utilizados para classificar e moldar o tráfego de acordo com estas classes. A configuração real destes mecanismos é determinada pelos provedores, o que abre margem para interpretações diferentes do padrão (por exemplo, a classe EF para um provedor pode oferecer a mesma garantia que a classe AF11 de outro provedor).

2.5.1 Parâmetros de Desempenho nas Outras Camadas da Pilha TCP/IP

Como já afirmado anteriormente, todas as camadas da pilha TCP/IP possuem seus próprios parâmetros de desempenho. Como os dados produzidos em uma camada superior precisam atravessar as camadas inferiores para serem transmitidos, o desempenho das camadas inferiores afeta o desempenho de todo o sistema.

Todos os parâmetros descritos na seção anterior foram projetados para descrever o desempenho da camada de rede. Embora a camada de rede seja muito importante para a pilha TCP como um todo, é inconveniente para os desenvolvedores e provedores das outras camadas precisar projetar suas implementações sempre pensando na camada de rede.

Com excessão da camada de rede, não existe mais nenhum padrão de protocolo e especificação de QoS amplamente usados. Na camada de enlace, existem diversos meios

de transmissão, cada um com suas características; na camada de transporte, existe uma grande quantidade de protocolos disponíveis; finalmente, na camada de aplicação, existem inúmeras categorias de aplicativos que precisam de gerenciamento de qualidade, o que torna a criação de um padrão único algo inviável na prática.

A partir de agora, serão apresentados alguns exemplos de padrões e estudos das outras camadas da pilha TCP/IP e como eles parametrizam QoS. Será apresentado também um exemplo de parametrização de qualidade percebida.

2.5.1.1 Camada de Enlace

Existem muitos padrões atualmente para oferecer conectividade na camada de enlace. Todos os padrões existentes, ATM, Frame Relay, IEEE 802.3 (redes cabeadas) e IEEE 802.11 (um dos padrões para redes sem fio), tratam qualidade de serviço de maneiras similares. A título de exemplo, detalhes sobre QoS para o padrão WiMAX para redes sem fio serão apresentados.

WiMAX (*Worldwide Interoperability for Microwave Access*), é uma tecnologia baseada no padrão IEEE 802.16e-2005 para a transmissão sem-fio em banda larga de dados para assinantes fixos e móveis (IEEE, 2005).

O padrão citado descreve as camadas física e de controle de acesso ao meio (*Medium Access Control* - MAC) enumerando canais, algoritmos e hardware de transmissão utilizados na camada física, e também como é feita criptografia, encapsulamento de outros protocolos (como IP) na interface sem-fio e QoS na camada MAC, dentre várias outras definições.

O padrão IEEE 802.16e-2005 define também tipos específicos de parâmetros de desempenho (IEEE, 2005):

***Traffic Priority*:** Indica priorização do tráfego. É um valor numérico de 0 a 7, sendo que quanto maior o número, mais alta é a prioridade;

***Maximum Sustained Traffic Rate*:** Define a taxa máxima de transmissão de dados. É um valor numérico expresso em bits por segundo;

***Maximum Traffic Burst*:** Define o tamanho máximo de rajada acomodado pelo serviço. É um valor numérico expresso em bytes.

***Minimum Reserved Traffic Rate*:** Define a taxa de transmissão mínima reservada para o fluxo de serviço. É um valor numérico expresso em bits por segundo;

***Uplink Grant Scheduling Type*:** Valor numérico de 0 a 255 que identifica o tipo da classe de serviço;

***Request/Transmission Policy*:** Valor numérico que define configurações do fluxo de serviço (o modo como a estação de base trata o fluxo);

***Maximum Latency*:** Define a latência máxima da transmissão em um sentido de um pacote entre a estação de base e o assinante, medida em milissegundos;

***Fixed-length Versus Variable-length SDU Indicator*:** Define se uma Unidade de Dados de Serviço (*Service Data Unit* — SDU) é de tamanho fixo ou variável;

***Tolerated Jitter*:** Define a variação de atraso máxima para a conexão, em milissegundos;

***Unsolicited Grant Interval*:** Define o intervalo de concessão de dados para o fluxo de serviço, em milissegundos;

***Unsolicited Polling Interval*:** Define o intervalo nominal máximo entre concessões de *polling* sucessivas para o fluxo de serviço, em milissegundos;

***SDU Size*:** Tamanho em bytes da SDU.

O provimento de qualidade de serviço pelo padrão 802.16-e-2005 (e consequentemente pela tecnologia WiMAX) especifica um conjunto de 5 classes de serviço. Cada conexão entre um assinante e a estação de base (definida no padrão como sendo um equipamento que promove a conectividade e faz o gerenciamento dos assinantes) do provedor mais próxima a ele é chamada fluxo de serviço e é alocada a uma destas classes (IEEE, 2005):

***Unsolicited Grant Service* (UGS):** Transmissão de dados de tempo real compostos de pacotes de tamanho fixo enviados em intervalos fixos de tempo. Essa classe engloba serviços como T1/E1 e VoIP sem supressão de silêncio. Para a classe UGS, devem ser definidos valores para os seguintes parâmetros:

- *Tolerated jitter*;
- *Fixed length SDU*;
- *SDU size*;
- *Minimum reserved traffic rate*;
- *Maximum Latency*;
- *Request/Transmission Policy*;
- *Unsolicited Grant Interval*.

***Extended Real-time Polling Service* (ertPS):** Fluxos de serviço de tempo real que geram pacotes de dados de tamanho variável em intervalos periódicos. Essa classe engloba serviços como VoIP com supressão de silêncio. Na classe ertPS, devem ser definidos valores para os seguintes parâmetros:

- *Maximum Latency*;
- *Tolerated Jitter*;
- *Minimum Reserved Traffic Rate*;
- *Maximum Sustained Traffic Rate*;
- *Traffic Priority*;
- *Request/Transmission Policy*;
- *Unsolicited Grant Interval*.

Real-time Polling Service **(rtPS):** Transmissão de dados de tempo real compostos de pacotes de tamanho variável enviados em intervalos fixos de tempo, englobando serviços como transmissão de vídeo MPEG. Para a classe rtPS, devem ser definidos valores para os seguintes parâmetros:

- *Maximum Latency*;
- *Minimum Reserved Traffic Rate*;
- *Maximum Sustained Traffic Rate* (Opcional);
- *Traffic priority*;
- *Request/Transmission policy*;
- *Unsolicited Polling Interval*.

Non-real-time Polling Service **(nrtPS):** Transmissão de dados tolerantes a atraso compostos de pacotes de tamanho variável para os quais é necessária uma taxa mínima de transferência, como FTP, por exemplo. Na classe nrtPS, devem ser definidos valores para os seguintes parâmetros:

- *Minimum Reserved Traffic Rate*;
- *Maximum Sustained Traffic Rate* (Opcional);
- *Traffic priority*;

- *Request/Transmission policy.*

***Best Effort* (BE):** Transmissão de dados que não requerem uma qualidade mínima e podem ser tratados de acordo com a disponibilidade, como HTTP, por exemplo. Para a classe BE, devem ser definidos valores para os seguintes parâmetros:

- *Maximum Sustained Traffic Rate*;
- *Traffic priority*;
- *Request/Transmission policy.*

A configuração e os valores ótimos destes parâmetros para cada classe de serviço fica a cargo do provedor, sendo que o padrão apenas sugere quais serviços as classes devem atender, serviços estes já descritos anteriormente.

Isso dá margem para interpretações diferentes das configurações das classes de serviço. Por exemplo, um provedor A pode definir que a *Minimum reserved traffic rate* para a sua classe UGS é de 1000000 bits por segundo (número fictício, sem nenhum embasamento) enquanto um provedor B pode definir essa configuração para sua classe ertPS.

2.5.1.2 Camada de Transporte

Nos primeiros anos da existência da Internet, os meios de transmissão eram pouco confiáveis quanto a entrega e a ordem de entrega dos pacotes. Isso refletiu-se no projeto dos protocolos da camada de transporte, resultando na criação de protocolos que visavam garantir a ordem e a entrega de pacotes, como o TCP, ou que não ofereciam garantia nenhuma, como o UDP (EXPOSITO; SÉNAC; DIAZ, 2004).

Com a melhoria dos meios de transmissão e o crescimento do uso de aplicativos multimídia e de tempo real através da Internet, o uso de protocolos com garantia total como o

TCP passou a prejudicar, de certo modo, estes aplicativos, pois o controle de congestionamento e a retransmissão de pacotes perdidos pode introduzir atraso e variações de atraso intoleráveis (CAI et al., 2006).

Atualmente, os protocolos de transporte de alto desempenho visam oferecer modos mais eficientes de realizar o controle do congestionamento e do fluxo, considerando aplicativos de tempo real. Protocolos como o *Datagram Congestion Control Protocol* (DCCP) (KOHLER; HANDLEY; FLOYD, 2006) permitem esquemas diferentes de controle de congestionamento. Alguns fazem o cálculo da vazão da janela de congestionamento através de equações (como o *TCP Friendly Rate Control* — TFRC) (FLOYD et al., 2008). Outros fazem este mesmo cálculo através de parâmetros simples para aumentar e diminuir a janela (como o *Additive-Increase, Multiplicative-Decrease* — AIMD). Estes esquemas utilizam-se também de outros parâmetros de desempenho da camada de rede para realizar seu trabalho, a saber: atraso, variação de atraso e perda, já mencionados anteriormente.

QoSTP (EXPOSITO; SÉNAC; DIAZ, 2004) é um exemplo de protocolo de transporte projetado desde o princípio pensando em como fornecer QoS na Internet. Tomando o TFRC como base para o controle de congestionamento, QoSTP expande a heurística do TFRC para fluxos multimídia, levando em conta restrições de tempo e diferenciação de fluxos. QoSTP também permite a escolha do esquema de controle de erro, utilizando como base o esquema *Automatic Repeat reQuest* (ARQ), introduzindo também ao ARQ configurações de restrições de tempo, diferenciação e nesse caso restrições de ordem.

Basicamente, QoSTP permite sua configuração quanto ao controle de congestionamento e controle de erro fornecendo as seguintes possibilidades, que podem ser interpretadas, a grosso modo, como seus parâmetros de qualidade:

- Controle de Congestionamento:
  - TFRC;
  - TFRC com restrições de tempo e diferenciação (TFRC *Time-constrained and Differentiated* ou TD-TFRC).
- Controle de Erro:
  - Parcialmente confiável/Partialmente ordenado (*Partially reliable/Partially ordered*, ou PR/PO);
  - Diferenciado e parcialmente confiável (*Differentiated and Partially reliable*, ou D-PR);
  - Diferenciado, parcialmente confiável e com restrições de tempo (*Time-constrained, Differentiated and Partially reliable*, ou TD-PR).

Combinações destes parâmetros de qualidade, junto a um conjunto de parâmetros de qualidade do nível de rede podem ser utilizados para fornecer qualidade de serviço em uma rede baseada em melhor esforço, como a Internet.

Em (EXPOSITO; SÉNAC; DIAZ, 2004), os autores não fazem menção de classes de serviço padrões, e o mesmo acontece em outros trabalhos da área, como (HSIEH; SIVAKUMAR, 2005) e (CAI et al., 2006). O foco principal de todos estes trabalhos é mostrar que os protocolos propostos podem tratar fluxos diferenciadamente e oferecer QoS para cada um deles, mas não discute como é feita sua classificação, ficando subentendido que poderia ser por tipo de aplicação (por exemplo em fluxos VoIP, VoD).

2.5.1.3 Camada de Aplicação

A camada de aplicação da pilha TCP/IP é a mais abrangente entre todas as camadas. Isto devido a existência de

uma enorme quantidade de aplicativos, desde programas de e-mail a *web services*, passando por VoIP e VoD. Cada um destes aplicativos possui um conjunto específico de parâmetros de desempenho, e tanto enumerar quanto criar um conjunto padrão de parâmetros que englobe todos estes conjuntos seria inviável em uma situação prática.

A título de exemplo, serão apresentados os parâmetros de qualidade do padrão G.711.1, da ITU-T. O G.711 é um padrão da ITU-T para a compressão e codificação de sinais de áudio. Ele utiliza uma taxa de amostragem de 8000 amostras por segundo, ou 8 kHz, e quantifica as amostras utilizando 8 bits, gerando uma taxa de bits de 64 kbits por segundo. A quantificação pode ser feita através de dois algoritmos similares: o μ-*law* e o A-*law* (ITU-T, 1988).

O G.711.1 (ITU-T, 2008), por sua vez, é uma extensão do G.711 para o uso em conexões de banda larga. Por padrão, taxas de amostragem de 16000 amostras por segundo (16 kHz) são utilizadas, mas amostragem de 8 kHz é permitida por questões de compatibilidade.

Quando amostrado em 16 kHz, o codificador trabalha em uma frequência de 50 a 7000 Hz, gerando taxas de bits de 80 e 94 kbits por segundo. Amostrado em 8 kHz, a frequência utilizada é de 50 a 4000 Hz, gerando taxas de 64 kbits/s. O tamanho de bits utilizado na quantificação varia de acordo com configuração do algoritmo de codificação, mas o tamanho das amostras é fixado em 5 milissegundos (ITU-T, 2008).

Pode-se afirmar, portanto, que os parâmetros de qualidade para o padrão G.711.1 são:

Taxa de amostragem, que pode ser 8 kHz ou 16 kHz, e

Taxa de bits, que é igual a 64 kbps no caso da amostragem em 8 kHz e podendo ser 80 ou 94 kbps no caso da amostragem de 16 kHz.

2.5.1.4 Qualidade Percebida

A qualidade percebida corresponde, como já dito anteriormente, à impressão que o usuário tem da qualidade fornecida por um aplicativo. Essa impressão é, portanto, dependente do tipo do aplicativo sendo utilizado: as impressões que um usuário tem de um aplicativo de telefonia são diferentes das que o mesmo usuário pode ter de um aplicativo de e-mail, embora pode-se dizer que a finalidade dos dois aplicativos é basicamente a mesma (transmitir uma mensagem — seja ela texto ou voz — de um usuário para outro).

A medição de qualidade percebida é um trabalho complexo e demorado, pois geralmente envolve a reunião de um grupo estatisticamente significativo de especialistas da área e o teste do aplicativo por estas pessoas, para obtenção das opiniões sobre a qualidade oferecida. Para aplicativos multimídia, esse tipo de avaliação é muito importante, uma vez que a opinião positiva do usuário é o que determina a aceitação do aplicativo.

A título de exemplo, esta seção apresenta a QoE para aplicações de voz. Na área de telefonia, foram criados padrões de realização de testes para a determinação da qualidade da voz, como o P.800, da ITU-T (ITU-T, 1996). O P.800 envolve a reunião de especialistas em qualidade de áudio e a avaliação da qualidade da voz do lado receptor para um aplicativo de telefonia. Essa avaliação compreende uma nota de 1 a 5 sendo que, quanto menor a nota, pior a qualidade percebida. A qualidade oferecida pelo aplicativo será a média das notas de todos os especialistas envolvidos no teste. Esse sistema de graduação da qualidade do áudio é também chamado de *Mean Opinion Score* (MOS).

Por causa do tempo e do esforço aplicado na realização destes testes subjetivos, foram padronizados também meios de mapear parâmetros de qualidade de rede e de aplicação em parâmetros subjetivos como o MOS. O E-model (padrão

ITU-T G.107) (ITU-T, 2009) propõe um cálculo objetivo da qualidade oferecida por um aplicativo de transmissão de voz, utilizando parâmetros objetivos da camada de aplicação (como razão sinal-ruído) e parâmetros de rede (como atraso).

Considerando então a MOS para um aplicativo de transmissão de voz, podemos considerar a existência dos seguintes parâmetros de qualidade (ITU-T, 1996):

MOS 5: qualidade excelente (sem sinais perceptíveis de degradação);

MOS 4: qualidade boa (com poucos sinais de degradação perceptível, que não prejudicam a comunicação);

MOS 3: qualidade razoável (a degradação do sinal começa a prejudicar a comunicação);

MOS 2: qualidade pobre (a degradação do sinal prejudica muito a comunicação);

MOS 1: qualidade ruim (a degradação do sinal impede a comunicação)

Existem padrões para determinação da qualidade percebida em outros tipos de aplicativos também, como transmissão de vídeo (ITU-R, 2002), por exemplo.

2.6 CONSIDERAÇÕES FINAIS

Todos os padrões e estudos de parametrização da qualidade nas diversas camadas da pilha TCP/IP apresentados são exemplos significativos de como a QoS é importante para o provimento de serviços de rede.

A situação ideal para o provimento de QoS, considerando a existência de tantos padrões de especificação de qualidade, seria a existência de uma linguagem comum de especificação,

que permitisse o mapeamento direto entre especificações de camadas diferentes. A proposta desta dissertação é justamente a criação dessa linguagem comum, definida utilizando recursos de semântica computacional, descritos no próximo capítulo.

3 SEMÂNTICA

Desde a criação e da consequente popularização da Internet e da *World Wide Web*, o volume de informação disponível na rede torna-se cada vez maior a cada ano. Os usuários da Internet passam a ter acesso a um imenso aglomerado de dados, o que causa uma "sobrecarga de informação" e dificulta o gerenciamento e a habilidade de tirar conclusões importantes a partir de tanta informação manualmente.

Esta dificuldade de gerenciamento aumentou ainda mais com a atualização das tecnologias de desenvolvimento de sistemas computacionais, que fez surgir a necessidade de lidar com informação legada, e com a tercerização de serviços que fez surgir a necessidade de interoperabilidade de dados entre provedores (DAVIES; STUDER; WARREN, 2006).

Fizeram-se necessários então meios de automatizar o gerenciamento e a filtragem dos dados disponíveis na Web, não somente para poder compreendê-los melhor, mas para utilizá-los de forma eficiente. Para gerenciar toda esta informação eficientemente, é preciso descrevê-la de maneira a torná-la compreensível por sistemas computacionais. Esse é um dos papéis da semântica: permitir a descrição formal de informação de modo que esta possa ser gerenciada automaticamente ou semi-automaticamente por computadores.

A descrição formal de documentos ou mesmo dos conceitos presentes em documentos tem uma grande variedade de aplicações: é possível organizar e buscar informação com base em seu significado; estabelecer equivalências entre conceitos sintaticamente distintos, permitindo a criação de mapeamentos entre terminologias de provedores diferentes (DAVIES; STUDER; WARREN, 2006).

Um dos meios de criar estas descrições semânticas é o uso de Ontologias. Inspiradas no conceito homônimo da Filosofia e baseadas em formas de Lógica Computacional, as ontologias permitem embutir significado em informações através de

metadados que são compreensíveis por sistemas computacionais.

O objetivo deste capítulo é apresentar os principais conceitos envolvidos na definição de ontologias, para que o leitor possa compreender a apresentação e a crítica dos trabalhos relacionados, além de facilitar a compreensão das decisões de modelagem tomadas na criação de NetQoSOnt, a ontologia proposta neste trabalho.

3.1 ONTOLOGIAS E OWL

Em computação, uma Ontologia pode ser definida como "uma especificação explícita e formal da conceitualização de um domínio de interesse" (DAVIES; STUDER; WARREN, 2006). Formal, nesse sentido, significa que esta especificação permite o processamento automático por computadores. Ontologias são formadas por conceitos, também chamados classes, que são representações da informação modelada. Um conceito pode ter uma ou mais instâncias (*e.g.* o conceito de País pode ter uma instância chamada Brasil), também chamadas de indivíduos. Indivíduos podem ter relações com outros indivíduos, ou com tipos primitivos (como strings ou números). Estas relações também são chamadas de propriedades.

A Web Ontology Language (OWL) é a linguagem padrão da W3C para a codificação de ontologias (W3C, 2004a). Derivada do Resource Description Framework (RDF), e serializada em XML, OWL provê construções para criar tudo o que foi citado anteriormente: classes, instâncias, relações entre classes ou instâncias, chamadas propriedades de objeto (*object properties*), relações entre classes/instâncias e tipos primitivos, chamadas propriedades de tipos de dados (*datatype properties*) e axiomas — utilizados para definir o domínio (*domain*) e o alcance (*range*) das propriedades, por exemplo. Todas estas construções podem ser denominadas

genericamente de recursos (do inglês *resource*), terminologia também usada em RDF.

OWL possui três sublinguagens. OWL Lite é a mais restrita, recomendada para usuários que precisam apenas de uma hierarquia de classificação e restrições simples, e é computacionalmente decidível. OWL DL é um superconjunto de OWL Lite, baseado em um tipo de lógica de primeira ordem chamado Lógica Descritiva. OWL DL permite maior expressividade sem comprometer a computabilidade. Finalmente, OWL Full é um superconjunto de OWL DL, que permite máxima expressividade, mas potencialmente sacrificando a decidibilidade da ontologia descrita (W3C, 2004a).

Existem duas versões de OWL sendo utilizadas atualmente. OWL 1.0 é o padrão atual, recomendação oficial da W3C. OWL 2.0 é uma atualização de OWL 1.0, atualmente no *status* de Recomendação Candidata. OWL 2.0 traz novas funcionalidades (muitas delas utilizadas nesta dissertação), e já é implementada em várias ferramentas que trabalham com ontologias, tais como os motores de inferência FaCT++ (TSARKOV; HORROCKS, 2009), Pellet (C&P, 2009), APIs como a OWL API (MANCHESTER U., 2009) e editores gráficos como o Protégé (BMIR, 2009).

Em OWL, ontologias estão sujeitas à chamada pressuposição de mundo aberto (*Open World Assumption*). Esta pressuposição define que se uma declaração não pode ser considerada verdadeira utilizando o conhecimento existente na ontologia, esta declaração não pode ser considerada falsa. Uma consequência disso é que conceitos ou indivíduos diferentes devem ser explicitamente declarados como tal. Por esta razão, OWL também provê axiomas para declarar conceitos como disjuntos, e indivíduos ou propriedades como diferentes. Outra funcionalidade importante de OWL que diz respeito à diferença entre recursos é que não existe suposição de nomes únicos (do inglês *Unique Name Assumption*,

referido também pela sigla UNA): dois ou mais recursos diferentes podem ter o mesmo nome através de ontologias diferentes, e isso não significa que são a mesma coisa. Na verdade, o que identifica um recurso em uma ontologia é seu URI, composta pelo URI base da ontologia mais o URI do recurso. Esse sistema de identificação foi herdado de RDF, assim como vários outros recursos de metalinguagem (rótulos, comentários, entre outros).

Uma das principais vantagens da utilização de ontologias é que nem todo o conhecimento precisa ser declarado explicitamente; novas relações entre conceitos e instâncias podem ser inferidas por um programa chamado motor de inferência (*reasoner*). Basicamente, a dedução feita por um motor de inferência sobre uma ontologia constrói uma hierarquia. Baseado nas propriedades de objeto e nas propriedades de tipos de dados de uma classe, um motor pode inferir se esta é mais especializada, mais geral, ou equivalente a outras classes. Desse modo, subsunções (relações de herança) e equivalências que não foram declaradas explicitamente são computadas. Também se baseando nas propriedades de um indivíduo, um motor pode inferir se ele pode ser considerado uma instância de outro conceito.

3.1.1 Funcionalidades de OWL

Como dito anteriormente, OWL provê várias construções características de uma ontologia computacional. O objetivo desta seção é destacar as principais construções de OWL DL, a sublinguagem de OWL utilizada neste trabalho, de modo que o leitor possa compreender a modelagem e os recursos utilizados. O objetivo aqui não é cobrir todas as construções extensivamente.

Neste capítulo, serão usados recursos gráficos de UML, tais como diagramas de classes e de objetos, para ilustrar as construções possíveis em OWL 2.0. Algumas pequenas

mudanças na sintaxe dos diagramas foram feitas, e serão explicitadas na medida em que aparecem. Alguns exemplos dispostos nessa seção foram emprestados de (HORRIDGE et al., 2009).

3.1.1.1 Classes

Classes representam os conceitos de um domínio de interesse. Por exemplo, a classe Veículo pode representar o conjunto de todos os veículos automotivos em uma ontologia sobre automóveis. As classes de uma ontologia podem formar uma hierarquia, com subclasses e superclasses, também chamada de taxonomia (HORRIDGE et al., 2009). Ainda na ontologia de automóveis, a classe Veículo pode ter como subclasses Carro, Caminhão e Motocicleta, por exemplo. Nas figuras 3.1 e 3.2, podemos visualizar a hierarquia de classes descrita, e o código OWL para as classes Veículo e Carro.

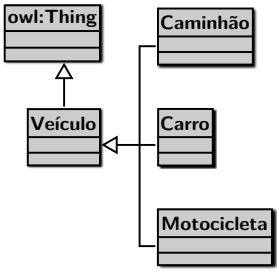

Figura 3.1: Hierarquia de classes OWL.

Nestas figuras, podemos ver que a subsunção entre classes é na verdade um axioma, descrito pelo código `rdfs:subClassOf`. Podemos ver também uma convenção da linguagem OWL: a classe owl:Thing é a superclasse de todas as classes em qualquer ontologia codificada em OWL. Na figura 3.1, podemos notar uma mudança na notação de UML: a notação *NomeDoObjeto : NomeDaClasse* é geralmente utilizada em diagramas de objetos para identificar um objeto e sua classe.

```
<owl:Class
  rdf:about="http://www.w3.org/2002/07/owl#Thing"/>

<owl:Class rdf:about="#Ve&#237;culo">
  <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<owl:Class rdf:about="#Carro">
  <rdfs:subClassOf rdf:resource="#Ve&#237;culo"/>
</owl:Class>

<owl:Class rdf:about="#Pessoa">
  <rdfs:subClassOf rdf:resource="&owl;Thing"/>
  <owl:disjointWith rdf:resource="#Ve&#237;culo"/>
</owl:Class>
```

Figura 3.2: Código OWL para as classes Veículo e Carro

Na figura 3.1, a notação *owl:Thing* na verdade representa uma abreviatura do URI da classe Thing de OWL. Portanto, *owl:* é uma abreviação de "http://www.w3.org/-2002/07/owl#". Note que não são usados espaços entre as palavras *owl*, *:* e *Thing*. Nos diagramas de classes e objetos utilizados nessa dissertação, a presença de *:* em um nome de classe composto de duas palavras e a ausência de espaços entre essas palavras indicam uma abreviação de URI (na verdade, a presença desta característica em qualquer nome dentro desta dissertação indica isso). Para enfatizar ainda mais essa diferença, em diagramas de objetos os nomes de classes com essa característica serão italizados.

Outro aspecto que deve ser frisado na figura 3.2, é a maneira como as classes são identificadas (mas, como veremos, esta construção se repete para outros recursos — indivíduos, propriedades — de OWL). O código `rdf:about=` não declara o nome da classe, declara na verdade um identificador para a mesma. Os identificadores fazem referência a uma URI, o que permite que dois recursos de ontologias diferentes possuam o mesmo "nome", ou mesmo que dois ou mais recursos de nomes distintos em uma mesma ontologia refiram-se ao mesmo recurso. O que diferencia realmente um recurso de

outro é o identificador.

Outro axioma importante, que também deve ser mencionado, é o de *disjunção*. Conforme descrito anteriormente, OWL assume preposição de mundo aberto, o que obriga ao usuário explicitar quando duas classes são de fato distintas. Pessoas e Veículos são duas coisas distintas, logo podemos explicitar isso na ontologia utilizando o código `owl:disjointWith`.

3.1.1.2 Indivíduos

Indivíduos representam concretizações, instâncias dos conceitos. Comparando rapidamente com Teoria de Conjuntos, Classes são conjuntos, enquanto Indivíduos são elementos dos conjuntos. No exemplo da ontologia de automóveis, a seguinte modelagem é possível: o invidíduo ToyotaHilux pode pertencer à classe Carro e o indivíduo KawasakiNinja pode pertenter à classe Motocicleta. A figura 3.3 mostra graficamente os indivíduos descritos e a figura 3.4 mostra o respectivo código OWL.

Figura 3.3: Indivíduos ToyotaHilux e KawasakiNinja.

```
<Carro rdf:about="#ToyotaHilux"/>

<Motocicleta rdf:about="#KawasakiNinja">
  <owl:differentFrom rdf:resource="#ToyotaHilux"/>
</Motocicleta>
```

Figura 3.4: Código OWL para os indivíduos ToyotaHilux e KawasakiNinja.

Assim como devemos explicitar quando queremos dizer que duas classes são distintas, podemos fazer o mesmo com indivíduos, utilizando o axioma `owl:differentFrom`.

3.1.1.3 Propriedades

Representam relações binárias entre classes ou entre indivíduos em uma ontologia. Por exemplo, `pertenceAPessoa` pode ser uma propriedade da classe Veículo na ontologia de automóveis. Toda propriedade pode ter um inverso. A propriedade possuiVeículo pode ser o inverso da propriedade pertenceAPessoa. A figura 3.5 mostra graficamente as propriedades descritas e a figura 3.6 mostra seu código OWL.

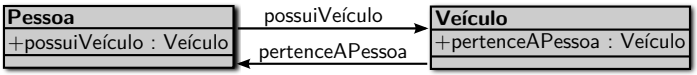

Figura 3.5: Propriedades pertenceAPessoa e possuiVeículo.

```
<owl:ObjectProperty rdf:about="#pertenceAPessoa">
  <owl:inverseOf rdf:resource="#possuiVe&#237;culo"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#possuiVe&#237;culo"/>
```

Figura 3.6: Código OWL para as propriedades pertenceAPessoa e possuiVeículo.

Propriedades também podem possuir características específicas que relacionam indivíduos ou classes, ao contrário do axioma de propriedade inversa que relaciona duas propriedades.

Se um indivíduo está relacionado a outro através de uma propriedade *funcional*, somente este indivíduo pode estar relacionado ao primeiro através desta propriedade. Um exemplo seria uma propriedade temMãeBiológica (HORRIDGE et al., 2009), mostrado na figura 3.7.

Um indivíduo pode ter somente uma mãe biológica. Logo, podemos codificar isso em uma ontologia através de uma propriedade funcional. Se por acaso um indivíduo estiver

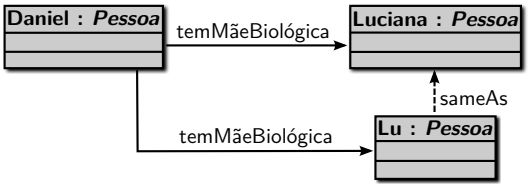

Figura 3.7: Exemplo de propriedade funcional.

relacionado a dois outros indivíduos através da mesma propriedade funcional, isso implicará que os dois indivíduos são na verdade o mesmo. Na figura 3.7, a equalidade entre indivíduos é nomeada sameAs ("mesmo que"), que é o mesmo nome do código OWL correspondente. OWL também possibilita a existência de propriedades *inversamente funcionais*. Considere o inverso da propriedade temMãeBiológica, éMãeBiológicaDe (HORRIDGE et al., 2009): se dois indivíduos estiverem relacionados a um primeiro através de uma mesma propriedade inversamente funcional, isso implicará que os dois indivíduos são na verdade o mesmo.

Uma propriedade *transitiva* implica que, se um indivíduo a está relacionado a um indivíduo b através de uma propriedade transitiva P, e o indivíduo b está relacionado a um indivíduo c através desta mesma propriedade, o indivíduo a estará relacionado ao indivíduo c através da propriedade P.

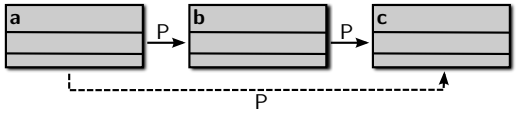

Figura 3.8: Propriedade transitiva.

Uma propriedade *simétrica* implica que, se um indivíduo a está relacionado ao indivíduo b através de uma propriedade simétrica, o indivíduo b também estará relacionado ao indivíduo a através desta propriedade. Já uma propriedade *antisimétrica* implica que, se um indivíduo a está relacionado ao indivíduo b através de uma propriedade antisimétrica, o

indivíduo b **nunca** poderá estar relacionado ao indivíduo a através desta propriedade.

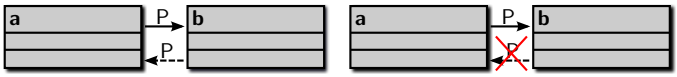

Figura 3.9: Propriedades simétrica e anti-simétrica, respectivamente.

Uma propriedade *reflexiva* é usada para indicar que o indivíduo está relacionado a ele mesmo através desta propriedade. Uma propriedade *irreflexiva* significa que, se um indivíduo está relacionado a outro através de uma propriedade irreflexiva, os indivíduos deverão ser diferentes.

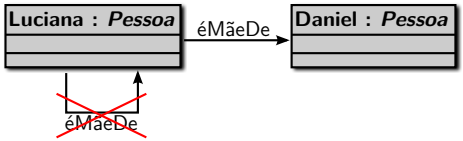

Figura 3.10: Propriedade irreflexiva.

Um exemplo de propriedade irreflexiva é a propriedade éMãeDe (HORRIDGE et al., 2009). Nesse caso, um indivíduo não pode ser mãe dele mesmo, logo isso é codificado na ontologia através de uma propriedade irreflexiva.

Propriedades podem ter restrições de classe, codificadas através dos axiomas de *domínio* e de *alcance*, que são melhor explicados diretamente através de exemplos. Considere a propriedade possuiVeículo da ontologia de automóveis. Dizer que esta propriedade possui como domínio a classe Pessoa implica que apenas indivíduos desta classe podem estar relacionados com indivíduos da classe veículo, ou seja, apenas pessoas podem possuir veículos. Se um indivíduo Daniel não pertence diretamente à classe Pessoa, mas possui esta propriedade, um motor de inferência concluiria que este Daniel é uma pessoa.

Declarar que o alcance da propriedade possuiVeículo é a classe Veículo, implica que apenas indivíduos da classe Veículo podem estar relacionados à indivíduos da classe Pessoa através desta propriedade (ainda considerando que o domínio de possuiVeículo é Pessoa). Um indivíduo de qualquer outra classe disjunta de Veículo resultaria em uma inconsistência.

```
<owl:ObjectProperty rdf:about="#possuiVe&#237;culo">
    <rdfs:domain rdf:resource="#Pessoa"/>
    <rdfs:range rdf:resource="#Ve&#237;culo"/>
</owl:ObjectProperty>
```

Figura 3.11: Definindo o domínio e o alcance de possuiVeículo.

Até o momento, foram apresentadas apenas propriedades de objeto, que associam classes ou indivíduos a outros indivíduos. As **propriedades de tipos de dados**, como já dito anteriormente, associam classes ou indivíduos a tipos de dados primitivos, como strings ou números. Propriedades de tipos de dados ligam indivíduos a qualquer tipo de dados suportado por XML Schema ou a qualquer literal RDF. Um exemplo simples desse tipo de propriedade resulta na seguinte modelagem: todo veículo possui um ano de fabricação, e este ano é simplesmente um número inteiro. Logo, podemos criar uma propriedade de tipo de dados possuiAnoDeFabricação, e declarar que seu alcance é um número inteiro. Por motivos óbvios, uma propriedade de tipos de dados só pode ter a característica de ser funcional, se necessário.

```
<owl:DatatypeProperty
  rdf:about="#possuiAnoDeFabrica&#231;&#227;o">
    <rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>
```

Figura 3.12: Definindo a propriedade de tipo de dados possuiAnoDeFabricação.

Em OWL, podemos definir classes que não possuem ne-

nhuma propriedade. No entanto, estas definições acabam não sendo úteis para a maioria dos domínios. Para que as ontologias façam alguma computação útil, devemos associar as propriedades às classes. Alguns dos tipos de associação são mostrados a seguir.

3.1.2 Padrões de Modelagem OWL

Associar as propriedades às classes significa restringir a definição destas classes. É possível restringir propriedades quanto a sua quantificação, quanto a sua cardinalidade e quanto ao seu valor.

3.1.2.1 Restrições de Quantificação: Restrições Existenciais e Universais

Uma Restrição Existencial descreve indivíduos que participam em *pelo menos um* relacionamento com outros indivíduos através de uma dada propriedade. Suponha que, no exemplo da ontologia de automóveis, queremos declarar que um Veículo possui rodas. Para isso, associaremos a propriedade possuiRoda com a classe Veículo utilizando uma Restrição Existencial.

```
<owl:Class rdf:about="#Ve&#237;culo">
  <rdfs:subClassOf rdf:resource="&owl;Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#possuiRoda"/>
      <owl:someValuesFrom rdf:resource="#Roda"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figura 3.13: Redefinindo a classe Veículo.

Uma Restrição Universal descreve indivíduos que participam em uma relação *somente* com um tipo de indivíduo

através de uma dada propriedade. Voltando à ontologia de automóveis, podemos descrever um veículo super seguro cujo *único* tipo de lataria é blindada. Para isso, criamos as classes VeículoSuperSeguro, Lataria com uma subclasse LatariaBlindada, uma propriedade de objeto possuiLataria, e relacionamos VeículoSuperSeguro com LatariaBlindada criando uma Restrição Universal.

```
<owl:Class rdf:about="#Ve&#237;culoSuperSeguro">
  <rdfs:subClassOf rdf:resource="#Ve&#237;culo"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#possuiLataria"/>
       <owl:allValuesFrom rdf:resource="#LatariaBlindada"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figura 3.14: Definindo a classe VeículoSuperSeguro.

3.1.2.2 Restrições de Cardinalidade

Restrições de Cardinalidade limitam a quantidade de vezes que um indivíduo pode participar em uma relação com outros indivíduos. Por exemplo, na ontologia de automóveis é possível modelar um tipo de carro denominado CarroNormal que possui exatamente 4 rodas.

Modelando CarroNormal dessa forma, queremos declarar que, para um indivíduo ser considerado um CarroNormal, ele **deve** partipar em **exatamente** 4 relacionamentos com indivíduos diferentes da classe Roda. Se um indivíduo dessa classe não tiver 4 rodas (seja menos ou mais), haverá uma inconsistência na ontologia. Os códigos para declarar o mínimo e o máximo de cardinalidade são `owl:minCardinality` e `owl:maxCardinality`, respectivamente.

```
<owl:Class rdf:about="#CarroNormal">
  <rdfs:subClassOf rdf:resource="#Carro"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#possuiRoda"/>
      <owl:onClass rdf:resource="#Roda"/>
        <owl:cardinality
          rdf:datatype="&xsd;nonNegativeInteger">
            4
        </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figura 3.15: Definindo a classe CarroNormal.

3.1.2.3 Restrições de Valor

Restrições de Valor limitam exatamente qual indivíduo (ou tipo de dado) pode participar em uma relação. Por exemplo, é possível modelar a subclasse de carro Toyota, e declarar que Toyota é um tipo de carro que tem como país de origem o Japão, sendo que Japão é um indivíduo da classe País.

```
<owl:Class rdf:about="#Toyota">
  <rdfs:subClassOf rdf:resource="#Carro"/>
  <rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty
          rdf:resource="#possuiPa&#237;sDeOrigem"/>
            <owl:hasValue rdf:resource="#Jap&#227;o"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figura 3.16: Definindo a classe Toyota.

É importante mencionar que esse tipo de modelagem não seria possível em OWL 1, uma vez que essa versão não permite que *indivíduos* sejam diretamente relacionados a *classes*. Em OWL 1, somente tipos de dados podem ser diretamente relacionados a classes.

A possibilidade de relacionar indivíduos a classes permite modelar, dentre várias coisas, restrições de unidades de

medida. É possível declarar, por exemplo, que o peso de um indivíduo da classe Pessoa é medido especificamente em kilogramas, sendo que kilograma é um indivíduo que representa uma unidade de medida. Em OWL 1, seria necessário relacionar a classe Pessoa com uma classe UnidadeDePesagem (considerando que kilograma seja um indivíduo dessa classe), e depois relacionar indivíduos da classe Pessoa com o indivíduo kilograma. Resumindo, não seria possível restringir especificamente a unidade de pesagem no conceito de Pessoa.

Outra funcionalidade importante de OWL 2, não existente em OWL 1, e que foi usada na proposta desta dissertação é a funcionalidade de modelar *limites* (do inglês *ranges*). A versão 2 de OWL tomou emprestada essa sintaxe de RDF Schema que permite declarar que uma classe pode estar associada são somente a um tipo de dado, mas a um intervalo ou limite de dados. Por exemplo, podemos definir a classe Adulto como sendo uma subclasse de Pessoa cuja idade é *maior ou igual* a 18. Este exemplo é descrito na figura 3.17.

Analisando a figura 3.17, é possível perceber que o código OWL `xsd:minInclusive` foi utilizado para especificar o limite. De maneira análoga, é possível utilizar `maxInclusive`, `maxExclusive` e `minExclusive` para definir intervalos *menores ou iguais a*, *menores que* e *maiores que* um certo valor, respectivamente.

3.1.2.4 Classes Primitivas e Classes Definidas

Todas as classes descritas até o momento são consideradas classes *primitivas*. As propriedades associadas a classes primitivas impõem condições *necessárias* para um indivíduo fazer parte da classe. Por exemplo, do modo como foi definida a classe CarroNormal, a restrição de cardinalidade imposta na propriedade possuiRoda foi declarada como uma

```
<owl:Class rdf:about="#Adulto">
  <rdfs:subClassOf rdf:resource="#Pessoa"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#idade"/>
      <owl:someValuesFrom>
        <rdf:Description>
          <rdf:type rdf:resource="&rdfs;Datatype"/>
          <owl:onDatatype rdf:resource="&xsd;int"/>
          <owl:withRestrictions rdf:parseType="Collection">
            <rdf:Description>
              <xsd:minInclusive
                rdf:datatype="&xsd;integer">
                  18
              </xsd:minInclusive>
            </rdf:Description>
          </owl:withRestrictions>
        </rdf:Description>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figura 3.17: Definindo a classe Adulto.

condição necessária, ou seja, para que um indivíduo seja considerado um CarroNormal ele deve ter 4 Rodas, mas o fato de um indivíduo ter 4 Rodas não significa que ele seja um CarroNormal.

Uma classe *definida* possui condições que são consideradas não somente necessárias, mas *suficientes* para que um indivíduo faça parte da classe. Podemos redefinir a classe CarroNormal de modo a torná-la uma classe Definida.

Embora seja um exemplo forçado do ponto de vista semântico, ilustra a descrição de uma classe Definida. Agora, um CarroNormal é todo indivíduo da classe Carro que possua exatamente 4 relacionamentos com indivíduos da classe Roda, ou seja, um carro com 4 rodas.

Como é possível notar observando o código das figuras 3.15 e 3.18, uma classe primitiva é definida utilizando o código OWL `owl:subClass` e uma classe primitiva usa o código `owl:equivalentClass`. Estes códigos, como é possível deduzir, são os mesmos utilizados para definir os axiomas de

```
<owl:Class rdf:about="#CarroNormal">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="#Carro"/>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#possuiRoda"/>
            <owl:onClass rdf:resource="#Roda"/>
              <owl:cardinality
                rdf:datatype="&xsd;nonNegativeInteger">
                  4
              </owl:cardinality>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Figura 3.18: Redefinindo a classe CarroNormal.

subclasse (relação de subsunção) e classe equivalente (duas classes que englobam os mesmos indivíduos).

3.1.3 Sobre o Motor de Inferência

Uma das grandes vantagens do uso de ontologias está no uso de motores de inferência. Com eles, é possível assegurar a validade dos conceitos e relações criados em um arquivo OWL. Foi mencionado em várias partes deste capítulo que o uso incorreto da sintaxe e da semântica das construções de OWL gera uma *inconsistência*.

Inconsistências são capturadas pelo motor de inferência, e geralmente constituem erros de modelagem que devem ser consertados. Vários exemplos de inconsistências foram dados ao longo deste capítulo.

3.1.4 Inferência Utilizando Regras

Como já dito anteriormente, a inferência na sublinguagem de OWL mais utilizada para a construção de ontologias, OWL DL, é baseada em Lógica Descritiva. Com o objetivo

de fornecer uma alternativa ao uso de Lógica Descritiva na construção de ontologias, foi proposto o uso de Lógica de Horn, através da *Semantic Web Rule Language* ou SWRL (W3C, 2004b), que combina OWL e RuleML, uma linguagem de marcação também baseada em RDF para expressar regras do tipo Horn. SWRL fornece tanto uma representação em RDF das regras quanto uma linguagem de alto nível para facilitar a criação de novas regras.

Regras do tipo Horn possuem um antecedente e um consequente. Se o antecedente for provado verdade, significa que o consequente também deve ser. Motores de regras percorrem uma ontologia tentando descobrir quais antecedentes de regras são verdade de acordo com o conhecimento na ontologia. Quando o antecedente de uma regra é provado verdade, o motor insere as relações presentes no consequente, criando novas relações entre os recursos da ontologia.

Um exemplo simples de regra SWRL é mostrado na figura 3.19. Nesse exemplo, Pai e Avo são classes OWL, e ?x, ?y e ?z são variáveis do corpo da regra.

```
Pai(?x, ?y) ^ Pai(?y, ?z) ^ Diferente(?x, ?y) -> Avo(?x, ?z)
```

Figura 3.19: Exemplo de regra SWRL.

SWRL é uma linguagem extensível, que provê todas as suas funcionalidades através dos chamados *built-ins*. Existe um conjunto de *built-ins* padrão, previsto na proposta da linguagem, que fornece sintaxe para operações matemáticas, comparações lógicas, tratamento de *strings*, entre outros. Há a possibilidade também de criar novos *built-ins*, inserindo novas funcionalidades na linguagem. A comunidade de desenvolvedores da OWL API 1 e do Protégé 3.* destacaram-se no desenvolvimento de novos *built-ins*, criando suporte para criação de novos indivíduos em ontologias, por exemplo. É

importante ressaltar que os *built-ins* padrões são considerados *DL-safe*, o que significa que os consequentes das regras não criam novos recursos nas ontologias, apenas acrescentam novas relações, uma operação considerada computável. O exemplo de *built-ins* criado pela comunidade é potencialmente não computável, uma vez que regras mal formuladas podem levar a repetições infinitas.

Existem algumas opções de motores de regras SWRL, ficando em destaque o Motor de Inferência Pellet (C&P, 2009), que além de calcular inferência em OWL também possui suporte para regras SWRL.

3.1.5 Formas de Consulta do Conhecimento em Ontologias

Além da OWL API, que fornece uma interface de programação para consultar e utilizar o conhecimento contido em ontologias OWL, existem outras formas de consulta a ontologias. Uma das formas mais conhecidas e utilizadas é a linguagem SPARQL (W3C, 2008). Proposto para realizar consultas em grafos RDF, SPARQL pode ser utilizada em qualquer derivação de RDF, como por exemplo OWL.

O padrão SPARQL é composto de uma linguagem de consulta, um protocolo de consulta, e um formato de documento XML para reproduzir os resultados. A grande vantagem do uso de SPARQL é a sintaxe da linguagem de consulta, bastante familiar a SQL, com a excessão do uso de triplas RDF.

Existem várias implementações de SPARQL, ficando novamente em destaque o Motor de Inferência Pellet, que também possui suporte para esta linguagem.

3.2 CONSIDERAÇÕES FINAIS

As várias funcionalidades de OWL descritas neste capítulo, aliadas aos padrões de modelagem e ao uso de moto-

res de inferência são vitais para a ontologia proposta nesta dissertação. A influência destes fatores será vista em mais detalhes no capítulo 5.

4 ONTOLOGIAS DE QOS

O objetivo desta seção é apresentar alguns trabalhos já existentes na área de semântica para a construção de ontologias de QoS, e esclarecer porque eles não foram utilizados como base para este trabalho.

4.1 ONTOLOGIAS DE QOS PARA REDES

4.1.1 Network Service Specification Ontology

(ALÍPIO; NEVES; CARVALHO, 2007) propõe uma ontologia para a especificação de serviços de rede. Para codificar esta ontologia, foi utilizada a linguagem de especificação de ontologias *Frame Logic* (também chamada de F-Logic ou FL). FL é uma linguagem de especificação cuja base possui uma forte correlação com o paradigma de orientação a objetos, incluindo conceitos como herança, polimorfismo, encapsulamento, entre outros (KIFER; LAUSEN; WU, 1995). A implementação de FL utilizada se chama FLORA-2 (FLORA-2, 2009) e constitui um dialeto de FL que suporta inferência através de regras.

Os conceitos e a terminologia utilizados nesse trabalho vêm todos do padrão DiffServ. A ontologia possui vários conceitos conceitos para a criação de SLSs, incluindo:

classifier: utilizado para declarar as características que classificam um tipo de tráfego;

scope: utilizado para declarar os nodos de ingresso e egresso de um domínio;

policer: utilizado para declarar as políticas de tratamento do tráfego de uma rede, como marcação, tipos de filas e descarte de tráfego;

metrics: utilizado para declarar a qualidade que o SLS pode prover.

Estes conceitos são utilizados pelo conceito service, que caracteriza um SLS. São fornecidos conceitos padrão especificando cada uma das classes de serviço previstas por DiffServ, junto a sua configuração (políticas e métricas) correspondentes.

Com a excessão de metrics, todos os conceitos citados estão especificamente dentro do contexto de DiffServ. Observando metrics em mais detalhes, descobre-se que ele possui quatro atributos:

throughput: corresponde a um valor inteiro;

loss: instância da classe qualitativeValues;

delay: instância da classe qualitativeValues;

jitter: instância da classe qualitativeValues.

A especificação da classe qualitativeValues não é detalhada em (ALÍPIO; NEVES; CARVALHO, 2007), mencionada apenas como um conceito utilizado para definir valores qualitativos.

As regras de FLORA-2 são utilizadas para validação da consistência da ontologia, através verificação da cardinalidade dos conceitos (por exemplo: para uma instância da classe metrics, somente a declaração do valor de throughput é obrigatória), para a comparação de configurações personalizadas com as classes padrão de DiffServ, e para realizar consultas na base de conhecimento.

Nesta ontologia, criar uma especificação de QoS implica em criar instâncias do conceito metrics, junto a instâncias de qualitativeValues referentes aos seus atributos (estas na verdade opcionais). Estas instâncias seriam por sua vez associadas a instâncias de service. Embora não mencionado em

(ALÍPIO; NEVES; CARVALHO, 2007), é possivel que as regras de inferência de FL possam ser utilizadas para comparar instâncias de service quanto aos seus parâmetros de qualidade.

Embora FL possua muitos recursos interessantes, principalmente a similaridade com o paradigma OO, a especificação não possui força como um padrão. Não existe uma organização que a mantenha, atualizando, inserindo novas funcionalidades e certificando implementações.

Como consequência disso, as várias implementações existentes de FL são por vezes incompatíveis. Embora a implementação de FLORA-2 possua um sistema sólido de regras, dentre várias outras funcionalidades, não existe a garantia de que essas regras poderão funcionar em outra implementação de FL. FLORA-2 não lança nenhuma nova versão estável desde setembro de 2007 (FLORA-2, 2009).

Todos estes problemas, aliados a forte orientação da ontologia proposta ao padrão DiffServ, pesam contra a utilização desta ontologia em outros contextos, como especificação de QoE por exemplo.

Para permitir a criação de especificações de QoS mais flexíveis, esta ontologia precisaria de um conceito mais genérico de métrica, que permitisse a criação de outros tipos de métricas de qualidade, envolvendo outras camadas da pilha TCP/IP. Uma especificação de QoS completa pode envolver outros parâmetros de qualidade além dos presentes na camada de rede. Por esses motivos, esta ontologia foi descartada para ser utilizada como base para esta dissertação.

4.1.2 MonONTO

(MORAES et al., 2008) propõe uma ontologia para auxiliar na tarefa de monitoramento do tráfego de rede e na recomendação de configurações de rede que assegurem o melhor desempenho de um aplicativo de interação multimídia qualquer para usuários não especialistas.

A ontologia é utilizada para a criação de uma base de conhecimento através da coleta de dados realizados por ferramentas de monitoramento de rede. Utilizando os dados coletados, informações sobre os requisitos de desempenho de rede de um dado aplicativo e as preferências do usuário, um sistema especialista é usado para inferir qual é a melhor configuração de rede para este usuário.

A ontologia proposta é codificada utilizando OWL 1. A figura 4.1, extraída de (MORAES et al., 2008), mostra os principais conceitos e relações de MonONTO:

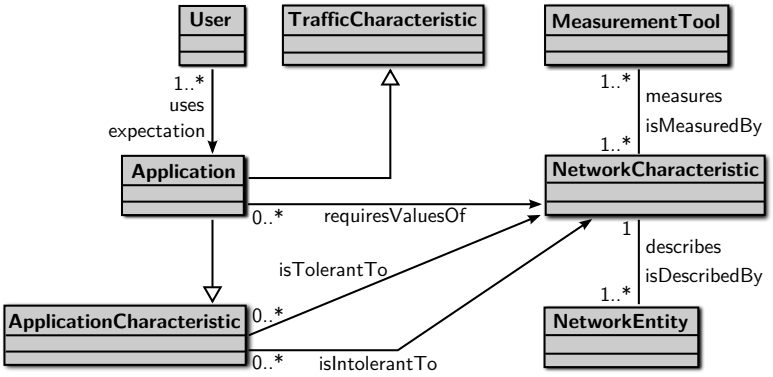

Figura 4.1: Principais conceitos e relações de MonONTO (MORAES et al., 2008).

O conceito User é utilizado para designar o usuário em questão. O conceito MeasurementTool define as ferramentas de monitoramento utilizadas para reunir informações sobre o tráfego da rede (parâmetros de desempenho como medições de atraso, jitter e vazão), informações estas definidas pelo conceito NetworkCharacteristic. Estas informações são ligadas a nós da rede, representados pelo conceito NetworkEntity.

Um aplicativo, representado pelo conceito Application, possui um conjunto de funcionalidades, representadas por

ApplicationCharacteristics, que podem ou não ser influenciadas pelo tráfego da rede, ou seja, por NetworkCharacteristic. TrafficCharacteristic, estranhamente, não é detalhada em (MORAES et al., 2008).

O sistema especialista utiliza regras SWRL integradas a base de conhecimento para inferir sobre os requisitos de rede de uma determinada aplicação, e se eles são atendidos pela rede monitorada. Através destas regras, um administrador de rede consegue listar de quais ferramentas de monitoramento são necessários dados para relacionar com os requisitos de rede de um aplicativo. Uma vez feita a reunião dos dados de monitoramento, é possível para um usuário final descobrir se a rede em que ele se encontra consegue prover qualidade para o uso de sua aplicação. As consultas sobre a base de conhecimento são realizadas utilizando SPARQL.

Analisando as características de MonONTO apresentadas, é possível concluir que existe um esforço em relacionar conceitos de qualidade de rede com conceitos de qualidade de aplicação e satisfação do usuário, ou seja, a ontologia permite especificar e relacionar requisitos de qualidade de duas camadas da pilha TCP/IP (rede e aplicação), e também, de certa forma, requisitos de QoE. A comparação de requisitos, no entanto, é feita somente a nível de rede.

O objetivo dessa dissertação, no entanto, é fornecer uma base para a especificação de qualidade em todos os níveis de rede, incluindo todas as camadas da pilha TCP/IP, possibilitando até a inclusão de novas camadas. Comparando esse objetivo com os modelagem feita em MonONTO, ainda faltam conceitos essenciais para que seja o objetivo seja alcançado. Conceitos de aplicação e de rede diferem apenas no superconceito do qual derivam; não existe nenhum recurso mais elaborado para distinguí-los, como por exemplo um conceito de camada de rede. Para determinados aplicativos, as configurações do protocolo utilizado na camada

de transporte também pode influenciar na qualidade percebida (por exemplo, o tamanho da janela de congestionamento pode influenciar na taxa de transmissão), e MonONTO não permite especificar essa relação.

4.1.3 SLA Ontology

(GREEN, 2006) propõe um conjunto de ontologias para formalização de SLAs. Esse conjunto é formado por várias ontologias de base, que servem de apoio para a posterior utilização na ontologia de SLA propriamente dita. Todas as ontologias são codificadas utilizando OWL 1.0, e é feito o uso de regras SWRL para impor restrições aos conceitos definidos.

A *Unit Ontology* é a primeira destas ontologias, e nela são definidos conceitos para a criação de unidades de medida, através do conceito Unit. Unit é especializado em SimpleUnit, que por sua vez é especializado em Discreet, que representa unidades que possuem apenas um conjunto limitado de valores possíveis (*e.g.* valores booleanos), e Continuous, que representa os outros tipos de unidades de medida. São definidos também conceitos e relações que representam tipos de comparações possíveis entre unidades de medida, através do conceito Comparison. Nessa ontologia, as regras SWRL funcionam relacionando quais unidades podem ser comparadas entre si.

A *Time Unit Ontology* é derivada a partir de *Unit Ontology*, definindo conceitos de unidades de tempo, como Minute, Second, entre outros. Nessa ontologia, SWRL é utilizado para inferir relações entre as unidades de tempo (*e.g.* que um minuto é equivalente a sessenta segundos).

A *Temporal Unit Ontology* utiliza os conceitos da *Time Unit Ontology* e define os conceitos Interval, para definir acontecimentos que possuem um início e um fim, e Event, que define acontecimentos "instantâneos". A todo Interval

e Event estão associadas ações a serem executadas durante toda a sua duração (no caso de Interval) e no instante em que ocorre (no caso de Event), permitindo que esses conceitos sejam utilizados para sinalizar acontecimentos diversos, como violações de SLA, por exemplo.

A *Currency Ontology* utiliza conceitos da *Unit Ontology* para definir diferentes unidades de moeda. Nessa ontologia, SWRL é utilizado para definir relações entre essas unidades (*e.g.* inferir que 1 dólar australiano é equivalente a 100 centavos de dólar australiano).

A *Network Metrics Ontology* define unidades de medida da camada internet da pilha TCP/IP. Estas unidades podem ser associadas ao conceito Metric, que é utilizado para representar um parâmetro de desempenho dessa camada, como atraso. Nessa ontologia, regras SWRL são utilizadas para comparar unidades de medida (*e.g.* inferir que um byte é igual oito bits), mas não são definidas regras para comparar parâmetros de desempenho: não é possível comparar o atraso definido por provedores diferentes, por exemplo.

Finalmente, é definida a *SLA Ontology*. Essa ontologia possui conceitos mais genéricos, com o objetivo de permitir o maior reuso possível: é definido apenas o conceito SLA, que representa o SLA em si, e o conceito ServiceObligation, que representa as obrigações contratuais dos envolvidos no SLA. O SLS e seu conjunto de parâmetros de rede é relacionado a ServiceObligation.

A SLA Ontology foi criada para apoiar um projeto em que o autor encontrava-se associado, ligado ao setor de telecomunicações. Logo foram definidas também, derivadas da SLA Ontology e suas ontologias de base, ontologias de apoio ao projeto (denominado *Managing Quality of experience Delivery In New generation telecomunication networks with E-negotiation* — QDINE), definindo os tipos de SLA suportados pelo projeto, modelos de cobrança de serviços, entidades

e tipos de violação de SLA. Também são definidos exemplos de SLA inspirados no trabalho do projeto TEQUILA, já mencionado nesta dissertação.

Analisando essa ontologia, pode-se notar de imediato que ela não preve a comparação de SLAs e SLSs. Embora o esforço de criação das ontologias de base seja válido, vários usos das regras SWRL podem ser substituídos por funcionalidades de OWL 2.0, com ganho na redução da verbosidade das definições (*e.g.* na definição de intervalos). O conceito Metric, embora sendo genérico o suficiente para definição de parâmetros de desempenho em qualquer camada, foi definido especificamente dentro de uma ontologia internet, limitando o seu reuso. Por estes motivos, a SLA Ontology foi desconsiderada como base para esta dissertação.

4.2 ONTOLOGIAS DE QOS PARA WEB SERVICES

Serviços Web são sistemas computacionais que possuem uma interface de aplicação acessível através da web, utilizando tecnologias padrão como HTTP e XML. Originalmente, as tecnologias de WS oferecem apenas a interoperabilidade no nível sintático. A especificação semântica de WS é assunto de vários trabalhos de pesquisa que visam tratar o problema da interoperabilidade semântica e que formam uma nova linha de pesquisa, chamada Serviços Web Semânticos (SWS) (DAVIES; STUDER; WARREN, 2006).

4.2.1 OWL-QoS

(ZHOU; CHIA; LEE, 2004) apresenta um conjunto de ontologias para realizar a definição de QoS para WS, para facilitar a negociação de qualidade. Essas ontologias foram originalmente definidas utilizando uma linguagem de especificação de ontologias chamada DAML+OIL, precursora de

OWL, razão pela qual essa proposta foi primeiramente batizada DAML-QoS. Com a padronização de OWL e a descontinuidade de DAML+OIL, a proposta foi renomeada e suas ontologias recodificadas em OWL.

OWL-QoS provê três ontologias, referidas no trabalho como camadas, para a definição de QoS. Na *QoS Profile Layer* é definido o conceito QoSProfile, que representa uma especificação de QoS. Esse conceito é então especializado em ProviderQoS, InquiryQoS e TemplateQoS, representando a especificação do provedor, a especificação do cliente, e um modelo genérico para especificações, respectivamente.

Na *QoS Property Definitions Layer*, o conceito QoSProfile é especializado em QoSCore, QoSInput, QoSOutput, QoSPrecondition e QoSEffect. Esses conceitos representam, respectivamente, uma especificação "normal" de QoS; uma especificação de QoS de entrada de um serviço (*e.g.* taxa de bits que um WS consegue processar como entrada); uma especificação de QoS de saída de um serviço (*e.g.* taxa de bits que um WS gera como saída); as condições necessárias para que um serviço possa executar (*e.g.* um serviço pode requisitar que os dados de entrada sejam transmitidos uma quantidade X de vezes antes de poder gerar uma saída); e os "efeitos colaterais" da execução do serviço (*e.g.* reinicialização do servidor).

Estas classes são usadas como restrições de domínio em propriedades de objeto, também definidas na *QoS Property Definitions Layer*. Por exemplo, uma propriedade responseTime pode associar um QoSCore com uma métrica de QoS. Estas métricas, correspondentes as restrições de alcance das propriedades descritas, são definidas na *QoS Metrics Layer*. Nessa camada, são definidos os conceitos Metric, especializados em AtomicMetric e ComplexMetric, que presentam parâmetros de desempenho simples e complexos (formados por duas ou mais AtomicMetrics). Metrics são associados ao conceito Unit, que representa unidades de medida.

Nessa proposta, especificações de QoS são definidas simultaneamente em duas camadas da ontologia: na *QoS Profile Layer* e na *QoS Property Definitions Layer*. Na *QoS Property Definitions Layer*, um provedor ou usuário que deseja criar sua especificação define novas instâncias dos conceitos QoSCore, QoSInput, QoSOutput, QoSPrecondition ou QoSEffect, associando-os a intâncias do conceito Metric, e por consequência a instâncias do conceito Unit.

Na *QoS Profile Layer* essas definições são repetidas, dessa vez utilizando apenas subconceitos. Devido às limitações de OWL, não seria possível definir intervalos para as métricas utilizadas. A solução encontrada por (ZHOU; CHIA; LEE, 2004) é o uso de restrições de cardinalidade. Por exemplo: considere o conceito ProviderAQoS, derivado de QoSProfile, representando a QoS de um provedor fictício "A". Esse conceito é associado a ResponseTimeMetric, derivado de Metric, utilizando a propriedade de objeto LessThan10MSresponseTime. Nessa propriedade, é imposta uma restrição de cardinalidade, declarando que no máximo 10 indivíduos ResponseTimeMetric podem ser relacionados a um indivíduo ProviderAQoS através dessa propriedade.

A comparação de especificações de QoS é feita computando relações de subsunção utilizando um motor de inferência: considere a especificação descrita no parágrafo anterior. Agora, considere o conceito User1QoS, derivado de QoSProfile, representando o pedido de QoS do usuário "*User1*". Se esse conceito estiver associado a ResponseTimeMetric com uma restrição de cardinalidade maior do que 10, um motor de inferência vai concluir que ProviderAQoS é um subconceito de User1QoS. Essa conclusão é usada para inferir que a QoS do provedor "A" atende o pedido de "*User1*".

Observe que a proposta ignora a semântica das restrições de cardinalidade de OWL. Como explicado no capítulo 3, estas restrições definem quantos indivíduos podem ser associados utilizando uma dada propriedade. Essa quebra da

semântica de OWL, por si só, já é suficiente para desconsiderar OWL-QoS como base para esta dissertação. Apesar da ontologia definir conceitos de unidades de medida, a modelagem exige que todos os valores sejam normalizados previamente, e os autores sugerem utilizar apenas uma unidade de medida para cada métrica. Existe, ainda, mais um problema em utilizar restrições de cardinalidade: são permitidos somente valores inteiros não negativos.

A *idéia* de modelar a comparação de especificações de QoS como um problema de herança de conceitos em ontologias, no entanto, tem um apelo muito grande. A utilização de um motor de inferência já é um pré-requisito para a validação da consistência de uma ontologia. Utilizar a computação de relações de subsunção para tirar outras conclusões além da própria subsunção em si é uma ótima utilização de recursos computacionais.

Por essa razão, essa idéia foi reutilizada na proposta desta dissertação. Como OWL 2.0 permite modelar corretamente intervalos, é descartado o uso de restrições de cardinalidade. Essa modelagem derivada é explicada em mais detalhes no capítulo 5.

4.2.2 QoSOnt

(DOBSON; LOCK, 2005) é mais uma proposta de conjunto de ontologias para modelar QoS para WS, com o intuito de facilitar a escolha dinâmica de WS similares baseado em suas propriedades não funcionais de QoS. Também especificada em OWL 1.0, essa proposta especifica uma ontologia básica, e prevê a criação de novas ontologias a partir dela.

A *Base Ontology*, como o nome diz, é a ontologia de base. Nela são definidos os conceitos QoSAttribute, especializado em MeasurableAttribute, e o conceito Metric. Na verdade, um MeasurableAttribute é um QoSAttribute associado a um conceito Metric, através da propriedade de objeto hasMetric.

Logo, o conceito MeasurableAttribute representa um atributo mensurável de um sistema.

Metric, por sua vez, é associado a um conjunto de valores, representados pelo conceito MetricValue. MetricValue, por sua vez, é associado a um valor numérico através de uma propriedade de tipo de dados, e a uma unidade, representada pelo conceito Unit. As conversões entre unidades diferentes são especificadas utilizando instâncias do conceito ConversionRate. Metric também é associado a um conceito referido em (DOBSON; LOCK, 2005) como *acceptability direction*, que representa a idéia de quais valores são preferidos para uma dada Metric: valores menores ou maiores. Na verdade, (DOBSON; LOCK, 2005) reconhece a deficiência de OWL 1.0 na definição de intervalos numéricos, e contorna essa deficiência através do conceito de *acceptability direction*, aliado ao uso de um algoritmo *ad hoc*. É inclusive feita uma menção de que o uso de DataRanges de RDF Schema seria uma maneira simples de corrigir essa deficiência, exatamente o que foi feito em OWL 2.0.

Novas ontologias devem ser especificadas para definir novos atributos de QoS. Consideremos o tempo de resposta de um WS. Para definir esse atributo, seria necessário primeiramente definir as métricas dele. Uma nova ontologia de métricas (*Metrics Ontology*) é então especificada para definir as métricas de tempo de resposta, utilizando instâncias de Metric, MetricValue e Unit. Utilizando estas métricas, é definido o atributo genérico ResponseTime, subconceito de MeasurableAttribute, e este é especificado em sua própria ontologia, *Response Time Ontology*, por exemplo. A razão pela qual atributos e métricas são especificadas em ontologias diferentes é a posterior necessidade de combinar atributos (*e.g.* o atributo de confiabilidade ou *reliability* pode ser definido como sendo uma junção dos atributos tempo de resposta e probabilidade de falha).

Para criar uma especificação de QoS, um provedor deve

instanciar indivíduos que herdam do atributo desejado e também do parâmetro específico do seu serviço, especificado utilizando a ontologia OWL-S, padrão W3C para especificação semântica de WSs.

Para um cliente escolher entre serviços similares baseado em sua oferta de QoS, foi desenvolvido um algoritmo *ad hoc*, que utiliza a ontologia para montar expressões utilizando operadores lógicos E/OU. Essas expressões são dispostas em uma árvore de *parsing*, e seus nodos são comparados, e por consequência as ofertas de QoS também. Conversões de unidade também são feitas utilizando pelo algoritmo. Esse algoritmo foi utilizado no contexto de uma ferramenta de comparação de serviços.

A contribuição dessa proposta é clara no sentido de apontar as deficiências de modelagem de OWL 1.0. A modelagem apresentada, que utiliza uma ontologia base com conceitos genéricos o suficiente para serem reaproveitados em vários contextos também é uma contribuição importante, e essa idéia também é reutilizada nesta dissertação. A ontologia QoSOnt, no entanto, não é reaproveitada, pois a modelagem que combina a comparação de intervalos como problema de subsunção de conceitos com a definição de intervalos de OWL 2.0 quebraria a compatibilidade com o algoritmo apresentado em (DOBSON; LOCK, 2005).

4.2.3 Service Level Ontology

(BLEUL; WEISE, 2005) também constitui um conjunto de ontologias para a criação de SLAs para auxiliar a escolha dinâmica de WSs. Este trabalho também evidencia a importância da criação de conceitos genéricos que permitam o reuso em várias situações diferentes.

Este trabalho especifica cinco ontologias para a definição de SLAs. A *Service Level Ontology* define os conceitos

Service-Level-Offer e Service-Level-Request, representando respectivamente a oferta do provedor e o pedido do cliente. O *workflow* proposto define que, uma vez que cliente e provedor tenham acordado em um nível de qualidade, este nível seja associado a uma instância de Service-Level-Agreement.

A *Service Level Ontology* usa, para definir parâmetros de desempenho (chamados neste trabalho de dimensões de qualidade), conceitos vindo da *QoS Ontology*. As dimensões de qualidade, por sua vez, possuem métricas que definem a unidade e o tipo de dado utilizados para medí-las. As métricas são definidas na *Metric Ontology*, e as unidades na *Unit Transformation Ontology*, que também especifica fórmulas de conversão para realizar a conversão entre unidades diferentes. Os meios de interação que podem ser utilizados por serviços para a realização da conversão de unidades, e também regras adicionais para garantias de serviço são definidas na *Grounding Ontology*.

Para definir uma especificação de QoS utilizando esta ontologia, é necessário criar instâncias de Service-Level-Offer ou Service-Level-Request, e associá-las a dimensões de qualidade da *QoS Ontology*. Essas dimensões, por sua vez, devem ser associadas a métricas da *Metric Ontology*, estes associados a valores numéricos e unidades da *Unit Transformation Ontology*.

A comparação de especificações de QoS nesse trabalho também é feita através de um algoritmo *ad hoc*, que funciona da seguinte maneira: primeiramente, todas as métricas e dimensões de qualidade são alinhadas, no sentido de descobrir quais garantias falam sobre a mesma dimensão, e quais métricas podem ser diretamente comparadas. Serviços de conversão de unidades são acessados para possíveis conversões de unidades, utilizando os conceitos da *Unit Transformation Ontology* e da *Grounding Ontology*. Uma vez que todas as dimensões estão alinhadas, é feita uma comparação quantitativa das mesmas, e escolhida a melhor especificação de

acordo.

É importante ressaltar que (BLEUL; WEISE, 2005) descreve a ontologia de maneira totalmente teórica. Somente em um trabalho derivado, (BLEUL; GEIHS, 2006), é descoberto que a ontologia é especificada em OWL 1.0 e também descobertos os detalhes sobre o algoritmo de comparação. Devido à deficiência de OWL 1.0 na definição de intervalos, são criadas propriedades de objeto para suprir essa deficiência. A utilização de OWL 2.0, no entanto, tornaria todas estas propriedades obsoletas. Novamente aqui, a modelagem que utiliza subsunção também quebraria a compatibilidade com o algoritmo de comparação descrito.

4.3 SOBRE UNIDADES DE MEDIDA

Como é possível notar, em praticamente todos os trabalhos apresentados neste capítulo, há a preocupação em expressar corretamente unidades de medida. Isso ocorre pois grande parte da motivação em utilizar ontologias para expressar qualidade vêm da necessidade de preencher a lacuna das terminologias adotadas entre diferentes provedores.

Considere um exemplo simples: a taxa de transmissão nominal de uma conexão. Um provedor pode achar comercialmente atrativo ofertar sua taxa de transmissão como sendo 1000 megabits por segundo (números maiores indicam melhor qualidade), enquanto outro provedor pode ofertar sua taxa de transmissão como sendo 1 gigabit (pode assumir que os consumidores entendam o significado do prefixo “giga”). Um rápido estudo de conversão de unidades leva à conclusão de que as duas taxas ofertadas correspondem a mesma quantidade.

No entanto, não é trivial expressar essa relação computacionalmente. Provedor e clientes devem acordar na mesma sintaxe e na mesma semântica, para tornar possível a conversão automática de unidades e a compreensão automática

da terminologia utilizada. O uso de ontologias ameniza esse problema, pois permite o mapeamento entre terminologias diferentes e a expressão de diferentes unidades de medida e os fatores de conversão necessários.

A *Measurement Units Ontology* (BERRUETA; POLO, 2008) é uma ontologia específica para a criação de unidades de medida e a representação de quantidades. A partir dos conceitos básicos presentes em MUO, e das unidades de medida listadas pelo Instituto Regenstrief nos E.U.A no *Unified Code for Units of Measure* (UCUM) (SCHADOW; MCDONALD, 2009), foi criada uma grande quantidade de indivíduos, num esforço de criação de um conjunto completo de representações das unidades de medida mais utilizadas internacionalmente em ciência, engenharia e negócios

A modelagem de unidades de medida em MUO é simples. Todo objeto possui uma quantidade física, que se expressa de duas maneiras: de maneira geral, através de tipos de quantidades físicas, como massa, peso, velocidade; e de maneira específica, através da medição dessa quantidade no objeto — o peso de uma mesa, a massa de uma pessoa, a velocidade máxima de um carro. Os principais conceitos presentes em MUO podem ser vistos na figura 4.2, extraída de (BERRUETA; POLO, 2008).

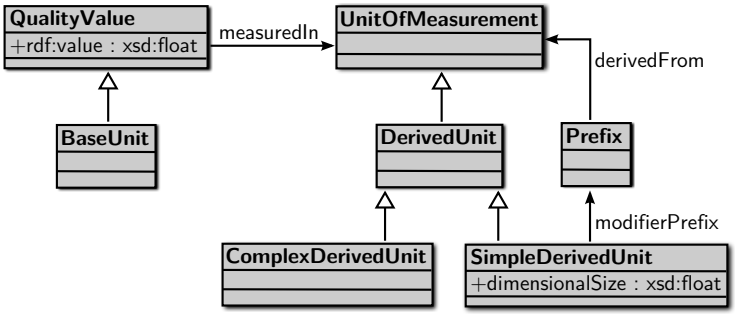

Figura 4.2: Principais classes da MUO (BERRUETA; POLO, 2008).

A expressão geral de uma quantidade física é medida uti-

lizando uma unidade, representada pelo conceito UnitOfMeasurement. Existem dois tipos de unidades de medida: unidades base (BaseUnit) e unidades derivadas (DerivedUnit). Unidades derivadas podem ser simples (SimpleDerivedUnit), resultado da adição de um prefixo à unidade base (**kilo**grama), ou através da adição de mais uma dimensão à unidade base (metro **quadrado**), ou complexas (ComplexDerivedUnit), resultado da criação de uma nova unidades utilizando duas ou mais unidades diferentes.

A expressão específica de uma quantidade física é representada pelo conceito QualityValue. QualityValue possui um valor específico de quantidade, relacionado através da propriedade de tipo de dados qualityLiteralValue, e uma unidade de medida, relacionado ao conceito UnitOfMeasurement através da propriedade de objeto measuredIn.

Desse modo, QualityValue é relacionado a um outro conceito (representando um objeto ou outra coisa qualquer) através da propriedade *qualityValue*. Essa propriedade é declarada inversamente funcional, significando que se um conceito está relacionado a dois indivíduos QualityValues diferentes através dela, implicitamente estes dois indivíduos devem representar a mesma quantidade física, possivelmente utilizando unidades diferentes.

Infelizmente, ainda não é possível expressar a conversão de unidades de medida utilizando inferência sobre Lógica Descritiva. Uma API de programação, ou o uso de regras SWRL amenizam esse problema, mas não são uma solução ideal. Enquanto SWRL assegura a interoperabilidade utilizando regras genéricas, o que não é possível utilizando uma API, as regras não asseguram computabilidade pois pode ser necessária a criação de novos indivíduos ou classes, utilizando *built-ins* personalizados.

A modelagem proposta por MUO, no entanto, é expressiva o suficiente para os fins dessa dissertação. Devido à incompatibilidade atual dos *built-ins* SWRL com OWL 2, é

sugerido o uso de programação para realizar a conversão entre unidades de medida. O problema de interoperabilidade pode ser amenizado utilizando serviços *web* como plataforma.

4.4 CONSIDERAÇÕES FINAIS

Já foi evidenciado, no texto explicativo das propostas apresentadas, os prós e contras de cada. Sumarizando, as ontologias de QoS para redes pecam ao focar especificamente na camada internet da pilha TCP/IP, embora MonONTO preveja a definição de conceitos de QoE. As ontologias de QoS para *Web Services* trazem contribuições interessantes na generalização de conceitos, embora utilizem algoritmos próprios para a comparação de especificações, com a excessão de OWL-QoS, que apresenta uma modelagem que permite a comparação dentro do contexto de inferência.

Uma ontologia de QoS mais completa deveria permitir a criação de especificações de qualquer camada de rede, além da diferenciação dos conceitos de cada camada. Combinando especificações de várias camadas, é possível a criação de especificações de QoS também mais completas. Nenhuma das propostas apresentadas neste capítulo atende a estes requisitos, portanto nesta dissertação é apresentada uma nova proposta. Essa proposta aprender com os erros e acertos das propostas apresentadas na apresentação de uma nova modelagem, descrita em detalhes no capítulo 5.

5 NETQOSONT: UMA ONTOLOGIA PARA ESPECIFICAÇÃO DE QOS

Este capítulo apresenta NetQoSOnt, uma ontologia orientada à especificação de QoS que provê uma lista extensível de parâmetros de qualidade em qualquer nível dos serviços de redes. Revisitando as melhores idéias das ontologias mostradas no capítulo 4, NetQoSOnt se propõe a ser uma evolução do cenário atual do uso de ontologias para especificação de qualidade no domínio de redes de computadores, possibilitando não somente a criação de novas especificações como também a interpretação e comparação das mesmas de maneira automatizada.

NetQoSOnt foi desenvolvida utilizando OWL 2.0, nova versão proposta pela W3C e descrita no capítulo 2, que supera limitações das versões anteriores de OWL usadas pelas ontologias de QoS mostradas no capítulo 4. Como qualquer outra ontologia, NetQoSOnt estará continuamente em desenvolvimento devido a necessidade de especificação de novos recursos que por ventura se fizerem necessários em novas aplicações. O suporte a restrições de tipos de dados possibilita a correta modelagem da idéia proposta em (ZHOU; CHIA; LEE, 2005), permitindo a comparação de parâmetros de QoS diretamente dentro do contexto de inferência em ontologias, sem a necessidade de motores de regras SWRL ou algoritmos *ad hoc*.

NetQoSOnt é uma ontologia de base a ser usada em soluções de gerenciamento de QoS oferecendo flexibilidade em termos de parâmetros de qualidade. As classes e módulos de base propostos em NetQoSOnt permitem expressar vários conceitos relacionados à especificações de QoS. Para esta dissertação, foram desenvolvidas várias modelagens de arquiteturas de QoS para diversas camadas, mas apenas de maneira simplificada. Não é objetivo desta dissertação modelar completamente nenhum padrão ou arquitetura de QoS existente

atualmente.

Todas as ontologias descritas neste capítulo podem ser encontradas na URI

http://www.lapesd.inf.ufsc.br/netqosont/

5.1 ORGANIZAÇÃO DA NETQOSONT

O desenvolvimento da NetQoSOnt é influenciado pela organização em camadas dos protocolos de rede. Estas camadas permitiram a redução da complexidade na implementação destes protocolos, assim como a diferenciação dos serviços de rede. Em NetQoSOnt, os recursos são organizados em módulos, e cada módulo é uma ontologia.

Na figura 5.1, a organização dos módulos de NetQoSOnt é descrita utilizando um Diagrama de Pacotes UML.

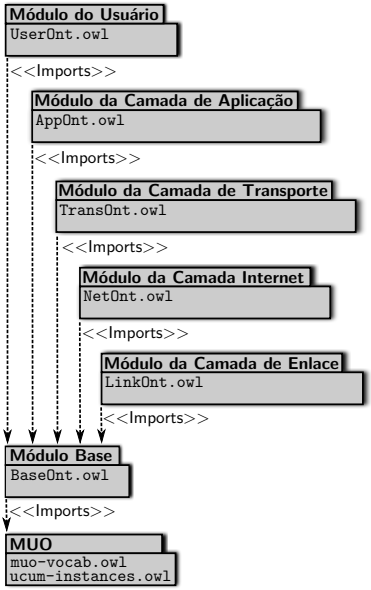

Figura 5.1: Diagrama de Pacotes UML, descrevendo a organização dos módulos da NetQoSOnt.

Nesta figura, é possível perceber que foram criados módulos para cada camada de rede. Isso reforça a diferenciação dos recursos relacionados a cada camada, sugerindo que cada recurso tem seu lugar específico.

O modelo de camadas utilizado pela NetQoSOnt é o TCP/IP. Esta escolha foi feita pelo fato do modelo TCP/IP ser considerado pelo autor o mais utilizado na prática para a categorização dos serviços de rede, principalmente por ser o modelo usado na organização na Internet.

Para auxiliar na criação de novos recursos nos módulos das camadas de rede e em módulos de terceiros, foi desenvolvido um módulo de base, que fornece recursos genéricos a serem reutilizados nos outros módulos. Como mostra a figura 5.1, o módulo base depende de uma outra ontologia, chamada MUO — *Measurement Units Ontology* (BERRUETA; POLO, 2008). Essa ontologia traz recursos de unidades de medida que serão reutilizados na definição dos parâmetros de qualidade, e foi explicada no capítulo 4. Também é fornecido um módulo para englobar recursos de QoE.

Novas ontologias de QoS deverão importar os módulos que os autores considerarem adequados e reutilizar os recursos destes na definição de novas especificações e parâmetros de qualidade, como será demonstrado mais adiante. Ontologias que representem a oferta de um provedor ou a requisição de um cliente devem especificar todas as suas classes como classes definidas, como descrito na seção 3.1.2.4: a utilização de classes definidas é o que permite a formação da hierarquia de classes utilizando inferência, possibilitando a correta modelagem da idéia proposta por (ZHOU; CHIA; LEE, 2005).

A idéia de camadas construídas umas sobre as outras é importante para modelar QoS em redes de computadores. Isto, pois os parâmetros de qualidade de uma camada podem depender (e geralmente dependem) das camadas abaixo dele. Isto é particularmente importante para o mapeamento entre parâmetros independentes de tecnologia e para parâmetros

dependentes de tecnologia. Um exemplo claro é o protocolo de transporte descrito na seção 2.5.1.2.

5.1.1 Módulo Base

O Módulo Base foi criado para agregar os recursos genéricos para a criação de parâmetros e especificações de QoS. As classes e propriedades definidas neste módulo podem ser vistas na figura 5.2.

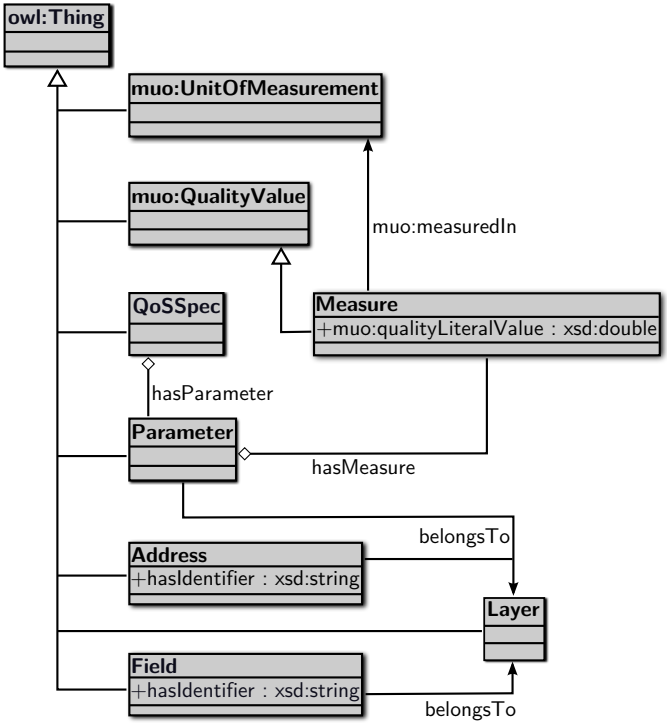

Figura 5.2: Diagrama de Classes UML descrevendo os conceitos presentes no Módulo Base da NetQoSOnt.

Na figura 5.2, a classe Layer é usada para identificar a camada a qual um recurso pertence. Um novo indivíduo Layer deve ser criado para cada módulo que representa uma camada. Por exemplo, para identificar os recursos da Camada

de Enlace, foi criado no Módulo da Camada de Enlace o Indivíduo LinkLayer. De modo análogo, um provedor que deseja criar uma nova camada (para criar todas as camadas do modelo ISO/OSI, por exemplo), deve criar um novo indivíduo Layer para esta camada. Como a classe Layer não se sobrepõe a nenhuma outra classe presente no módulo base (ou seja, nunca existirá herança múltipla envolvendo esta classe), ela foi declarada como disjunta a todas as outras classes.

Para relacionar um recurso a uma camada específica, foi criada a propriedade de objeto belongsTo. O alcance desta propriedade é a classe Layer, enquanto o domínio foi deixado indefinido. Isso permite que qualquer recurso seja relacionado a um indivíduo ou subclasse de Layer.

A classe Address foi criada como uma superclasse para as várias formas de endereçamento utilizadas em diversas camadas de rede, como para a Camada de Enlace e a Camada Internet, por exemplo. Como dentro da mesma camada podem existir várias formas de endereçamento (considere a Camada de Enlace e os padrões para rede cabeada e sem fio), no desenvolvimento de um novo módulo é sugerido ao usuário da NetQoSOnt criar uma subclasse de Address. Esse padrão de modelagem é exemplificado em mais detalhes na descrição dos módulos da Camada de Enlace e da Camada Internet.

A classe Field é destinada a ser uma superclasse utilizada para especificar campos de cabeçalho de protocolos de comunicação. Como existem diversos protocolos utilizados em várias camadas de rede, o autor acredita ser interessante existir um recurso genérico para designar campos, permitindo a modelagem de um protocolo de rede, ou somente de alguns campos de interesse de um protocolo (por exemplo campos que dizem respeito à QoS). Um exemplo de uso da classe Field é mostrado na descrição do Módulo da Camada Internet.

Em NetQoSOnt, uma especificação de QoS é expressa por uma especialização da classe QoSSpec ou seja, QoSSpec

é a superclasse de qualquer especificação de QoS. A definição de uma especificação de QoS utilizada em NetQoSOnt é simples: uma especificação de qualidade é um conjunto de parâmetros de qualidade. Como a classe QoSSpec não pode se sobrepõe a nenhum outra classe presente no módulo base, ela foi declarada como disjunta a todas as outras classes.

Para definir parâmetros de qualidade, é fornecida a classe Parameter. O significado de parâmetro em NetQoSOnt é análogo ao signicado de métrica presente nas RFCs da IETF: refere-se a um *tipo* de medição de qualidade. Atraso, por exemplo, refere-se a um tipo de medida de qualidade em redes, logo ele pode ser considerado uma métrica de qualidade, ou, utilizando a terminologia adotada em NetQoSOnt, um parâmetro.

Complementando a classe Parameter, é fornecida a classe Measure. O significado de medição complementa o significado de parâmetro atuando como uma forma de persistir os valores que realmente dão substância à uma especificação de qualidade. Por exemplo, quando um provedor fornece uma conexão com um atraso de 100 milissegundos, esse valor refere-se na verdade a uma medição do parâmetro atraso, e é esse valor que dá significado à especificação de qualidade deste provedor. Assim como QoSSpec, Measure é declarada disjunta a todas as outras classes desse módulo, pois elas nunca se sobrepõem.

Quanto a sua definição formal em NetQoSOnt, Measure é uma subclasse de muo:QualityValue. São reutilizadas também as propriedades muo:measuredIn e muo:qualityLiteralValue, para definir que Measure é medida utilizando uma unidade de medida de MUO (muo:UnitOfMeasurement), e possui um valor numérico (xsd:double). Esse valor numérico pode ser representado por um valor único ou um intervalo de valores, utilizando a sintaxe de DataRange que OWL 2 importou de RDF: desse modo, é possível criar formalmente a definição de uma medição *maior que* um determinado valor, por exemplo.

Finalmente, a classe Parameter é relacionada a classe Measure utilizando a propriedade de objeto hasMeasure. É fornecida também a propriedade inversa de hasMeasure, measureOf.

Os demais módulos de NetQoSOnt definem especializações da classe Parameter para definir parâmetros de cada uma das camadas de rede. Já foram definidas especializações de Parameter nos módulos fornecidos. A partir desta abordagem, e como será exemplificado nas seções que seguem, é possível afirmar que NetQoSOnt é extensível e permite a definição de novos parâmetros de QoS nas mais diversas camadas.

O conjunto de recursos definidos no Módulo Base é fundamental para a definição dos módulos superiores. Eles permitem, inclusive, realizar o mapeamento entre parâmetros de diferentes módulos, através do uso da equivalência de classes em ontologias. Conforme será exemplificado em mais detalhes adiante, é possível relacionar parâmetros de especificações de qualidade em camadas diferentes definindo que estas especificações são equivalentes, ou, formalmente, que as subclasses de QoSSpec em camadas diferentes são equivalentes.

5.1.2 Módulo da Camada de Enlace

Este módulo agrega recursos relacionados à camada de enlace da pilha TCP/IP, mostrados na figura 5.3.

Como se pode ver na figura 5.3, neste módulo a classe Address foi especializada para criar a classe LinkAddress que representa, de modo genérico, os tipos de endereços existentes nesta camada. A classe Parameter, por sua vez, foi especializada em LinkPerformanceParameter, de modo a definir parâmetros de desempenho específicos para a camada de enlace. Finalmente, classe Field foi especializada em LinkProtocolField, representando campos específicos de protocolos de

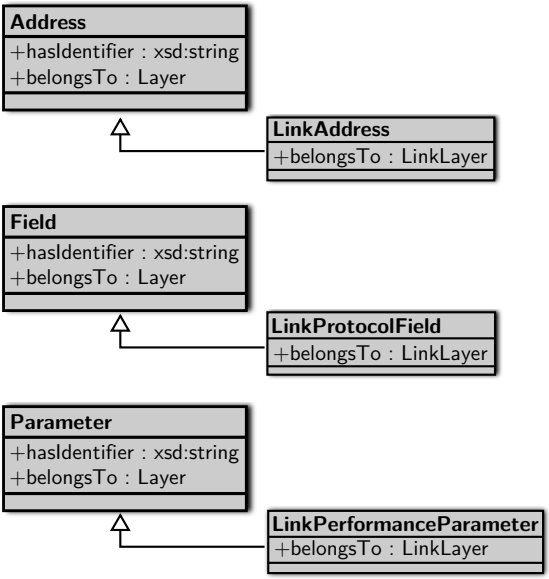

Figura 5.3: Diagrama de Classes UML descrevendo os conceitos presentes no Módulo da Camada de Enlace da NetQoSOnt.

enlace. Todas as classes estão relacionadas, através da propriedade de objeto belongsTo, ao indivíduo LinkLayer (que pertence a classe Layer), e essa relação é herdada por todas as subclasses, classes equivalentes e indivíduos dessas classes.

As classes Measure e QoSSpec não possuem subclasses nesse módulo, e em nenhum outro módulo de NetQoSOnt. Isto não quer dizer que não exista qualquer tipo de especialização dessas classes em NetQoSOnt, e sim que elas não pertecem a um módulo em particular. No caso de QoSSpec, isso ocorre pois não existe razão para atrelar uma especificação de QoS a uma camada específica, pois ela pode ter parâmetros de várias camadas diferentes. No caso de Measure, não existe especialização pois não existe unidade de medida específica somente de uma camada de rede, unidades podem ser reutilizadas em várias camadas (bits por segundo pode ser uma unidade de medida de um parâmetro de enlace ou

de internet, por exemplo).

Para demonstrar como seria feita a modelagem de um sistema de QoS em nível de enlace utilizando NetQoSOnt, será utilizado como exemplo o padrão WiMAX, descrito na seção 2.5.1.1. Para modelar este padrão, primeiramente a classe LinkPerformanceParameter deve ser especializada em TrafficPriority, MaximumSustainedTrafficRate, MaximumTrafficBurst, MinimumReservedTrafficRate, UplinkGrantSchedulingType, RequestTransmissionPolicy, MaximumLatency, SDUIndicator, ToleratedJitter, UnsolicitedGrantInterval, UnsolicitedPollingInterval, SDUSize. A classe QoSSpec é então especializada em UGS, ertPS, rtPS, nrtPS e BE, para representar as classes de serviço de WiMAX. A modelagem descrita pode ser vista da figura 5.4.

Cada CoS definida deve então ser relacionada aos parâmetros criados anteriormente através da propriedade hasParameter (por exemplo, nrtPS tem como parâmetros MinimumReservedTrafficRate, MaximumSustainedTrafficRate, TrafficPriority e RequestTransmissionPolicy). Como o padrão menciona unidades e valores possíveis para cada dos parâmetros existentes, é possível criar subclasses de Measure especificando unidades de medida para cada parâmetro (por exemplo, MaximumTrafficBurst pode ser relacionado a uma classe MaximumTrafficBurstMeasure através da propriedade hasMeasure. MaximumTrafficBurstMeasure é por sua vez relacionado a unidade de medida ucum:byte, utilizando a propriedade muo:measuredIn).

Esta modelagem é bastante simplificada mas ilustra suficientemente, para fins de demonstração, o padrão WiMAX. A figura 5.4 mostra a fundo apenas a modelagem da CoS nrtPS e da medida MaximumTrafficBurstMeasure. A modelagem das outras CoS possíveis e das medidas do padrão WiMAX são feitas de modo análogo.

Um provedor de enlace criaria suas especificações WiMAX importando a ontologia de WiMAX descrita e criando

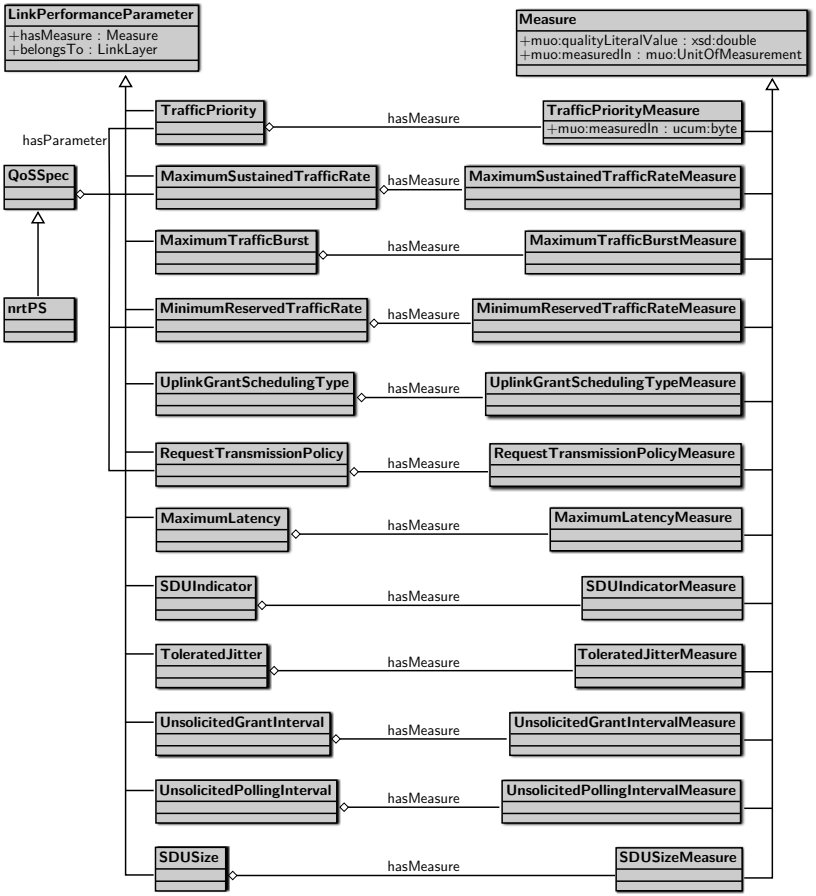

Figura 5.4: Modelagem do padrão WiMAX utilizando NetQoSOnt.

suas próprias subclasses das classes contidas nela. A principal mudança seria na definição de intervalos específicos para as subclasses de Measure, através do uso da propriedade muo:qualityLiteralValue. Esse procedimento é análogo a criação de especificações em todos os módulos e será descrito em detalhes mais adiante.

5.1.3 Módulo da Camada Internet

Analogamente ao módulo da camada de enlace, o módulo da camada Internet contém recursos relacionados à Camada Internet da pilha TCP/IP. A classe Parameter foi especializada na classe InternetPerformanceParameter, a classe Field foi especializada em InternetProtocolField e a classe Address foi especializada em InternetAddress. Graficamente, a organização destas classes é análoga àquela vista na figura 5.3. Estas novas classes estão relacionadas ao indivíduo InternetLayer através da propriedade belongsTo.

Para exemplificar a criação de parâmetros de QoS nessa camada, tomaremos como exemplo os parâmetros definidos pelo IPPM WG. As classes modeladas podem ser vistas na figura 5.5.

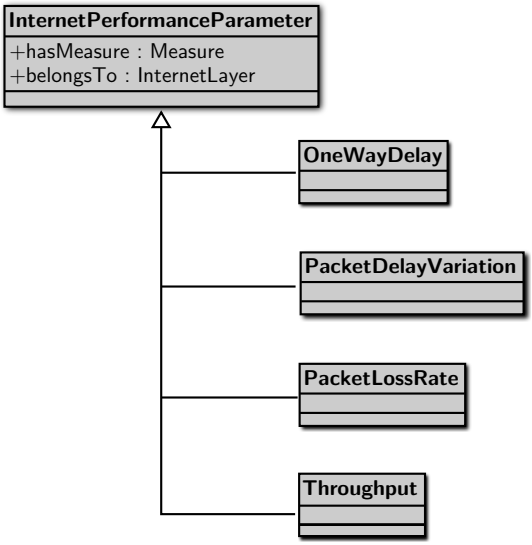

Figura 5.5: Modelagem dos parâmetros de desempenho da Camada Internet definidos pelo IPPM WG utilizando NetQoSOnt.

De acordo com as especificações do grupo, descritas na seção 2.5, a classe InternetPerformanceParameter foi especializada em PacketLossRate, OneWayDelay, PacketDelayVariation

e Throughput. Como as RFCs destes parâmetros não especificam unidades de medida padrão para estes parâmetros, não é possível especializar Measure para especificar unidades fixas, como foi feito na ontologia do padrão WiMAX. Fica a critério dos provedores e clientes então, escolher a unidade de medida de acordo com sua necessidade.

Para modelar CoSs na camada Internet, é possível então importar a ontologia de parâmetros da figura 5.5. Como exemplo serão modeladas as CoSs de DiffServ, também descritas na seção 2.5. Para isso, iremos especializar diretamente a classe QoSSpec, criando a classe PHB. Como o que identifica as CoSs em DiffServ é o campo DSCP, será exemplificado também a criação de campos de cabeçalho de protocolo. Para isso, será criada uma subclasse de InternetProtocolField para o campo DSCP. Um indivíduo DSCP é relacionado a uma PHB criando a propriedade de objeto hasDSCP. PHB é então especializada em DF, AF, EF e CS. Esta modelagem para DiffServ pode ser visualizada na figura 5.6.

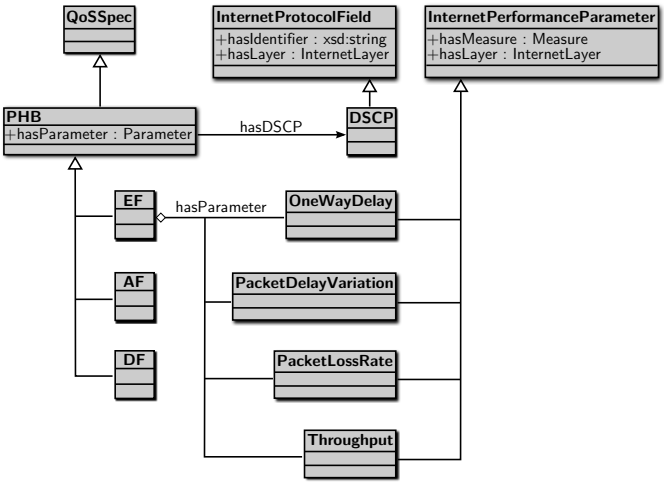

Figura 5.6: Modelagem da arquitetura DiffServ utilizando NetQoSOnt.

Na figura 5.6, é possível ver que a modelagem feita para

este exemplo também é simplificada, mas adequada para desmonstrar o uso de NetQoSOnt. Nela, é detalhada a especificação da CoS EF, mostrando os relacionamentos com os parâmetros de desempenho Internet. Como modelagem alternativa e mais detalhada, seria possível especializar a classe AF, refletindo cada um de seus casos específicos.

5.1.4 Módulo da Camada de Transporte

A exemplo dos anteriores, este módulo representa recursos relacionados à Camada de Transporte da pilha TCP/IP. Nele a classe Parameter foi especializada em TransportPerformanceParameter e a classe Field foi especializada em TransportProtocolField. O indivíduo TransportLayer foi criado para categorizar os conceitos desse módulo. Novamente, a organização das classes criadas para este modo é análoga àquela presente na figura 5.3.

Para ilustrar o uso dos recursos do módulo de transporte na modelagem de um sistema de QoS, será tomado como exemplo o protocolo QoSTP, descrito na seção 2.5.1.2. A modelagem dos parâmetros presentes em QoSTP pode ser vista na figura 5.7.

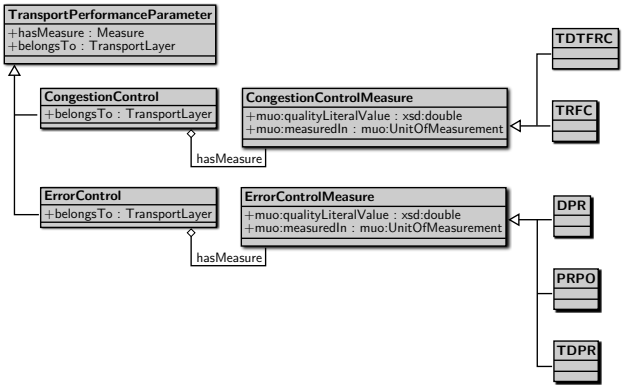

Figura 5.7: Modelagem do protocolo de transporte QoSTP utilizando NetQoSOnt: parâmetros de desempenho.

Este protocolo possui apenas dois parâmetros de qualidade, que configuram o algoritmo de controle de congestionamento e o de controle de erro. Logo, TransportPerformanceParameter é especializado em CongestionControl e ErrorControl. Cada um destes parâmetros tem poucas possibilidades de configuração, logo Measure foi primeiramente especializada em CongestionControlMeasure, por sua vez especializada em TFRC e TDTFRC, representando os tipos de controle de congestionamento. Depois, Measure foi especializada em ErrorControlMeasure, por sua vez especializada em PRPO, DPR e TDPR, representando os tipos de controle de erro.

Uma especificação de QoS completa para o protocolo QoSTP, no entanto, precisa também de parâmetros da Camada Internet. Importando a ontologia de QoSTP e a ontologia de parâmetros IPPM descrita na seção anterior, um provedor da camada de transporte poderá modelar a configuração completa do protocolo QoSTP para suas classes de serviço.

5.1.5 Módulo da Camada de Aplicação

Designado para conter recursos pertencentes à Camada de Aplicação da Pilha TCP/IP, este módulo possui especializações das classes Parameter, Field e Address, chamadas ApplicationPerformanceParameter, ApplicationProtocolField e ApplicationAddress respectivamente. Estas especializações estão relacionadas ao indivíduo ApplicationLayer.

Como já dito anteriormente na seção 2.5.1.3, a abrangência dessa camada é realmente grande: aplicativos *web*, como *webmails*, serviços multimídia como telefonia e VoD, entre outros. Os Serviços *Web* também se enquadram nesta camada, logo é possível afirmar que todas as ontologias de QoS descritas na seção 4.2 pertencem também a esta camada. De fato, estas ontologias de QoS poderiam ser remodeladas para serem módulos da NetQoSOnt, com maior ou menor grau de

compatibilidade. Essa discussão, no entanto, já foi feita no capítulo 4 e não precisa ser recapitulada aqui.

Como exemplo de criação de especificações de QoS utilizando o módulo da camada de aplicação da NetQoSOnt, será utilizado como exemplo o padrão G.711.1, descrito na seção 2.5.1.3. Inicialmente, é necessário definir duas especializações de ApplicationPerformanceParameter: SampleRate e BitRate. Como a unidade de medida desses parâmetros é fixada no padrão, a classe Measure pode ser também especializada em SampleRateMeasure e BitRateMeasure, fixando as unidades em kilohertz e kilobitspersecond, unidades derivadas segundo descrito na seção 4.3. Esta modelagem pode ser vista na figura 5.8.

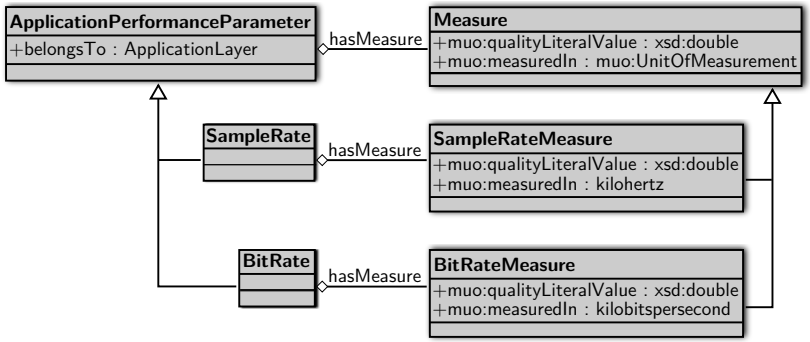

Figura 5.8: Modelagem do protocolo de aplicação G.711.1 utilizando NetQoSOnt.

Neste ponto é interessante demonstrar como funciona o uso de equivalência de conceitos em NetQoSOnt. Observando o parâmetro da taxa de bits do protocolo G.711.1, modelado com a classe BitRate, é possível cometer o equívoco de pensar que a medida desse parâmetro é equivalente à medida da vazão gerada na Camada Internet. Com uma análise mais cuidadosa, chega-se a conclusão de que na verdade a vazão gerada na Camada Internet é maior, pois o

cabeçalho e o *payload* do protocolo G.711.1 é encapsulado como *payload* no protocolo RTP, por sua vez encapsulado em UDP e somente então encapsulado em IP (ITU-T, 2008). Para maior clareza, será considerado nesta especificação apenas um tipo de qualidade, e somente o Parameter BitRate.

Como a intenção desta modelagem é demonstrar a dependência entre as camadas de rede, será suposto o uso de quatro ontologias (uma para cada protocolo de rede): a ontologia G.711.1 descrita anteriormente, duas ontologias que modelam os protocolos RTP na camada de aplicação e UDP, e a ontologia de parâmetros de desempenho IPPM na camada de transporte, descrita na seção 5.1.3. Uma organização de normalização da área de VoIP, desejando modelar os tipos de qualidade possíveis para o padrão G.711.1, importa todas estas ontologias na sua própria ontologia OWL. Para modelar a qualidade mínima oferecida, 8 kHz a 64 kbps (e analogamente todas as outras qualidades possíveis utilizando esse padrão), esse provedor deverá criar quatro subclasses de QoSSpec, cada uma contendo parâmetros de qualidade de uma ontologia utilizada, conforme pode ser visto na figura 5.9.

Para a ontologia G.711.1, foi criada a QoSSpec MinimumG7111QualitySpec. Ela está relacionada ao Parameter MinimumG7111BitRate, subclasse de BitRate. Esse Parameter tem a Measure MinimumG7111BitRateMeasure, subclasse de BitRateMeasure, que declara qualityLiteralValue = 64.0, representando o valor de 64 kbps da qualidade oferecida.

É suposta a existência de ontologias modelando a qualidade para os protocolos RTP e UDP, e também a existência de um parâmetro Throughput em cada uma destas ontologias. A organização de normalização deveria então criar QoSSpecs MinimumG7111RTPQuality e MinimumG7111UDPQuality, cada uma contendo os Parameters MinimumG7111RTPThroughput e MinimumG7111UDPThroughput, com Measures MinimumG7111RTPThroughputMeasure e MinimumG7111UDPThrough-

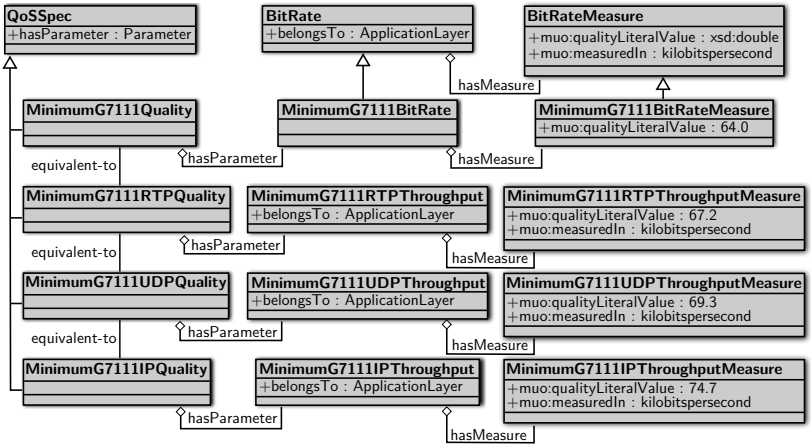

Figura 5.9: Especificação de qualidade de um provedor fictício utilizando a ontologia para o padrão G.711.1.

putMeasure. Para este exemplo, é feito um cálculo simples de vazão, em que a vazão em kbps do protocolo é igual ao tamanho do *payload* mais o tamanho do cabeçalho, e é considerado o tamanho do pacote de voz em 30 ms. Para o protocolo RTP, o cálculo da vazão resulta em aproximadamente 67,2 kilobits por segundo, logo MinimumG7111RTPThroughputMeasure especifica qualityLiteralValue = 67.2, representando uma vazão de 67,2 kbps. Para o protocolo UDP, a vazão resultante é 69,3 kilobits por segundo. Logo, MinimumG7111UDPThroughputMeasure especifica qualityLiteralValue = 69.3, representando uma vazão de 69.3 kbps.

Para a Camada Internet, foi criada a QoSSpec MinimumG7111IPQuality, que tem o Throughput MinimumG7111IPThroughput com a Measure MinimumG7111IPThroughputMeasure. Para o protocolo IP, a vazão para MinimumG7111IPThroughputMeasure resulta em 74,7 kilobits por segundo, logo é especificado qualityLiteralValue = 74.7. Lembrando novamente que para esse exemplo foram desconsiderados outros parâmetros dentro da mesma camada, para fins de simplificação.

Finalmente, para expressar que estes parâmetros de diferentes protocolos representam a mesma qualidade, este provedor fictício declara todas as suas subclasses de QoSSpecs como sendo equivalentes. Essa declaração, aliada ao uso de um motor de inferência, leva a conclusão de todas as QoSSpecs herdam todos os parâmetros declarados em todas as camadas, permitindo comparar a QoSSpec MinimumG7111-Quality, que sem o uso da inferência possui somente somente parâmetros da Camada de Aplicação, com uma QoSSpec qualquer que possua somente parâmetros da Camada Internet, através de inferência.

O uso de uma subclasse QoSSpec para cada módulo ou camada não é desnecessário: enfatiza a separação dos recursos. Com o uso da inferência, todas as relações necessárias entre eles são calculadas posteriormente.

5.1.6 Módulo do Usuário

Último módulo presente em NetQoSOnt, contém recursos para a criação de especificações de QoE. A princípio, apenas a classe Parameter é especializada nesse módulo (na classe UserPerformanceParameter relacionada ao indivíduo UserLayer), uma vez que QoE não é associada diretamente a protocolos de aplicação ou Internet, e sim à impressão que o usuário tem durante a utilização desses. Logo, não existe a necessidade de especializar Field ou Address.

Essa impressão do usuário sobre o serviço mencionada é plenamente subjetiva, embora existam maneiras de representá-la quantitativamente, discutidas na seção 2.5.1.4. Como exemplo de modelagem de especificação de QoE utilizando NetQoSOnt, será utilizado a *Mean Opinion Score* para telefone, também descrita na seção mencionada.

Para modelar os parâmetros de MOS, será utilizada a sintaxe de declaração de intervalos descrita no capítulo 3.

Isso ocorre pois na verdade os valores de MOS não são números discretos. Logo, quando se fala em MOS 1, na verdade está se falando de um valor $0 < \text{MOS} \leq 1$. Quando se fala em MOS 2, na verdade fala-se do intervalo $1 < \text{MOS} \leq 2$, e assim por diante. Uma possível modelagem para estes parâmetros pode ser vista na figura 5.10.

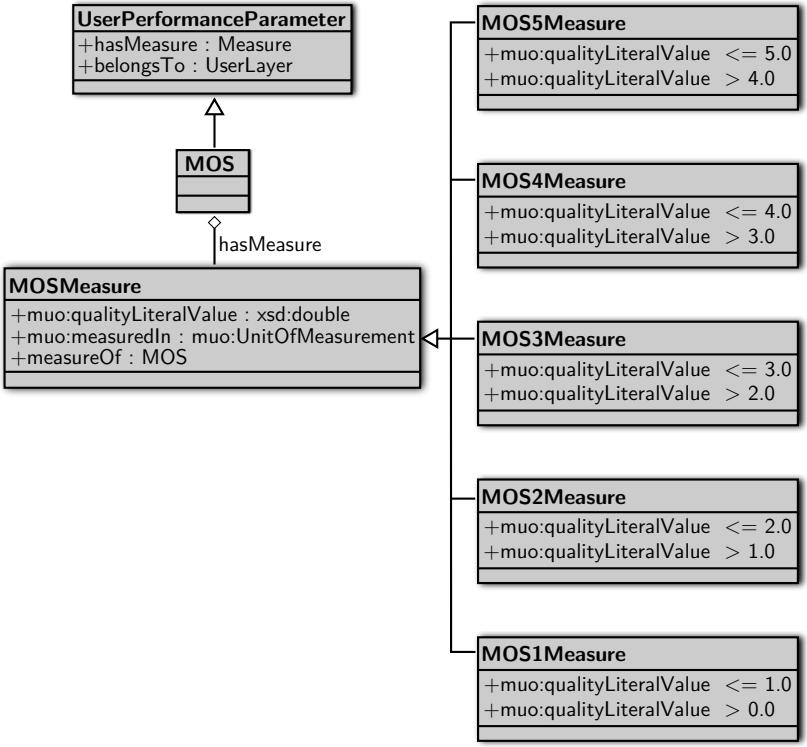

Figura 5.10: Modelagem de MOS para áudio/telefonia utilizando NetQoSOnt.

Primeiramente, UserPerformanceParameter é especializado em MOS. Lembrando sempre que o identificador do parâmetro é o URI da ontologia mais o URI do recurso, logo podemos chamar um parâmetro de MOS para áudio e outro

de MOS para vídeo com o mesmo nome, contanto que eles estejam em ontologias diferentes.

Continuando a descrição da ontologia de MOS para áudio-/telefonia, são criadas cinco Measures padrões. É criada uma superclasse comum a todas, subclasse de Measure, chamada MOSMeasure. Depois, MOSMeasure é especializada em MOS1Measure, MOS2Measure, MOS3Measure, MOS4Measure e MOS5Measure. Essas Measures são relacionadas, através da propriedade qualityLiteralValue, aos intervalos apropriados. MOS1Measure, por exemplo, será relacionado ao intervalo "> 0" e ao intervalo "≤ 1", uma vez que OWL não tem sintaxe para criar intervalos no estilo $0 <$ qualityLiteralValue ≤ 1, sendo necessário utilizar a intersecção dos intervalos mencionados. Para as outras subclasses de MOS1Measure, a modelagem é feita de modo análogo.

Como o valor de MOS não possui unidade de medida, esta foi deixada indefinida. No entanto, foi definida a relação inversa entre as MOSMeasure e MOS, para assegurar que os valores de qualityLiteralValue não sejam comparados com valores em Measures que não tem a ver com MOS durante a inferência. Essa relação inversa foi especificada utilizando a propriedade measureOf.

Observando a modelagem da *Mean Opinion Score*, surge a discussão se qualquer tipo de parâmetro, mesmo subjetivo, é representável através da modelagem proposta. No caso de MOS, o fato dos valores possíveis poderem ser divididos em intervalos definidos facilitou a modelagem. Quando se fala de MOS, também são utilizados por vezes os adjetivos "excelente" (referindo-se a MOS 5), "bom" (referindo-se a MOS 4), "razoável" (referindo-se a MOS 3), "pobre" (referindo-se a MOS 2) e "ruim" (referindo-se a MOS 1).

Estes adjetivos poderiam ser acrescentados à ontologia de MOS na forma de classes, permitindo expressar qualidade de maneira ainda mais subjetiva. Relacionando estas novas classes aos intervalos de MOS já definidos através de

equivalência, seria possível inclusive permitir a comparação entre especificações que utilizem estes parâmetros.

A associação do mapeamento em intervalos com adjetivos subjetivos na forma de classes demonstra que é possível criar parâmetros subjetivos utilizando termos familiares a um usuário leigo que ao mesmo tempo permitem a comparação por programas automatizados.

5.1.7 Comparação automatizada de parâmetros

Em NetQoSOnt, como visto anteriormente, uma especificação de QoS é definida através de uma especialização da classe QoSSpec, os parâmetros de qualidade são especializados a partir dos conceitos genéricos criados para cada camada, e as medições para cada parâmetro especializadas de Measure. NetQoSOnt modela a comparação de parâmetros como um problema de herança de conceitos da ontologia.

Desse modo, ao usar um motor de inferência e gerar uma hierarquia com os novos conceitos criados, um provedor pode descobrir quais de suas CoSs atendem a demanda consultando o motor por quais de suas subclasses de QoSSpec são subclasses da especialização de QoSSpec que representa o pedido do cliente. Esta solução será ilustrada na próxima seção.

5.2 CENÁRIO ILUSTRATIVO DE USO DA NETQOSONT

Esta seção demonstra como os conceitos presentes em NetQoSOnt tratam o problema da heterogeneidade dos parâmetros de desempenho em diferentes camadas. O cenário ilustrativo de uso da NetQoSOnt descrito aqui foi inteiramente modelado utilizando o editor de ontologias Protégé (BMIR, 2009). Para realizar a inferência sobre os conceitos modelados, foi utilizado o motor de inferência Pellet (C&P, 2009), de modo que os resultados da inferência apresentados

aqui refletem uma situação real de comparação de parâmetros.

Este cenário é composto por duas ontologias: uma representa a oferta do provedor, e outra a requisição do cliente. Considere que um provedor de serviços Internet deseja permitir que seus clientes façam invocação explícita de qualidade para uma chamada VoIP através de sua rede. Este provedor, que já possui várias classes de serviço associadas aos seus serviços, possui uma CoS chamada "*Provider Voice*", que já é alocada para tráfego VoIP. O objetivo desse provedor fictício de permitir aos seus usuários escolher a qualidade desejada antes da chamada (além de coincidir com o objetivo de demonstrar a NetQoSOnt) pode ser obter estatísticas de quantos usuários realmente desejam a qualidade fornecida pela CoS "*Provider Voice*" em função de outros fatores, como preço (não modelado nesta ontologia de exemplo).

Sabendo da versatilidade de NetQoSOnt, este provedor a utiliza para especificar todas as suas CoSs para todos os seus serviços. As suas ontologias, derivadas de NetQoSOnt e das ontologias de exemplo demonstradas anteriormente, são utilizadas para diversos fins, como persistir a configuração de seus roteadores, comparar as suas CoSs com as CoSs de outros provedores, entre outros. Estas ontologias são disponibilizadas para consulta de clientes e provedores parceiros. A modelagem da CoS "*Provider Voice*" é mostrada na figura 5.11.

A CoS "*Provider Voice*", que já é alocada ao tráfego VoIP dos clientes, como dito anteriormente, é modelada utilizando a subclasse de QoSSpec ProviderVoiceIPSpec. ProviderVoiceIPSpec está relacionada a parâmetros de desempenho IP derivados dos parâmetros contidos na ontologia do IPPM WG, mostrados na figura 5.5. Esse provedor opta por oferecer garantias de atraso, variação de atraso e perda de pacotes de 100 milissegundos, 0 milissegundos e 0 %, respectivamente. Para a especificação ProviderVoiceIPSpec, essas

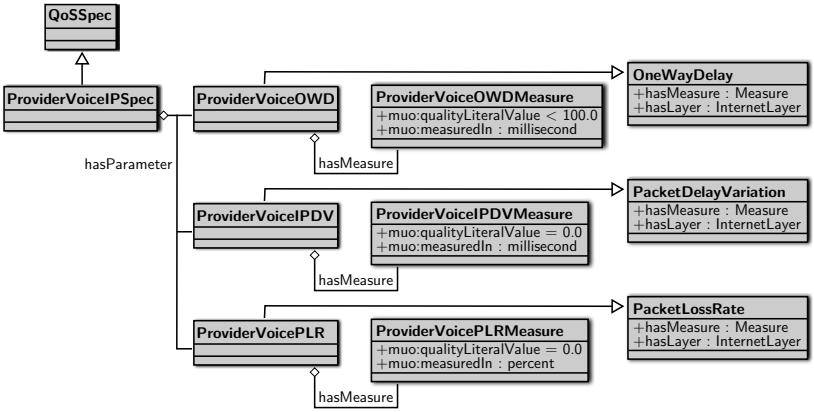

Figura 5.11: Especificação de um provedor para VoIP envolvendo parâmetros da Camada Internet utilizando NetQoSOnt.

garantias são declaradas através dos parâmetros ProviderVoiceOWD, ProviderVoiceIPDV e ProviderVoicePLR, e das medidas ProviderVoiceOWDMeasure, ProviderVoiceIPDVMeasure e ProviderVoicePLRMeasure, respectivamente.

Ao cliente é permitida então a escolha da qualidade durante a invocação do serviço, utilizando parâmetros de qualidade percebida, neste caso MOS. Esse cliente pode escolher a qualidade desejada e fazer o pedido ao provedor utilizando um protocolo de sinalização como SIP. Esse processo é descrito em detalhes em (WILLRICH et al., 2009). Basicamente, o processo envolve criar uma especificação de QoS utilizando NetQoSOnt e parâmetros importados da ontologia de MOS descrita na seção 5.1.6. Para este exemplo, será suposto uma extensão desta ontologia MOS, contento mapeamentos de valores MOS em parâmetros de qualidade da camada Internet, descritos em (ETSI, 1998).

A ontologia criada pelo cliente e sinalizada via SIP pode ser vista na figura 5.12. Supondo que o cliente deseja qualidade MOS 5, podemos ver nesta figura MOS5Measure sendo utilizada como medida do parâmetro MOS5, que é o parâ-

metro de qualidade da especificação MOS5Spec, que por sua vez representa o pedido deste cliente.

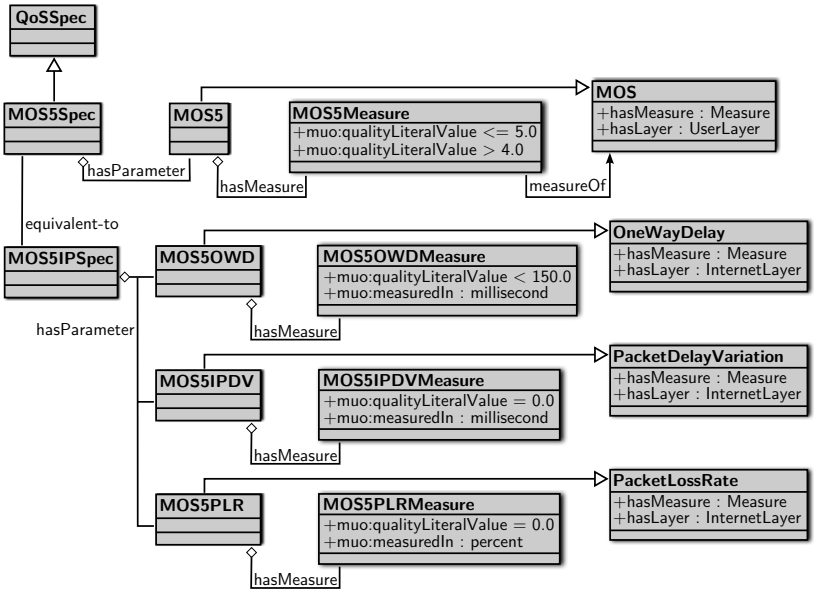

Figura 5.12: Especificação da requisição MOS de um cliente utilizando NetQoSOnt.

A partir dos mapeamentos supostos anteriormente, ao pedido do cliente é acrescentada a especificação equivalente MOS5IPSpec. Esta especificação possui três parâmetros da camada Internet, MOS5OWD, MOS5IPDV e MOS5PLR, representando o atraso, variação de atraso e perda de pacotes exigida pela MOS 5. As medidas para estes parâmetros são atraso $<$ 150 milissegundos, variação de atraso 0 milissegundos e perdas de 0 % (ETSI, 1998). Ao sinalizar a qualidade desejada via SIP, o cliente aponta sua classe MOS5Spec. Como o provedor que recebe esse pedido irá compará-lo com ProviderVoiceIPSpec?

A resposta está na relação de equivalência de MOS5Spec com MOS5IPSpec. Essa equivalência faz com que o motor

de inferência conclua que MOS5Spec e MOS5IPSpec possuem as mesmas propriedades, e por consequência, os mesmos parâmetros: MOS5Spec herda MOS5OWD, MOS5IPDV e MOS5PLR. Isso permite comparar diretamente MOS5Spec com ProviderVoiceIPSpec. A figura 5.13 mostra a ontologia do cliente carregada no editor Protégé, e a relação de equivalência descrita.

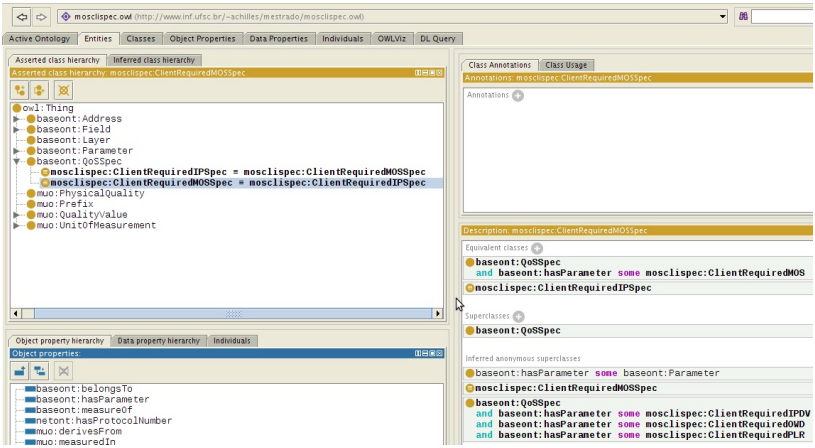

Figura 5.13: Ontologia criada pelo cliente, carregada no Editor Protégé.

Carregando a sua ontologia de oferta de qualidade e a ontologia de requisição do cliente em um motor de inferência, e ordenando ao mesmo para realizar a classificação da hierarquia, o resultado da classificação pode ser utilizado para concluir se a oferta do provedor atende ou não à requisição do cliente. Esse resultado é mostrado novamente através da interface do Protégé, na figura 5.14.

A classificação realizada pelo motor de inferência é realizada de baixo para cima (lembrando que todas as classes descritas são Classes Definidas). Para fins de clareza, será explicado apenas como ocorre a inferência para o parâmetro

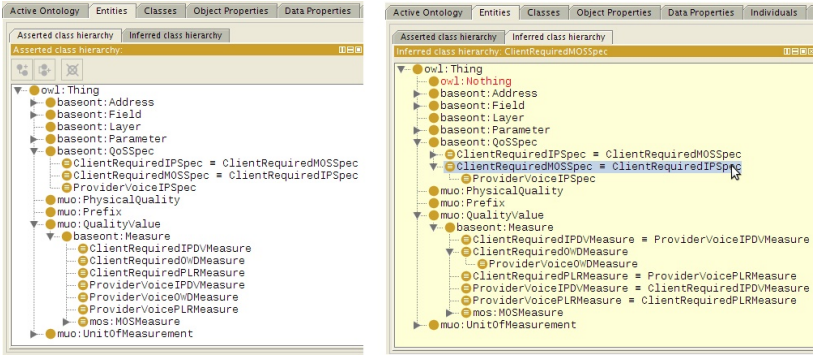

Figura 5.14: Ontologias do provedor e cliente carregadas no Protégé, antes (esquerda) e depois (direita) do uso do motor de inferência Pellet.

de atraso. Para todos os outros parâmetros das especificações, a inferência ocorre de maneira análoga.

Primeiramente, ProviderVoiceOWDMeasure é comparada com MOS5OWDMeasure. Ambas as classes definem a unidade como milissegundos, através da propriedade muo:measuredIn. MOS5OWDMeasure define em muo:qualityLiteralValue um intervalo < 150.0, enquanto ProviderVoiceOWDMeasure define um intervalo < 100.0 para a mesma propriedade. Como o segundo intervalo está contido no primeiro, o motor de inferência conclui de ProviderVoiceOWDMeasure é subclasse de MOS5OWDMeasure.

Em seguida, são comparadas ProviderVoiceOWD e MOS5OWD. ProviderVoiceOWD define como medida ProviderVoiceOWDMeasure através da propriedade hasMeasure, enquanto MOS5OWD define como medida MOS5OWDMeasure. Como ProviderVoiceOWDMeasure foi inferida como subclasse de MOS5OWDMeasure, o motor de inferência conclui que ProviderVoiceOWD é subclasse de MOS5OWD.

Para os parâmetros ProviderVoiceIPDV, ProviderVoicePLR, MOS5IPDV e MOS5PLR, são realizadas comparações semelhantes às descritas anteriormente, concluindo o motor de

inferência que ProviderVoiceIPDV e MOS5IPDV são equivalentes, e que ProviderVoicePLR e MOS5PLR também são equivalentes (uma vez que tanto para o provedor quanto para o cliente são definidos valores de medida de 0 milissegundos de variação e 0 % de perda).

Finalmente, são comparadas as classes ProviderVoiceIPSpec e MOS5Spec. ProviderVoiceOWD, ProviderVoiceIPDV e ProviderVoicePLR estão relacionadas a ProviderVoiceIPSpec através da propriedade hasParameter. MOS5OWD, MOS5IPDV e MOS5PLR estão relacionadas a MOS5Spec através da mesma propriedade. Uma vez que ProviderVoiceOWD foi inferida subclasse de MOS5OWD, e ProviderVoiceIPDV e ProviderVoicePLR foram inferidas equivalentes a MOS5IPDV e MOS5PLR respectivamente, o motor de inferência conclui que ProviderVoiceIPSpec é subclasse de MOS5Spec. Esta conclusão permite ao provedor descobrir que sua especificação ProviderVoiceIPSpec atende à especificação MOS5Spec do cliente.

5.3 PROTÓTIPO DE VALIDAÇÃO DA NETQOSONT

Nesta seção, é apresentado um protótipo usado para validar a modelagem proposta por NetQoSOnt. Nesse protótipo, as especificações descritas na seção anterior são criadas dinamicamente através da OWL API (MANCHESTER U., 2009), e posteriormente carregadas e classificadas utilizando o motor de inferência FaCT++ (TSARKOV; HORROCKS, 2009).

O protótipo possui um servidor de inferência simples, que utiliza o motor de inferência FaCT++. Esse motor de inferência foi escolhido no lugar do motor Pellet (C&P, 2009) pelo fato de apresentar um tempo de resposta melhor durante o cálculo da inferência em várias situações, durante todo o desenvolvimento desta dissertação. Em relação às funcionalidades, no entanto, o motor Pellet continua sendo superior. A modelagem do protótipo é descrita na figura 5.15.

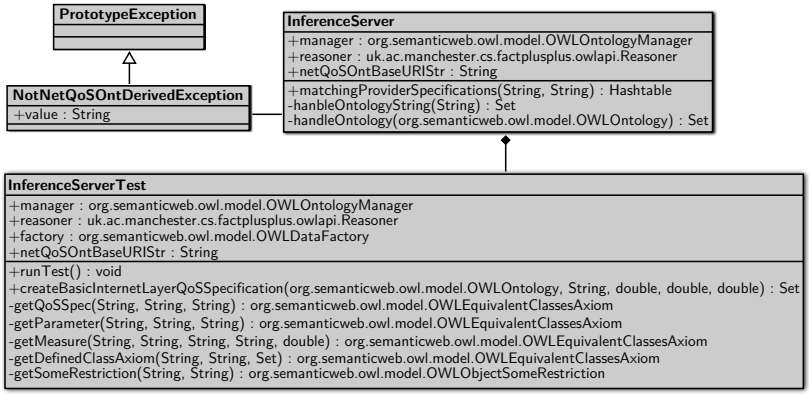

Figura 5.15: Modelagem UML do protótipo de validação da proposta.

A figura 5.15 mostra várias classes da OWL API sendo utilizadas no protótipo. De fato, a OWL API 2 foi utilizada como base para o desenvolvimento do mesmo. No entanto, o código do protótipo foi todo escrito em Python, e o suporte Python da linguagem Java foi dado pelo interpretador Jython (JYTHON, 2009).

É possível notar na figura 5.15 duas classes principais: InferenceServer e InferenceServerTest. A classe InferenceServerTest foi criada apenas para testar as funcionalidades de InferenceServer, e todos os seus métodos envolvem criar dinamicamente as especificações de QoS mencionadas na seção 5.2, para depois passá-las como argumento para um objeto InferenceServer.

A classe InferenceServer possui apenas um método principal, chamado matchingProviderSpecifications, que realiza todo o trabalho. Como já explicado anteriormente, a modelagem de NetQoSOnt funciona de maneira que, após a utilização de um motor de inferência, todas as especificações do(s) provedor(es) são inferidas como subclasses (ou classes equivalentes) da especificação do cliente.

Logo, o algoritmo por trás de matchingProviderSpecifications envolve carregar a ontologia do cliente e a ontologia do provedor no motor de inferência, realizar a classificação da hierarquia, e depois consultar o motor de inferência por quais especificações do provedor que são subclasses ou classes equivalentes da especificação do cliente.

O resultado do algoritmo é uma tabela de Hash cujas chaves são as especificações do cliente e cujos valores são os listas de especificações do provedor que atendem à especificação do cliente utilizada na respectiva chave.

5.3.1 Resultados

O programa InferenceServerTest possui uma saída em modo texto, que é mostrada na figura 5.16.

```
Creating Client Ontology: \\
http://www.inf.ufsc.br/~achilles/mestrado/ \\
  inferenceservertest-client.owl
Done.
Creating Providers Ontology: \\
http://www.inf.ufsc.br/~achilles/mestrado/\\
  inferenceservertest-provider.owl
Done.
Classifying ontologies (and measuring Reasoner response \\
  time)
Done.
Classified in 1062 ms
Matches to the client specification: \\
  {ClientRequiredIPSpec=[ProviderVoiceIPSpec], \\
  ClientRequiredMOSSpec=[ProviderVoiceIPSpec]}
```

Figura 5.16: Saída em modo texto de uma das execuções do programa InferenceServerTest. Os caracteres "\\" indicam uma quebra de linha.

Como é possível perceber analisando a saída do programa, a execução do mesmo ocorreu como esperado. A representação gráfica da tabela de hash do resultado da inferência, presente na última linha da saída, mostra claramente que ProviderVoiceIPSpec foi classificada como subclasse tanto

de ClientRequiredMOSSpec quanto de ClientRequiredIPSpec (lembrando que estas últimas duas classes são equivalentes).

O tempo de classificação das ontologias mostrado na figura 5.16 não inclui o tempo de carregamento das ontologias no motor de inferência, somente o tempo de cálculo da inferência sobre as ontologias já carregadas.

Através do resultado obtido através deste protótipo, o provedor citado no exemplo da seção 5.2 pode utilizar as informações contidas na classe ProviderVoiceIPSpec para configurar sua rede para o atender ao pedido do cliente, com certeza de que irá garantir a qualidade requerida por este.

5.3.1.1 Sobre o Tempo de Resposta do Motor de Inferência

O computador onde o protótipo foi executado possui um processador AMD Turion X2 64®, com 4 GB de memória principal. As ontologias do cliente e provedor utilizam, combinadas, em torno de 20 classes, um número relativamente pequeno, mas significativo, uma vez que provedores possuem poucas classes de serviço para seu tráfego.

Para demonstrar o tempo de resposta do motor de inferência utilizado no protótipo, uma amostra pequena, de cinco execuções, foi tomada. Para todas estas execuções, as ontologias de entrada são as mesmas, que são aquelas descritas no cenário de uso. O tempo de resposta envolve apenas o tempo de classificação da hierarquia, e não o tempo de carregamento das ontologias no motor de inferência. As execuções e seus respectivos tempos podem ser vistos na tabela 5.1.

Observando a tabela 5.1, nota-se que o tempo de resposta é, em média, 768,2 milissegundos. Este é um resultado aceitável, mesmo para negociações dinâmicas. Como a OWL API permite que as relações inferidas por um motor possam ser salvas em uma nova ontologia (ou na própria ontologia de

	Execução	Tempo (Milissegundos)
	1	716
	2	631
	3	896
	4	957
	5	731
Média		786,2
Desvio Padrão		135,36

Tabela 5.1: Tempos de execução da classificação da hierarquia no protótipo de implementação.

origem), é possível criar uma espécie de *cache* das classificações já realizadas, de modo que a mesma consulta realizada sobre o *cache* teria um tempo de resposta ainda menor.

Em um exemplo como o mostrado, onde existe uma pequena quantidade de possibilidades de configuração dos parâmetros de qualidade, seria possível inclusive criar modelos genéricos de especificações completas, e guardar os resultados da inferência sobre os modelos. Cada nova requisição do cliente poderia se referir aos modelos, cujos resultados da inferência já seriam conhecidos.

5.3.2 Limitações desta Proposta

Mesmo permitindo a especificação de qualidade em vários níveis de serviços de rede com grande expressividade, ainda existem problemas que não podem ser resolvidos através da modelagem proposta por NetQoSOnt e o uso de inferência. Estes problemas podem ser resolvidos através da criação de programas especializados utilizando a OWL API, por exemplo. No entanto, a solução ideal para estes problemas seria a adição de sintaxe própria ao padrão OWL, suportada pela implementação nos motores de inferência.

A equivalência de medidas de parâmetros de qualidade em unidades diferentes é um destes problemas. Embora seja

possível expressar que duas medidas com unidades diferentes representam a mesma quantidade, ainda não é possível fazer com que um motor de inferência faça a conversão entre unidades automaticamente. Para isso, seria necessário uma sintaxe para representar algum tipo de fórmula matemática de conversão.

Outro uso para representação de fórmulas matemáticas seria a conversão direta entre especificações de qualidade. Tomando o exemplo da MOS, descrito na seção 2.5.1.4, sabemos que existe uma fórmula matemática que pode ser utilizada para converter valores de desempenho da Camada Internet em valores de MOS. Se existisse alguma sintaxe de expressões matemáticas em OWL, apoiada por implementações em motores de inferência, essa conversão seria possível, e poderia ser feita de maneira genérica.

6 CONCLUSÃO

Neste capítulo serão apresentados os resultados obtidos pela realização desta dissertação, os artigos publicados através da pesquisa realizada e as perspectivas de trabalhos futuros.

6.1 RESULTADOS OBTIDOS

Nesta dissertação foi proposta NetQoSOnt, uma ontologia para a definição de especificações de QoS que oferece flexibilidade em termos de parâmetros de qualidade. Com base nesta ontologia, podem ser definidas soluções de QoS permitindo aos clientes e usuários de serviços de rede expressar suas necessidades em termos de QoS usando parâmetros de qualidade quantitativos e qualitativos, em todos os níveis de rede, desde a qualidade percebida a parâmetros da camada de enlace.

Modelagens anteriores que abordam o mesmo problema foram avaliadas, e suas melhores contribuições utilizadas na definição da nova ontologia de QoS voltada para os serviços de redes. Além disso, foram utilizadas novas tecnologias para o desenvolvimento de ontologias, neste caso o padrão W3C de autoria de ontologias OWL 2.0 e a API de programação OWL API 2. As definições da modelagem foram descritas e sua utilização apresentada através de um exemplo de invocação explícita de qualidade em uma chamada de voz.

A ontologia proposta se limita em especificar os parâmetros e especificações de QoS, além de permitir a definição de equivalências entre especificações de diferentes níveis, possibilitando a criação de especificações mais completas, com parâmetros de várias camadas.

Apesar de NetQoSOnt ter sido utilizada para modelar padrões de arquiteturas de QoS de maneira simplificada, as

modelagens citadas tem apenas fins de demostração. A proposta desta dissertação é o conjunto de ontologias de base que permitem a criação de novas ontologias para modelar diversos tipos de especificações de QoS.

Na modelagem proposta, a comparação de parâmetros de QoS é realizada automaticamente por um motor de inferência. Ao carregar a ontologia que contêm o pedido do cliente e a ontologia que contêm as CoSs do provedor, o cálculo da nova hierarquização das classes permite ao provedor saber quais de suas CoSs atendem ao pedido do cliente simplesmente consultando o motor de inferência por quais de suas CoSs que foram inferidas como subclasses ou classes equivalentes da classe que representa especificação do cliente.

O exemplo apresentado foi reproduzido através de um protótipo, que obteve um bom tempo de resposta de inferência e demonstrou a viabilidade do uso da proposta em sistemas de gerenciamento de QoS.

6.2 TRABALHOS PUBLICADOS PELA PESQUISA REALIZADA

No curso da realização da pesquisa para a criação da ontologia proposta nesta dissertação, foram publicados trabalhos em eventos nacionais e internacionais.

Em (PRUDÊNCIO et al., 2009b) e em (PRUDÊNCIO et al., 2009a), NetQoSOnt foi apresentada como uma ontologia de especificação de QoS que permite a flexibilização em termos de parâmetros de desempenho, sendo demonstrada através da invocação implícita de qualidade em uma chamada de voz.

Em (WILLRICH et al., 2009), NetQoSOnt foi utilizada para especificar a QoS em uma invocação explícita de qualidade em uma chamada de voz através de uma extensão do protocolo de sinalização SIP.

6.3 TRABALHOS FUTUROS

Como trabalhos futuros a esta dissertação, são propostos os seguintes temas:

- A adição de recursos que representem fórmulas matemáticas, permitindo a criação de algoritmos *ad hoc* para a conversão automática de unidades de medida e o mapeamento automatizado (sem a necessidade de criação explítica de equivalências entre subclasses de QoSSpec) de especificações de QoS;
- A modelagem extensiva de padrões de arquiteturas que suportem QoS, como WiMAX e DiffServ, de parâmetros de configuração de protocolos de transporte e de padrões da camada de aplicação (CODECs, entre outros) que influenciem da qualidade percebida;
- A implantação da NetQoSOnt em um sistema real de gerenciamento de QoS.

REFERÊNCIAS BIBLIOGRÁFICAS

ALÍPIO, P.; NEVES, J.; CARVALHO, P. An ontology for network services. *Computing and Informatics*, Eslováquia, v. 26, n. 4, p. 543–561, 2007.

ALMES, G.; KALIDINDI, S.; ZEKAUSKAS, M. *A One-way Delay Metric for IPPM*. EUA, 1999. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc2679>. Acesso em: Ago. 2009.

ALMES, G.; KALIDINDI, S.; ZEKAUSKAS, M. *A One-way Packet Loss Metric for IPPM*. EUA, 1999. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc2680>. Acesso em: Ago. 2009.

BABIARZ, J.; CHAN, K.; BAKER, F. *Configuration Guidelines for DiffServ Service Classes*. EUA, 2006. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc4594>. Acesso em: Jan. 2010.

BERRUETA, D.; POLO, L. *Measurement Units Ontology*. 2008. Disponível em: <http://forge.morfeo-project.org/wiki_en/index.php/Measurement_Units_Ontology>. Acesso em: Out. 2009.

BLAKE, S. et al. *An Architecture for Differentiated Services*. EUA, 1998. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc2475>. Acesso em: Jan. 2010.

BLEUL, S.; GEIHS, K. Automatic quality-aware service discovery and matching. In: *13th Annual Workshop of HP OpenView University Association (HP-OVUA)*. Alemanha: Infocomics-Consulting, 2006. p. 109–118.

BLEUL, S.; WEISE, T. An ontology for quality-aware service discovery. In: *First International Workshop on Engineering Service Compositions (WESC)*. [S.l.: s.n.], 2005. p. 35–42.

CAI, L. et al. QoS support in wireless/wired networks using the TCP-friendly AIMD protocol. *IEEE Transactions on Wireless Communications*, IEEE, Canadá, v. 5, n. 2, p. 469, 2006.

CHEN, J.-C. et al. *Dynamic service negotiation protocol (DSNP)*. EUA, 2006. IETF Draft. Disponível em: <http://tools.ietf.org/id/draft-itsumo-dsnp-03.txt>. Acesso em: Jul. 2009.

CHIMENTO, P.; ISHAC, J. *Defining Network Capacity*. EUA, 2008. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc5136>. Acesso em: Ago. 2009.

CLARK & PARSIA, LLC. *Pellet: The Open Source OWL Reasoner*. 2009. Disponível em: <http://clarkparsia.com/pellet/>. Acesso em: Jan. 2010.

CLERCQ, J. D. et al. *PPP Diffserv SLA Negotiation*. [S.l.], 2000. IETF Draft. Disponível em: <http://tools.ietf.org/id/draft-declercq-ppp-ds-sla-negotiation-00.txt>. Acesso em: Jul. 2009.

DAVIES, J.; STUDER, R.; WARREN, P. *Semantic Web Technologies Trends and Research in Ontology-based Systems*. Reino Unido: John Wiley & Sons Ltd, 2006.

DEMICHELIS, C.; CHIMENTO, P. *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)*. EUA, 2002. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc3393>. Acesso em: Ago. 2009.

DOBSON, G.; LOCK, R. Developing an ontology for qos. In: . [S.l.: s.n.], 2005. p. 128–13. Proceedings of the 5th Annual DIRC Research Conference.

EUROPEAN TELECOMMUNICATIONS STANDARDS INSTITUTE. *Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS)*. França, 1998.

EXPOSITO, E.; SÉNAC, P.; DIAZ, M. Uml-sdl modelling of the fptp qos oriented transport protocol. In: *10th International Multimedia Modeling Conference (MMM 2004)*. Austrália: IEEE, 2004. p. 153–160.

FLORA-2 PROJECT. *FLORA-2: An Object-Oriented Knowledge Base Language*. 2009. Disponível em: <http://flora.sourceforge.net/>. Acesso em: Dez. 2009.

FLOYD, S. et al. *TCP Friendly Rate Control (TFRC): Protocol Specification*. EUA, 2008. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc5348>. Acesso em: Dez. 2009.

GODERIS, D. et al. A service-centric IP quality of service architecture for next generation networks. In: *Network Operations and Management Symposium (NOMS)*. [S.l.: s.n.], 2002. p. 139–154.

GREEN, L. J. Service level agreements: an ontological approach. In: *International Conference on Electronic Commerce (ICEC)*. Canadá: ACM, 2006. v. 156, p. 185–194.

GROSSMAN, D. *New Terminology and Clarifications for Diffserv*. EUA, 2002. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc3260>. Acesso em: Jan. 2010.

HORRIDGE, M. et al. *A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools*. 1.2. ed. Reino Unido: University Of Manchester, 2009. Disponível em: <http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/>. Acesso em: Jul. 2009.

HSIEH, H.-Y.; SIVAKUMAR, R. Parallel transport: a new transport layer paradigm for enabling internet quality of service. *IEEE Communications Magazine*, IEEE, v. 43, n. 4, p. 114–121, Abril 2005.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. *IEEE Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1*. EUA, Fev. 2005.

ITU RADIOCOMMUNICATION SECTOR. *Methodology for the subjective assessment of the quality of television pictures*. Suíça, 2002. ITU-R Recommendation BT.500-11.

ITU TELECOMMUNICATION STANDARDIZATION SECTOR. *Pulse Code Modulation for Voice Frequences*. Suíça, 1988. ITU-T Recommendation G.711.

ITU TELECOMMUNICATION STANDARDIZATION SECTOR. *Methods for subjective determination of transmission quality*. Suíça, 1996. ITU-T Recommendation P.800.

ITU TELECOMMUNICATION STANDARDIZATION SECTOR. *Wideband embedded extension for G.711 pulse code modulation*. Suíça, 2008. ITU-T Recommendation G.711.1.

ITU TELECOMMUNICATION STANDARDIZATION SECTOR. *The E-model: a computational model for use in transmission planning*. Suíça, 2009. ITU-T Recommendation G.107.

JINGWEN, J.; NAHRSTEDT, K. QoS specification languages for distributed multimedia applications: A survey and taxonomy. *IEEE Multimedia*, IEEE, v. 11, n. 3, p. 74–87, Jul.-Set. 2004.

JYTHON TEAM. *Jython: Python for the Java Platform*. 2009. Disponível em: <http://www.jython.org/>. Acesso em: Jan. 2010.

KIFER, M.; LAUSEN, G.; WU, J. Logical foundations of object-oriented and frame-based languages. *Journal of ACM*, EUA, n. 42, p. 741–843, 1995.

KOHLER, E.; HANDLEY, M.; FLOYD, S. *Datagram Congestion Control Protocol (DCCP)*. EUA, 2006. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc4340>. Acesso em: Dez. 2009.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet*. 3a. ed. [S.l.]: Pearson Addison Wesley, 2006.

LU, G. *Communication and Computing for Distributed Multimedia Systems*. [S.l.]: Artech House Inc., 1996.

MARCHESE, M. *QoS over Heterogeneous Networks*. Reino Unido: John Wiley & Sons, Ltd., 2007.

MESCAL CONSORTIUM. *D1.3: Final specification of protocols and algorithms for inter-domain SLS management and traffic engineering for QoS-based IP service delivery*. Reino Unido, Jun. 2005. Relatório Técnico.

MORAES, P. et al. MonONTO: A domain ontology for network monitoring and recommendation for advanced internet applications users. In: *Network Operations and Management Symposium (NOMS) Workshops. IEEE*. Brasil: IEEE, 2008. p. 116–123.

NGUYEN, T. et al. COPS-SLS: a service level negotiation protocol for the internet. *IEEE Communications Magazine*, IEEE, v. 5, n. 40, p. 158–165, Maio 2002.

OBJECT MANAGEMENT GROUP. *Ontology Definition Metamodel*. E.U.A, 2009.

PARK, K. I. *QoS in Packet Networks*. EUA: Springer Science + Business Media, Inc., 2005.

PAXSON, V. et al. *Framework for IP Performance Metrics*. EUA, 1998. IETF RFC. Disponível em: <http://tools.ietf.org/html/rfc2330>. Acesso em: Ago. 2009.

PRUDÊNCIO, A. C. et al. NetQoSOnt: uma Ontologia para a Especificação Semântica de QoS em Redes de Computadores. In: *Workshop de Gerência e Operação de Redes e Serviços*. Brasil: Sociedade Brasileira de Computação, 2009.

PRUDÊNCIO, A. C. et al. Quality of Service Specifications: A Semantic Approach. In: *International Symposium on Network Computing and Applications*. E.U.A: IEEE Computer Society, 2009.

SALSANO, S. et al. *Definition and usage of SLSs in the AQUILA consortium*. Itália, Nov. 2000. Internet Draft.

SCHADOW, G.; MCDONALD, C. J. *The Unified Code for Units of Measure*. 2009. Disponível em: <http://unitsofmeasure.org/>. Acesso em: Jan. 2009.

STANFORD CENTER FOR BIOMEDICAL INFORMATICS RESEARCH. *The Protégé Ontology Editor and Knowledge Acquisition System*. Stanford Center for Biomedical Informatics Research, 2009. Disponível em: <http://protege.stanford.edu/>. Acesso em: Jul. 2009.

TEQUILA CONSORTIUM. *D3.4: Final System Evaluation (Part B) Final Architecture, Protocol and Algorithm Specification.* Bélgica, Out. 2002. Relatório Técnico.

TSARKOV, D.; HORROCKS, I. *FaCT++*. 2009. Disponível em: <http://owl.man.ac.uk/factplusplus/>. Acesso em: Jan. 2010.

UNIVERSITY OF MANCHESTER. *The OWL API*. Reino Unido, 2009. Disponível em: <http://owlapi.sourceforge.net/>. Acesso em: Jan. 2010.

WILLRICH, R. et al. Invocação dinâmica de serviços com qos em sessões multimídia sip. In: *International Information and Telecommunication Technologies Symposium*. Brasil: IEEE Computer Society, 2009.

WORLD WIDE WEB CONSORTIUM. *OWL Web Ontology Language Guide*. [S.l.], 2004. W3C Recommendation. Disponível em: <http://www.w3.org/TR/owl-guide/>. Acesso em: Jul. 2009.

WORLD WIDE WEB CONSORTIUM. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. [S.l.], 2004. W3C Member Submission. Disponível em: <http://www.w3.org/Submission/SWRL/>. Acesso em: Jan. 2010.

WORLD WIDE WEB CONSORTIUM. *SPARQL Query Language for RDF*. [S.l.], 2008. W3C Recomendation. Disponível em: <http://www.w3.org/TR/rdf-sparql-query/>. Acesso em: Jan. 2010.

ZHOU, C.; CHIA, L.-T.; LEE, B.-S. Daml-qos ontology for web services. In: *IEEE International Conference on Web Services*. EUA: IEEE Computer Society, 2004.

ZHOU, C.; CHIA, L.-T.; LEE, B.-S. Qos measurement issues with daml-qos ontology. IEEE Computer Society, EUA, p. 395–403, 2005.